Novel algorithms for efficiently accumulating, analysing and visualising full-waveform LiDAR in a volumetric representation with applications to forestry

submitted by

Milto Miltiadou

for the degree of Doctor of Engineering

of the

University of Bath

Centre for Digital Entertainment

and of the

Plymouth Marine Laboratory

NERC Airborne Research Facility

April 2017

Abstract

no more than 300 words

NOTES:

Blue colour: additions according to Neill's feedback,

Purple colour: addition/corrections according to Mike's comments

Red colour: notes

Gray colour: text that is going to be modified

To be added on top

Abstract

This study focuses on enhancing the visualisations and classifications of forested areas using coincident full-waveform (fw) LiDAR data and hyperspectral images. The ultimate aim is use both datasets to derive information about forests and show the results on a 3D virtual, interactive environment. Influenced by Persson et al (2005), voxelisation is an integral part of this research. The intensity profile of each full-waveform pulse is accumulated into a voxel array, building up a 3D density volume. The correlation between multiple pulses into a voxel representation produces a more accurate representation, which confers greater noise resistance and it further opens up possibilities of vertical interpretation of the data. The 3D density volume is then aligned with the hyperspectral images using a 2D grid similar to Warren et al (2014) and both datasets are used in visualisations and classifications.

Previous work in visualising fw LiDAR has used transparent objects and point clouds, while the output of this system is a coloured 3D-polygon representation, showing well-separated structures such as individual trees and greenhouses. The 3D density volume, generated from the fw LiDAR data, is polygonised using functional representation of object (FReps) and the marching cubes algorithm (Pasko and Savchenko, 1994) (Lorensen and Cline, 1987). Further, an optimisation algorithm is introduced that uses integral volumes (Crow, 1984) to speed up the process of polygonising the volume. This optimisation approach not only works on non-manifold object, but also a speed up of up to 51% was achieved. The polygon representation is also textured by projecting the hyperspectral images into the mesh. In addition, the output is suitable for direct rendering with commodity 3D-accelerated hardware, allowing smooth visualisation.

In future work, the effects of combining both hyperspectral imagery and fw LiDAR in classifications and visualisations are examined. At first, two pixel wise classifiers, a support vector machine and a Bayesian probabilistic model, will be used for testing the effects of the combination in generating tree coverage maps. Higher accuracy classification results are expected when metrics from both datasets are used together. Regarding the visualisations, the differences of applying surface reconstruction versus direct volumetric rendering will be discussed and an ordered tree structure with integral sums of the node values will be used for speeding up the ray-tracing of direct volumetric rendering and improving memory management of aforementioned optimisation algorithm with integral volumes. Further, deferred rendering is suggested for testing the visual human perception of projecting multiple bands of the hyperspectral images on the FW LiDAR

polygon representations. At the end of this project the combination of the datasets will be used along with the watershed algorithm for tree segmentation, which is useful for measuring the stem density of a forest and for tree species classifications.

from EDE:

Firstly, a new and fast way of aligning the FW LiDAR with Remotely Sensed Images has been developed in DASOS and by generating tree coverage maps it was shown that the combination of those datasets confers better remote survey results. This work was presented at the 36th ISRSE International Conference.

Secondly, automated detection of dead trees in native Australian forests has a significant role in protecting animals, which live in those trees and are close to extinction. DASOS allow the generation of 3D signatures characterising dead trees. A comparison between the discrete and FW LiDAR is performed to demonstrate the increased survey accuracy obtained when the FW LiDAR are used.

Finally, the last application is for improving visualisations for foresters. Foresters have a great knowledge about forests and can derive a wealth of information directly from visualisations of the remotely sensed data. This reduces the travelling time and cost of getting into the forests. This research optimises visualisations by using the new FW LiDAR representations and a speed of up to 51% has been achieved.

FW LiDAR has great potentials in forestry and this research has already started to have an impact in the FW LiDAR community by making those huge datasets easier to handle. DASOS is now used at Interpine Group Ltd, a world leading Forestry Company in New Zealand and it has been tested from a PhD student at Bournemouth University who looks into estimating bird distribution in the New Forest. In the future, it is expected that DASOS will be widely used in remote forest surveys (i.e. estimating the commercial value of a forest and detecting infected trees at early stages for treatment).

Acknowledgements

Above all, I would like to express my great gratitude to my industrial supervisors Dr. Michael Grant who had supported me continuously during my research and gave me the freedom to create a project of my own interest.

Then, I would like to thanks Dr. Matthew Brown, who helped me during my first years of my studies by giving me valuable and informative feedback. He was always there to keep me working on the right track.

Equally important is my current supervisor Dr. Neil D.F. Campbell and he is not to be missed from the acknowledgements.

Furthermore, special thanks are given to Dr. Mark Warren, Dr. Darren Cosker, MSc Susana Gonzalez Aracil and Dr. Ross Hill who occasionally advised me during my studies.

It further worth giving credits to my data providers, the Natural Environment Research Council's Airborne Research Facility (NERC ARF) and Interpine Group Ltd.

Last but not least, I am extremely grateful to my funding organisations, the Centre for Digital Entertainement and Plymouth Marine Laboratory, who supported financially and consequently made this research possible.

Abbreviations and Glossary

AGC	Automatic Gain Controller
ALS	Airborne Laser Scanning
APL	Airborne Processing Library
ARF	Airborne Research Facility
CG	Computer Graphics
CHM	Canopy Height Model
CUDA	parallel computing platform available on nvidia graphic cards
DASOS	($\delta\alpha\sigma o\varsigma$=forest in Greek), the open source software implemented for managing FW LiDAR data
DEM	Digital Elevation Model
DTM	Digital Terrain Model (DTM)
FW	Full-Waveform
GB	Gigabyte
GPU	Graphics Processing Unit
LiDAR	Light Detection And Ranging
MRI	Magnetic Resonance Imaging
NASA	National Aeronautics and Space Administration
NDVI	Normalised Difference Vegetation Index
NERC	Natural Environment Research Council
NIR	Near-Infrared Region of the electromagnetic spectrum
QGIS	Quantum Geographic Information System
SIMD	Single Instruction, Multiple Data
TB	Terabyte
VIS	VIsual Spectrum
VLR	Variable Length Records
WPDF	Waveform Packet Descriptor Format
UK	United Kingdom

Publications

DASOS-User Guide, M. Miltiadou, N.D.F Campbell, M. Brown, S.C. Aracil, M.A. Warren, D. Clewley, D.Cosker, and M. Grant, Full-waveform LiDAR workshop at Interpine Group Ltd, Rotorua NZ, 2016

Improving and Optimising Visualisations of full-waveform LiDAR data, M. Miltiadou, M. Brown, N.D.F Campbell, D. Cosker, M. Grant, *EuroGraphics UK, Computer Graphics & Visual Computing*, 2016

University of Bath **Alignment of Hyperspectral Imagery and Full-Waveform LiDAR data for visualisation and classification purposes**, M. Miltiadou, M. A. Warren, M. Grant, and M. Brown, *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 7, p. 1257, 2015.

Reconstruction of a 3D Polygon Representation from Full-Wavefrom LiDAR data, M. Miltiadou, M. Grant, M. Brown, M. Warren, and E. Carolan,*RSPSoc Annual Conference, New Sensors for a Changing World*, 2014.

Awards

EDE and Ravenscroft Prize - Finalist: Selected as one of the five finalists for this is a prestigious prize that recognises the work of best postgraduate researchers.

Student Poster Competition at Silvilaser.

Conference Presentations

Remote Sensing Cyprus (RSCy) Conference, 2017 , Paphos, Cyprus - Oral Presentation

ForestSAT Conference,2016 , Santiago, Chile - Oral Presentation

Computer Graphics & Visual Computing (CGVC),2016, Bournemouth, United Kingdom - Poster Presentation

Silvilaser, 2015, La Grant Motte, France - Oral Presentation

International Symposium of Remote Sensing of the Environment (ISRSE), 2015, Berlin, German - Oral Presentation

Remote Sensing and Photogrammetry Society (RSPSoc) Conference, New Sensors for a Changing world , 2014, Aberystwyth, United Kingdom - Oral Presentation

Workshops

Full day workshop about FW LiDAR and DASOS at *Interpine Ltd Group*, 2016, Rotorua, New Zealand

Demonstration of DASOS_v2 at the practical LiDAR session at *the NERC ARF annual workshop*, 2017, Plymouth, United Kingdom

Contents

List of Figures

Chapter 1

Introduction

1.1 Forest Monitoring: Importance and Applications

1.2 Background Information about Remote Sensing and Airborne Laser Scanning Systems

Remote sensing refers to the acquisition of information about objects, for example vegetation and archaeological monuments, without physical contact and the subsequent interpretation of that information. The sensors used to capture the information are divided into passive and active. For example satellite photography is passive because information are collected from the reflected natural sun light, while Airborne Laser Scanners (ALS) are active because they emit laser beams and collects information from the backscattered laser energy [12].

According to Wanger et al, Airborne Laser Scanning (ALS) is a growing technology used in environmental research to collect information about the Earth, such as vegetation and tree species. Comparing ALS with traditional photography, ALS is not influenced by light and it is therefore less dependent on weather conditions (ie. it collects information from below the clouds, or at night). The laser beam also partially penetrates the tree canopies allowing it to record information about the forest structure below the canopy, as well as the ground [13]. ALS methods are divided into pulse systems, which repeatedly emit pulses, and continuous wavelength systems that continuously emit light. They both acquire information from the backscattered laser intensity over time, but continuous wavelength systems are more complicated because they obtain one extra physical parameter, the frequency of the ranging signal. Further, according to Wehr and Lohr, continuous wavelength systems are 85 times less accurate than pulse systems [14].

LiDAR (Light Detection And Ranging) systems are active and pulse laser scanning systems [14]. They are divided into two groups according to the diameter of the footprint left by the laser beam on the ground, which is primarily dependent on the distance between the sensor and the target (altitude, in most remote sensing) and the beam divergence. The small-footprint group has a 0.2-3m diameter, is widely commercialised and the sensors are mostly carried on planes (ALS systems). In contrast, the large-footprint systems have a wider diameter (10-70m) and during experiments they were mostly mounted on satellites. Small-footprint systems record at higher resolution but cannot guarantee that every pulse will reach the ground due to the small diameter of their footprint, making topographic measurements difficult, and are limited to smaller survey areas due to the cost and availability of aircraft. In contrast, large-footprint scanners have wider diameters and can therefore scan wider areas with the likelihood of recording the ground to be higher [15] .

In addition, there are two types of LiDAR data: discrete and full-waveform (FW). Discrete LiDAR records a few peaks of the reflected laser intensity, while FW LiDAR stores the entire backscattered signal. The discrete LiDAR has been widely used and a 40% reduction of fieldwork has been achieved at Interpine Ltd Group, New Zealand, with that technology. Regarding the newer FW LiDAR, scientists understand their concepts and potentials but due to the shortage of available tools able to handle these large datasets, there are very few uses of FW LiDAR [16].

The design of the first FW LiDAR system was introduced in 1980s, but the first operational system was developed by NASA in 1999 [17]. The vastly increased amount of information recorded within the FW LiDAR suggests many new possibilities and problems from the point of view of image understanding, remote surveying and visualisation. As an indication, a 9.3GB discrete LiDAR from New Forest, UK, corresponds to 55.7GB of FW LiDAR.

This research is focused on the representation and efficient use of FW LiDAR data and contributes both to forestry visualisations and classifications. Two datasets are used for testing and evaluation: the New Forest and the RedGum dataset. An in-depth explanation of LiDAR systems and the specifications, differences and challenges of the two datasets are given in Section 2. An overview of the specific aims, objectives and contributions of this thesis, set in the context of these datasets, is then given at Section 3.

Chapter 2

Acquire Data

The aim of this section is to give a practical and scientific insight into the acquisition of data, because a good knowledge of these methods and their limitations is essential for understanding the related research undertaken. The relations between the two main datasets used in this project are depicted on Figure 2-1 and briefly explained here:

- The **New Forest dataset** from the UK was provided by the Natural Environment Research Council's Airborne Research Facility (NERC ARF). Measurements were collected simultaneously a Leica ALS50-II LIDAR and AISA Eagle/Hawk hyperspectral radiometers on the 8th of April in 2010. It contains Discrete LiDAR, FW LiDAR and hyperspectral images.

- The **RedGum dataset** was acquired in Australia using a Trimble AX60 integrated LIDAR/Camera instrument over the time period from the 6th of March in 2015 until the 31st of March in 2015. It was provided by the RPS Australia East Pty Ltd. Only the FW LiDAR data are used here.

The ALS data are explained first, because they are the main focus of this research, and hyperspectral imagery is towards the end of the chapter. In Section 2.1, an in-depth description of ALS systems and the differences between discrete and FW LiDAR data is given. Section 2.2 briefly discusses the binary file format of the acquired LiDAR data and Section 2.3 is a discussion on the limitations, the differences and the advantages of two LiDAR instruments; the Leica and Trimble. The essential information about the hyperspectral imagery, which is only associated with the New Forest dataset, is then covered in Section 2.4.

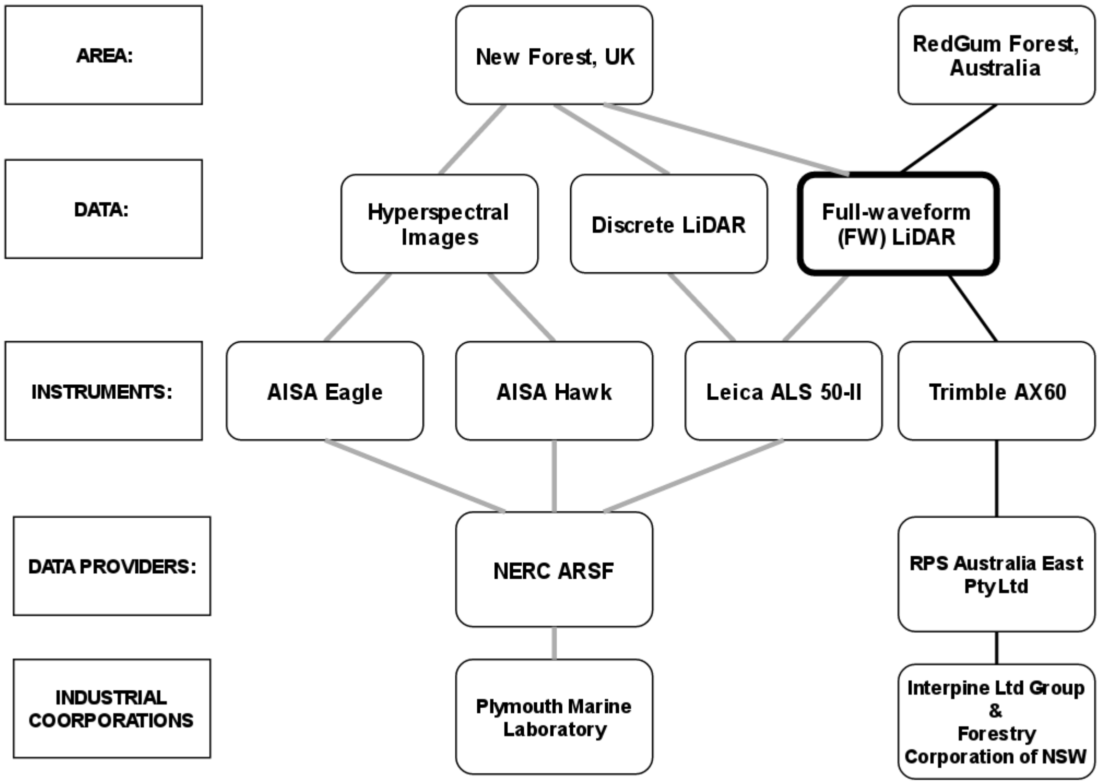
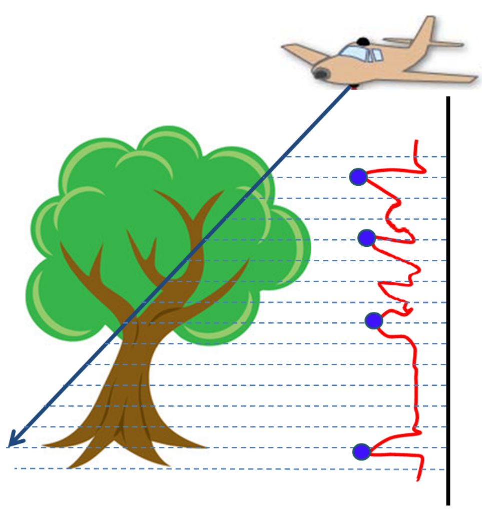
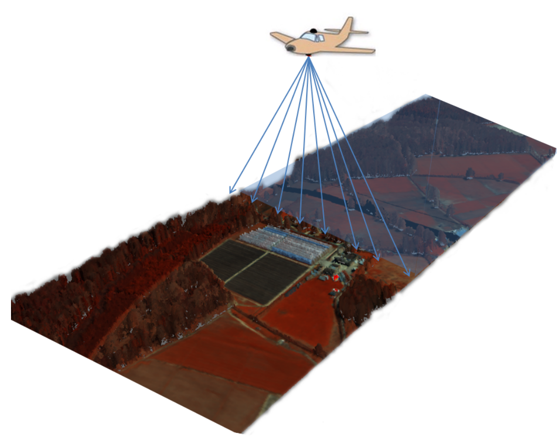


Figure 2-1: Data and Instruments



(a) A Pulse: Discrete Vs Full-Waveform



(b) A Flightline

Figure 2-2: Airborne Laser Scanning System

2.1 Airborne LiDAR systems: An in-depth Explanation

The ALS systems emit laser pulses from sensor mounted in a plane and collects information from the time-of-flight and the returned laser intensity. By the time the pulse has travelled the approximately 1-3km from the aircraft to the ground, it is roughly 20cm in width due to beam divergence. When the pulse hits an object (e.g. the forest canopy), then some of it reflects back while the rest penetrates through holes between leaves and branches. The laser pulse continues to hit structures, scattering and partially returning to the sensor until it reaches a solid barrier such as the ground and is fully blocked from further progress. The LiDAR systems record information from the backscattered laser pulse, measuring its round trip time and the returned intensity.

As mentioned at Section 1.2, there are two types of LiDAR data, discrete and FW. The discrete LiDAR observes the returned intensity signals and identifies and records a few peak intensity returns of the signal, while the FW LiDAR system digitises and stores the enitre backscattered signal into equally spaced time intervals (Figure 2-2a). The delivered data for the discrete LiDAR is a set of hit points ("returns"), which are associated with laser intensities. The world position of every return is calculated by measuring the round trip time of the laser return, giving a distance from the sensor, which is combined with the precisely known position and orientation of the aircraft/sensor (from GPS, an inertial measurement unit and precise shot direction of the laser pulse). The waveform recordings are triggered by and attached to first returns of discrete LiDAR data (to avoid sampling the uninteresting time period while the pulse travels through the atmosphere) and they are a list of intensities that correspond to the laser intensity returned over time. There is also an offset vector which defines the distance and direction between each wave sample (effectively a compression mechanism, by avoid recording the world position of every sample, replacing it with the location of the first return and this vector).

As shown in Figure 2-2b, the pulses are scanned back and forth across the landscape below (by a rotating mirror) as the plane travels forward. The scanned data has a limited maximum width according to the flight height and the field of view scan angle. During processing the track of the plane is divided into easier-to-handle pieces (flightlines) and saved into separate binary files. In this project the LAS1.3 file format is used for both datasets.

2.2 Brief Description of the LAS1.3 File Format

There are a few LiDAR file formats but the LAS1.3 was the first format to contain FW data and it is the one used to store the data for both New Forest and RedGum datasets. According to the LAS1.3 file specifications [18], a .LAS file contains information about both discrete and FW LiDAR data, with the waveform packets are attached to discrete returns and saved either internally at the end of the .LAS file or externally in a .WVS file.

As shown at (Figure 2-3) the .LAS file is divided into four sections and a brief explanation of each section is given here:

1. The **Header** contains general information about the entire flightline. For example, it includes the maximum scan angle used during the flight, whether the waveform packets are recorded internally or externally and the number of **Variable Length Records** (VLR).

2. Regarding the **VLR**, which contain arbitrary "extension" data blocks, the most important information given is the waveform packet descriptors that contain essential information on how to read the waveform packets (i.e. an ID, the number of wave samples and the size of each intensity in bits).

3. The **Point Data Records** are the discrete points and the waveforms are associated with first return discrete points. Each Point Data Record has a spatial location, an intensity and optionally a pointer to a waveform packet as well as the ID of the corresponding waveform packet descriptor.

4. The waveform packets is a list of intensities and they are either saved internally into the **Extended Variable Length Records** section of the .LAS file or inside an external .WVS file. Starting from the associated first return point, the spatial locations of the waveform packet (wave sample intensity) are calculated by adding an offset defined in the associated Point Data Record.

2.3 Leica Vs Trimble Instruments: Limitations, Differences and Advantages

As shown in Figure 2-1, the Leica ALS 50-II instrument was used to capture the LiDAR data of New Forest dataset and the Trimble AX60 for collecting the RedGum Forest FW LiDAR data. It is therefore important to clarify the differences, the limitations and the advantages of each instrument.

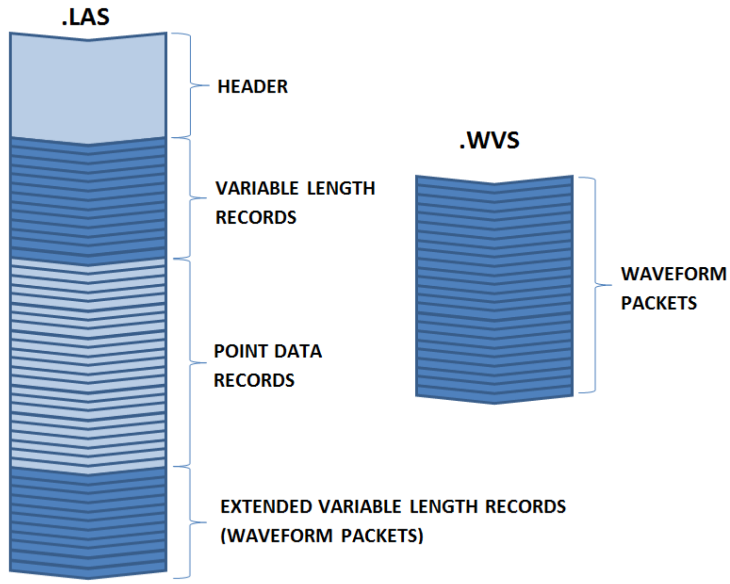


Figure 2-3: How the FW LiDAR data are stored into a binary file, according to the LAS1.3 file format specification

The Trimble performs at a pulse frequency of 400kHz, while the Leica's maximum pulse frequency is 120kHz. Nevertheless, during experiments there were occasions when the Leica discarded every other waveform due to I/O limitations despite being at or below the maximum pulse frequency [19]. The New Forest dataset has been affected by this and, on average, one third of the saved pulses only contain discrete data. We should therefore be extremely careful when comparing Discrete with FW LiDAR data. While [16] concludes that FW LiDAR data worth the extra processing because they have a better vertical profile, [20] states that extra information (the echo-width) from the FW LiDAR data are relatively unimportant. But the New Forest datasets were used for the comparison at [20] and there is no mention about the significantly less waveforms recorded in comparison to the discrete data. It is therefore suspected that their results has been affected by the missing waveforms.

Another problem with the Leica sysyem is the small dynamic range of intensities due to the number of bits used for recording them; the Leica system uses 8-bit integers (0-255 range) while the Trimble uses 16-bit integers (0-16385 range). For increased dynamic range and finer intensities without doubling storage costs, Leica introduced an Automatic Gain Controller (AGC). The AGC is an 8-bit number that defines how the recorded intensity range is shifted across a wider range of intensities. The AGC value is adjusted according to the reflected laser intensity of the previous 64 pulses and it therefore varies across a flightline. Consequently, the raw intensities are incomparable to each other and, since the relation between AGC and the intensities is not linear, the range normalisation is complicated [21] [22]. In this thesis, the intensities of the Leica system are used as boolean values (whether something existed or not, using a user-defined threshold) to quickly overcome that issue and focus on the major research objectives. Regarding the Trimble instrument, there is no AGC value because the intensities are saved into a 16bit integer and as long as the flight height is constant no normalisation is required. In a few words, the raw intensities recorded using the Leica system are not normalised and therefore not comparable to each other, while the intensities of the Trimble instrument are more meaningful.

The footprint of the laser on the ground depends on the scanning pattern of the instruments and the field of view. The sinusoidal scanning pattern of the Leica system results in a higher density of returns at the edges of the flightline. The footprint of the Trimble instrument is more equally spaced because they are scanned using a rotating polygon. The uneven density pattern of the Leica system is resolved by normalisation during the voxelisation process, but the Trimble's equally spaced pulse pattern is more prone to aliasing when voxelised. Regarding the field of view, the Leica is wider but both systems avoid large angles because otherwise data look deformed at edges of the flightlines.

Last but not least, the Trimble instrument is a native full-waveform sensor; the discrete LiDAR are produced by extracting peak points in post-processing. Therefore one of the purported advantages of a FW system, the concept of extracting a denser point clouds using Gaussian decomposition [13], does not apply in the Trimble's case. This was proven by extracting peak points from Trimble FW LiDAR data using the pulseextract from LAStools [23]; the number of points extracted was exactly the same as the number of points saved into the associated discrete LiDAR files. Therefore discrete data from the Trimble instrument are the same as those generated by echo decomposition and peak points extraction from the FW samples.

To sum up, the Trimble AX60 instrument is a newer sensor and therefore has less problems or design compromises in comparison to the Leica ALS50-II instrument. Table 2.1 summarises the differences between the two sensors.

2.4 Hyperspectral Imagery

Hyperspectral imagery has a positive impact in remote sensing because it contains information beyond human visibility. The human eye receives light from the visual spectrum into three bands (red, green and blue). The hyperspectral sensors captures a larger spectrum and divides its light components into hundreds of bands, recording

Table 2.1: Specifications of the LiDAR instruments used

Instrument Name:	**Leica ALS550-II**	**Trimble Ax60**
Scanned Area	New Forest, UK	RedGum, Australia
Year of Introduction:	Discrete LiDAR 2009 & FW LiDAR 2010	2013
Max Scan Frequency (kHz):	120	400
Recorded Intensity (bits):	8	16
AGC:	Yes	No
Scanning Pattern:	Sinusoidal	The footprints are more equally spaced on the ground
Max field of view (degrees):	75	60

this way more information than a human eye can receive [12].

Nevertheless, there are other compromises - for example, the time taken to integrate incoming light as the aircraft carrying the sensors moves. This means the raw airborne images appear deformed because the pixel length varies across the flightline. NERC-ARF geo-corrects the data using the Airborne Processing Library (APL) [24]. The processing levels are numbered. At ‘level 3’ (world coordinate system) the pixels are equally spaced and sized, which requires resampling and thus may look slightly blurred. The ‘level 1’ data (what the sensor saw) are non geo-corrected but they are associated with a file that defines the spatial location of each pixel. In this thesis, the ‘level 1’ data are used to preserve the highest possible quality.

In practise, the level 1 data are held in two files, the ‘.bil’ and the ‘.igm’. The ‘.bil’ file contains the hyperpsectral cube (Figure 2-4), all the pixel values at different wavelengths, and the .igm file gives the x, y, z coordinates of each pixel.

The number of bands and the spectrum range captured depends on the hyperspectral sensor. The data from New Forest were collected using the following instruments:

- the Eagle, which captures the visible and near infra-red spectrum (400-970nm)
- the Hawk, which covers short wave infra-red wavelengths (970-2450nm)

Both sensors divide their spectral range into 252 bands (programmable) and each band is a 2D vector as shown in Figure 2-4).

The hyperspectral images also come with a number of drawbacks. A few are mentioned here but since hyperpsectral imagery is not the main focused of the thesis there are not addressed:

- System faults sometimes occurs and the affected areas are masked out. This results in blank areas.

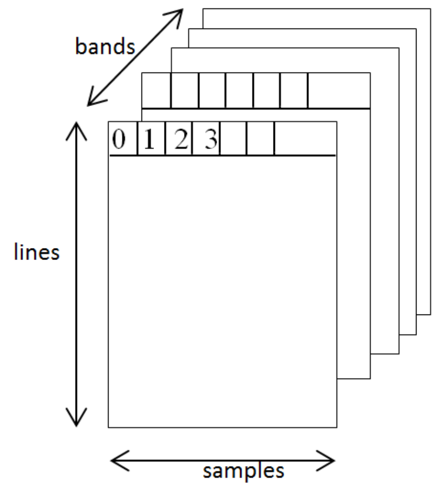


Figure 2-4: This figure shows the order of the hyperspectral pixels saved into the the binary .bil file.

- As a passive sensor, it is dependent on the sun for illumination and thus vulnerable to poor weather conditions
- Due to the high refraction of light at some wavelengths, some bands are highly influenced by humidity (i.e. wavelength 1898.33nm).

To sum up, hyperpsectral images contain information beyond the visible and they are delivered in two files, one contains the hyperspectral cube and the other one the geo-locations of each pixel. In this project, they are used in chapter (Chapter 7), where it is shown that the combination of Remote Sensing data confers better results for generating tree coverage maps.

Chapter 3

Overview of Thesis

3.1 Problem

FW LiDAR systems have been available for a number of years but there still very few uses of FW LiDAR data. NERC-ARF has been acquiring airborne data for the UK and overseas since 2010 and it has more than 100 clients of new and archived data. Many clients request FW LiDAR data to be acquired, but despite the significant number of requests,the majority of research still only uses discrete LIDAR. Some of the factors regarding this slow intakes are:

- Typically FW datasets are 5 – 10 times larger than discrete data, with data sizes in the range of 50GB – 2.5TB GB for a single area of interest. NERC-ARF's datasets are up to 100GB each because most clients are research institutes but for commercial purposes each FW dataset is a couple of TB.

- Existing workflows are only able to work with the discrete data since the increased amount of information recorded within the FW LiDAR makes handling the quantity of data very challenging.

3.2 Aims and Objectives

This thesis explores visualisation and data-understanding for FW LIDAR systems and the overarching aim is to increase the accessibility FW LiDAR in remote forest surveying. The objectives are listed in Table 3.1 and they are associated with the Sections that tackles them.

No.	Objective	Related Chapters
1	Enable forestry experts with no computer science expertise to visualise and work with the FW LiDAR data.	5
2	Enable forest understanding through 3D visualisations of FW LiDAR data.	5
3	Improve and optimise visualisations of FW LiDAR data and hyperspectral images.	6 & 7
4	Enable browsing of very large scale datasets and spectral bands in an efficient manner.	6 & 7
5	Investigate data structures for faster iso-surface extraction of large volumetric datasets and efficient management of voxels.	6
6	Estimate tree coverage and investigate the potential of integrating multiple remote sensing datasets in forestry.	7
7	Dead tree detection in comparison to human detection and remote surveying with discrete LiDAR that will benefits biodiversity management.	8
8	Research whether terrain classification can be improved by the inference of high quality 3D information, for example, using priors over the space of 3D elements.	8

Table 3.1: Values of divisible sides

3.3 Overview

*** the following text has been taken from the IAA2 funding application

To address the limitations of existing workflows for using with FW data we developed the open source software DASOS (from $\delta\alpha\sigma o\varsigma$ meaning forest in Greek) and novel algorithms that allow users, without computer science expertise, to work with and visualise large volumes of FW LiDAR data. Our open source software DASOS aims to remove the barriers preventing the use of FW LiDAR. Its contributions, and those of the new representations of the FW LiDAR, are demonstrated in three applications:

- Firstly, foresters can exploit their domain expertise to derive a wealth of information by observing the FW LiDAR data. We therefore improve visualisations for deriving information directly from the data, thus reducing travelling time and the associated expenses of getting into the forests. This cost includes appropriate cars and sometimes helicopters depending on the accessibility of the forests. While previous work on FW LiDAR visualisation talks about point cloud visualisation [25] and transparent voxels [26], DASOS is able to reconstruct the surfaces from the scanned area in 3D. This research further optimisesvisualisations by using the new FW LiDAR representations to accelerate this process by ****%. *** I will complete the percentage once related test are completed

- Secondly, a fast way of aligning the FW LiDAR with Remotely Sensed Images has been developed in DASOS. Subsequently, by generating tree coverage maps, it has been shown that the combination of these datasets confers better remote survey results [27].

- Finally, DASOS allows the generation of 3D priors. An example usage of this information is characterising dead standing Eucalyptuses, which as explained at Section 1.1 are extremely beneficial for managing biodiversity in native Australian forests. This is work in progress and a comparison between the discrete and FW LiDAR will be performed to demonstrate the increased survey accuracy obtained when the FW LiDAR is used.

In summary, FW LiDAR has great potential to improving automated surveying accuracy and consequently reduce the expensive fieldwork conducted in forestry and this research has already started to have an impact in the FW LiDAR community. DASOS is now used at Interpine Group Ltd, a world leading Forestry Company in New Zealand, and a PhD student at Bournemouth University is evaluating it for use in the estimation of bird distributions in the New Forest in the UK.

*** end of copied text

3.4 Thesis Structure

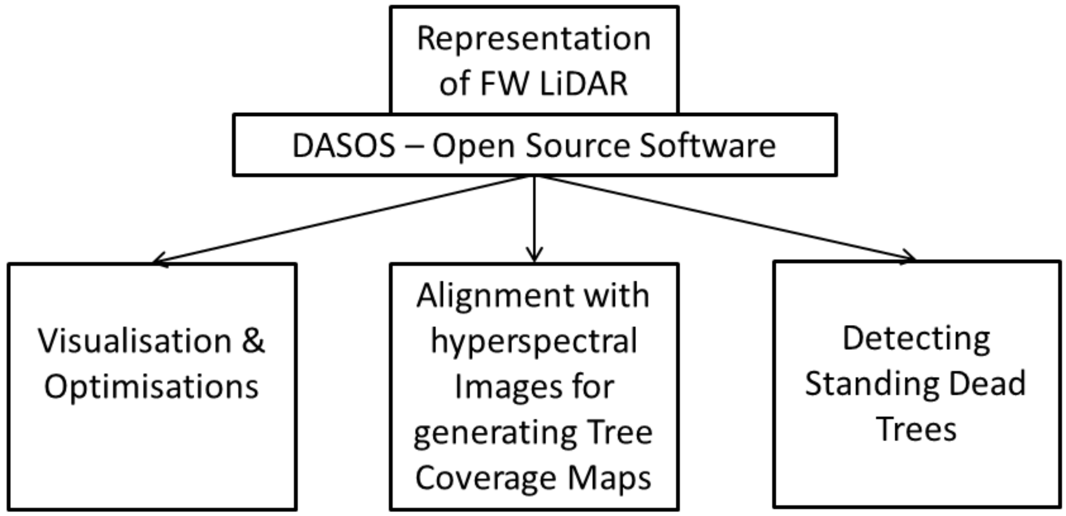


Figure 3-1: The pipeline of the thesis

Chapter 4

The open source software DASOS and the Voxelisation Approach

Chapter 5

Surface Reconstruction from Voxelised FW LiDAR Data

5.1 Introduction

To briefly summarise the previous sections, FW LiDAR data (Section 2) are laser scanning data particularly useful in Forestry, but the huge amount of information recorded make handling of the data difficult. The open source software DASOS (Section **??**) was developed along with this thesis to ease the usage of the data. DASOS voxelises (Section **??**) the data before interpretation and this approach is fundamentally different from the related, state-of-art software packages. The output of the voxelisation is a 3D discrete density volume.

In order to visualise a voxel volume, it must be rendered in some form. This chapter explains the process of reconstructing the surface of the scanned area from the 3D voxelised FW LiDAR. At first, volumetric rendering[1] approaches are briefly explained in Section 5.2. Section 5.3 gives a mathematical definition to the voxelised data, while Section 5.4 describes the actual algorithm used to extract a surface. Finally, the results are given in Section 5.5.

5.2 Rendering Approaches of Volumetric Data

Even though the concept of visualising 3D discrete density volumes (Volumetric Visualisations) is new in forestry and remote sensing, it has been widely researched in medical imaging and visual effects. There are two approaches to visualising volumetric data.

[1] Volumetric rendering refers the process of visualising 3D Volumes.

The first approach is direct rendering, which repeatedly generates 2D images according to the view point (the camera). It is like "taking photos" *** Neill: Not sure about the analogy? Can you include a diagram from a papar or textbook to show what you mean? from a camera and putting them in a sequence to produce an interactive video. An example of direct rendering approach is ray-tracing. Ray-tracing generates images by "taking photos"; rays are cast from the view point, passing through each pixel on a screen and carrying on into the volume. Intensity values are assigned to the pixels according to the nearest intersections [38]. Ray-tracing can be time expensive depending on the complexity of the scene and, for that reason, some of the literature focuses on parallellising the ray-casting process. By introducing parallelisation, real time rendering of small volumetric data (256^3) was achieved by Pfister et al. in 1999 [39]. Also, after the release of the CUDA hardware [40] (which is a parallel computing platform on recent nvidia graphics cards), Crassin et al. achieved real-time rendering of billions of voxels in 2009 [41].

The second approach is rasterisation, which is a method that maps primitive polygons (typically triangles) to pixels. It is widely used in computer games, supported directly by common hardware acceleration systems and it is significantly faster than ray-tracing. Furthermore, interactive operations (e.g. measuring the distance between two trees) are trivial calculations on primitives/polygonal meshes and they are easy to implement. In order to use this approach with volumes, they must be first converted to primitives. This is commonly accomplished by surface reconstruction, referring to the extraction of a polygonal mesh, which is a set of primitives like triangles, from the volumetric data. Constructing a surface may take several minutes, but real time visualisations of polygonal meshes are supported by free animation packages (like Blender and Meshlab), in addition to being easy to implement. So, even though it is possible to implement real-time interactive environments using direct rendering of the big voxel data, volumetric visualisation of FW LiDAR data is a new concept in remote sensing and, for simplicity, this thesis uses surface reconstruction.

5.3 Algebraic Definition of the Volume

In computer graphics, objects can be defined using a function rather than being constructed from primitives. Those objects are called either implicit or algebraic. Implicit representation of objects enables a mathematical definition of the 3D discrete density volume generated from the FW LiDAR data (Section **??**).

Algebraic objects were firstly introduced in computer graphics by Blinn in 1982 [42] to enable the definition of complex objects without saving a large amount of primitives;

in some cases, primitives cannot accurate represent a shape (e.g. a sphere cannot be represented fully by a triangle mesh). Each object is defined by a function $f(X)$ and the iso-surface value α. The iso-surface value (iso-level) defines the boundaries of the object; for an object $[f(x), a]$ every n-dimensional point X that lies on the surface of the object satisfies the condition $f(X) = \alpha$. To be more accurate, the following rules apply according to Pasko et al. [43]:

- f(X) $= \alpha$, when X lies on the surface of the algebraic object
- f(X) $> \alpha$, when X lies inside the algebraic object and
- f(X) $< \alpha$, when X lies outside the algebraic object

Regarding the algebraic representation of the 3D voxelised FW LiDAR data, X is a three dimensional point (x, y, z) representing the longitude, latitude and height respectively and $f(X)$ is a function that takes X as input and returns the accumulated intensity value of the voxel that X lies inside. Also, the iso-surface value α is a user defined parameter. Even though it closely related to the noise threshold used for filtering during voxelisation (Section **??**), it is different. The noise threshold filters low intensity samples before the volume is constructed, while the iso-surface value defines the boundaries of the object and it can be modified after the voxelisation because it doesn't affect the intensity values of the 3D voxelised FW LiDAR. Figure 5-5 demonstrates how the iso-level parameter affects the output of the surface reconstruction of the voxelised FW LiDAR data in comparison to the noise filtering.

5.4 Surface Reconstruction with the Marching Cubes Algorithm

Even though numerical implicitisation is beneficial in reducing storage memory, visualising implicit objects is not straight forward, since they contain no discrete values. As described above in rendering volumes, this problem can be addressed either by direct rendering or surface reconstruction (Section 5.2).

The Marching Cubes algorithm is an algorithm that polygonises implicit objects using a look up table. Let's assume that $f(X)$ defines an implicit object. At first the space is divided into cubes. Each cube is defined by eight corner points and each corner point lies either inside or outside the implicit object. By enumerating all the possible cases and linearly interpolating the intersections along the edges, the surface of the implicit iso-surface is constructed [44]. The output is a polygonal mesh, a number of adjacent triangles constructed according to the user-defined iso-surface value α of the implicit object.

The normals[2] are calculated afterwards. According to Lorensen and Cline [44], the normal of each vertex is calculated by measuring the local gradient change. Even though this work well on smooth object (e.g. a sphere defined by its equation), because of the high gradient changes in the voxelised FW LiDAR data this algorithm results into normals pointing into inconsistent directions. This is a problem because when the normals are not consistent, the surface of the object appears rough. For that reason, in DASOS the normal of each vertex is derived by the average normal of its adjacent triangles.

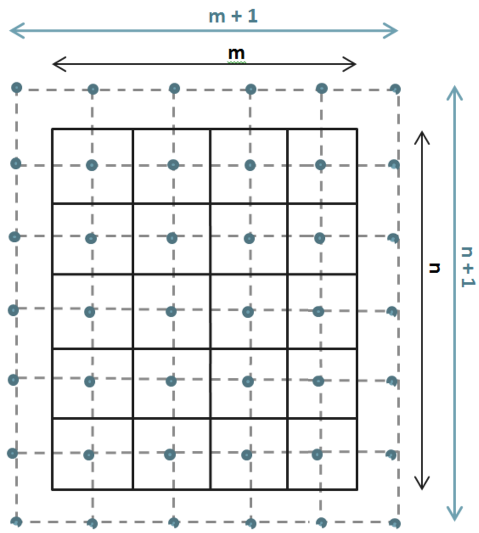
Additionally it is worth highlighting that the sampling of the Marching cubes is independent from the sampling of the 3D density volume. But consistency between the two is required to avoid artefacts. Let's assume the discrete volume has $(n \times m \times k)$ voxels, then the suggested sampling of Marching Cubes is $((n+1) \times (m+1) \times (k+1))$, as shown on Figure 5-1; the black grid represents a 2D density grid and the blue grid represents the suggested sampling of the polygonisation. Please note that every point that lies outside the volume is considered to be outside the implicit object. Figure 5-1b shows the effects of oversampling on a low resolution 3D density volume. On the right image the sampling of the volume appears as linear lines and squares on the forested areas because of the Marching Cubes' oversampling. Even though the right polygonal mesh looks blurred, it has been correctly sampled and the blur is because of the low resolution of the volume. Nevertheless there are no geometrical shapes on forested areas and once the resolution is increased then blur will disappear.

5.5 Results

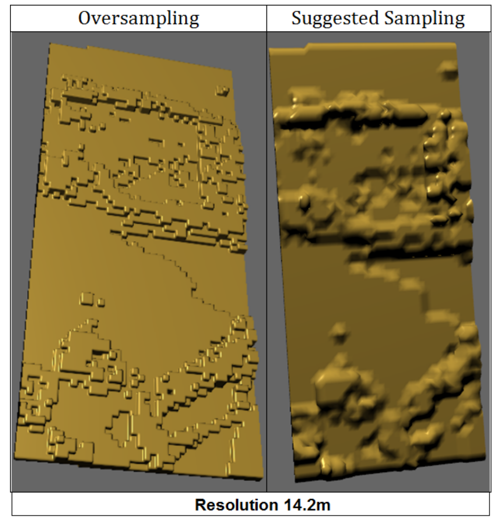
The output of DASOS is a polygonal mesh exported as an .obj file, which is a standard graphics format. The .obj files can be loaded into various animation software tools like Maya and Meshlab (Figure 5-2). Figure 5-3 shows polygonal meshes generated using NERC-ARF data from three different areas in the UK. The region of interest is also user defined. The user defines whether an entire flightline or selected area is polygonised (Figure 5-4).

Furthermore, there are three main user-defined parameters and Figure 5-5 shows how the results are affected once modified:

[2] A normal is a vector that is perpendicular to the surface of a polygonal mesh. In graphics, the normals are important for calculating light illumination and each vertex is associated with one for smooth rendering of surfaces.



(a) Suggested Sampling



(b) Effects of Oversampling

Figure 5-1: The suggested sampling during polygonisation using the Marching Cubes Algorithm

1. The voxel length controls the resolution of the output; the bigger the voxel length is the lower the resolution and the number of cubes are.

2. The iso-level is the boundary that defines whether a voxel is inside or outside the implicit object. When the iso-level is increased, the number of voxels that are considered inside the implicit object decreases. For that reason, when it is too high most of the voxels are outside the boundary and the object seems to disappear.

3. The noise level is the threshold of the low level filtering applied during voxelisation (Section **??**). If the noise level is too low, then the noise covers significant features of the data and when it is too high important information are discarded and the object seems to disappear again.

Aside from computer-based visualisation, it is even possible to 3D print the meshes using something like MakerBot. There are some difficulties as the meshes are not manifold[3] (figure 5-6). Simplification of the mesh would have eased the processing of the .obj file in MakerBot.

[3]A non-manifold polygonal object may have triangle below the outside surface of the object

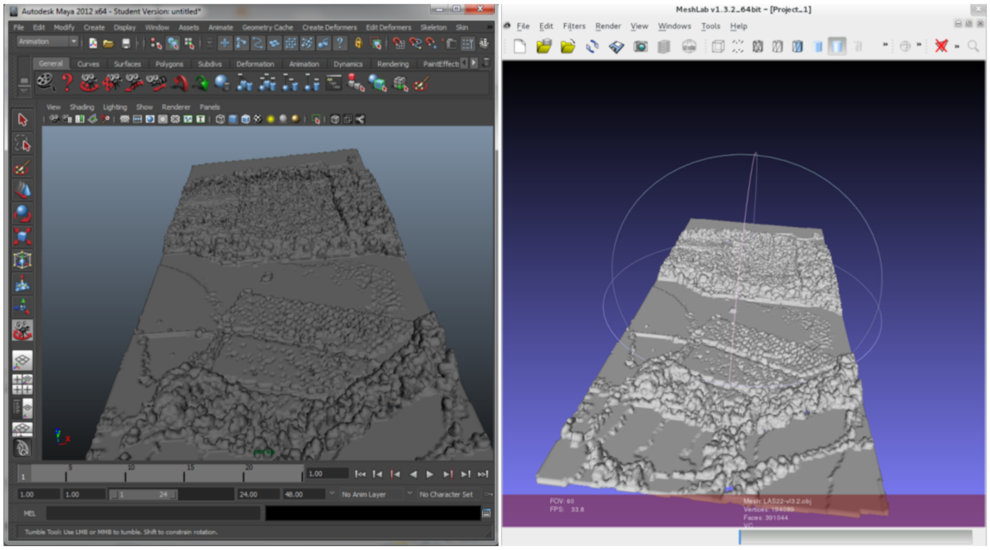


Figure 5-2: Visualising the output of DASOS into animation software packages (Maya and Meshlab)

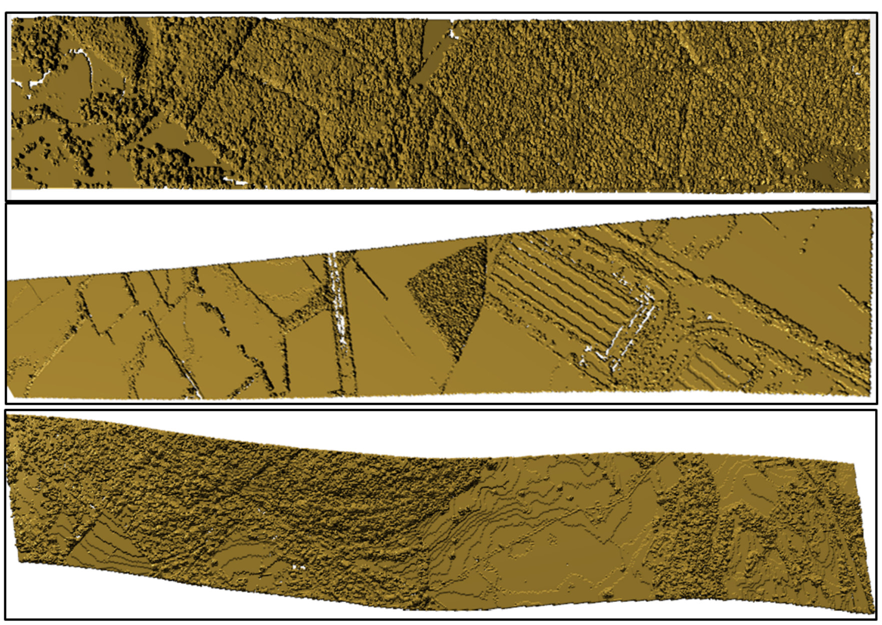


Figure 5-3: Polygonising NERC-ARF FW LiDAR data captured at different areas (New Forest, Milton Keynes and Eaves Wood)

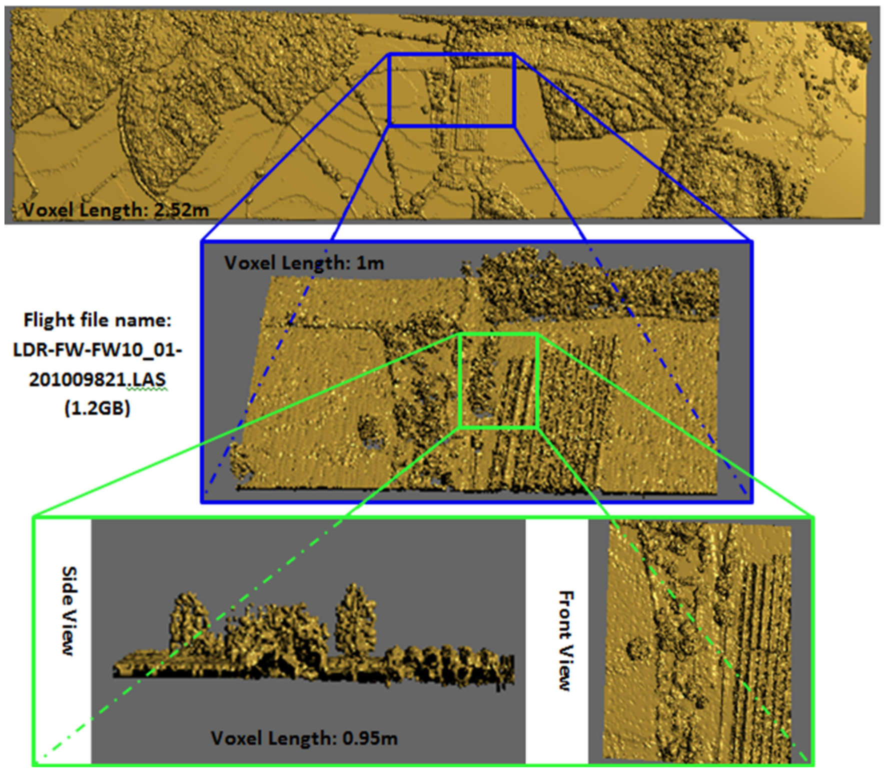


Figure 5-4: Selecting Region of Interest

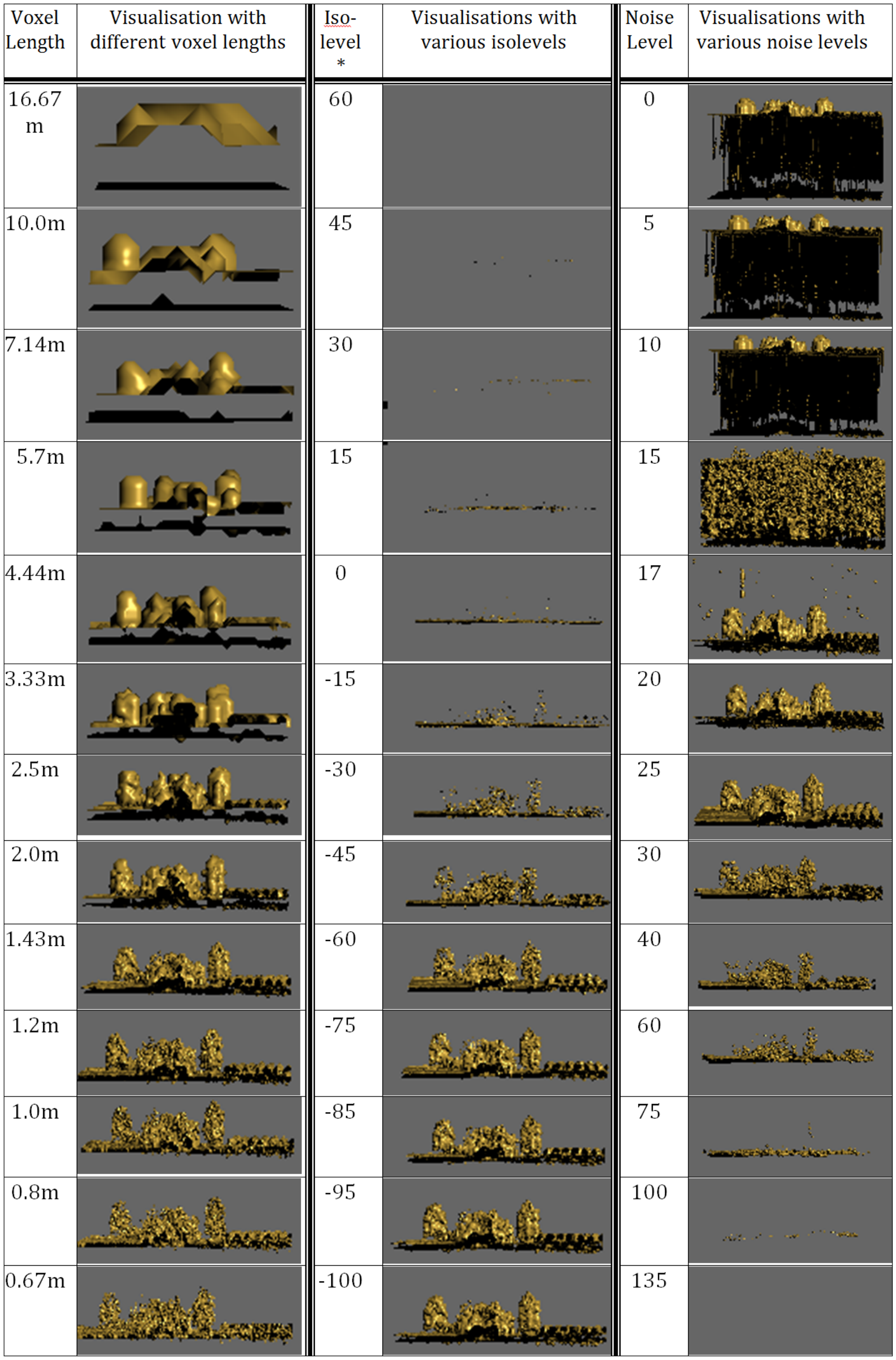


Figure 5-5: How the output polygon mesh is affected by modifying the user-defined parameters (voxel length, isolevel[4] and noise level). Please note that the intensities were scaled to be within the range [-100,100] and that the currently released version of DASOS does not scale the intensities.

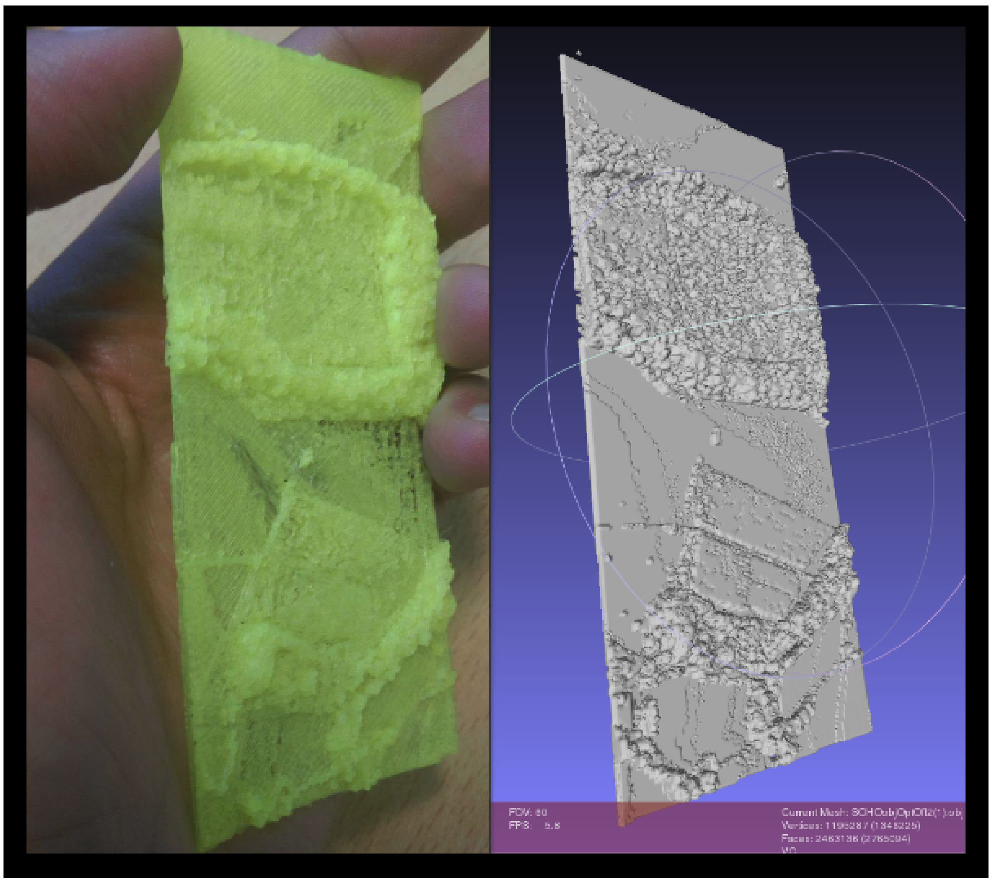


Figure 5-6: 3D printing of New Forest FW LiDAR data

Chapter 6

Optimisation Attempts for the Surface Reconstruction

6.1 Problem and Challenges

While Section 5 explains a simple approach of extracting a polygonal surface from the voxelised FW LiDAR data, this section is mainly focused on objective No. 5 from table (3.1); it tests the performance of six different data structures on the surface reconstruction and it attempts to improve the interpretation of volumetric data by introducing new data structures. The main challenges raised for this task are because the input data is real laser scanning data that contains noise. Some of the challenges that this chapter attempts to tackle are listed below:

1. The LiDAR sensors are vulnerable to clouds and seagulls being misinterpreted and recorded as hit points. Those outliers are much higher than tree canopies but they are within the boundaries of the scanned area. As a result, on average 97.5% of the voxels are empty.

2. The Marching Cube, as described in Section 5.4, is a scan line algorithm, which implies looping through every single voxel, including the empty ones. This is very time consuming and therefore, algorithms that quickly identify and ignore empty areas are essential.

3. While loading an entire volume, the huge amount of empty voxels may lead into exceed memory usage. It is therefore preferable to store the voxels into structure that avoids storing the empty ones (i.e. hierarchically).

4. When extracting a surface from real data, it is very likely to generate non-manifold objects. Non-manifold objects are not homeomorphic to Euclidean 1-space be-

cause they have crossing points. This also occurs at the polygonal meshes generated by DASOS as explained at Chapter 5.

***NEILL: I would make these comments higher level until you have explained more in the chapter.

6.2 Related work

6.2.1 Full-Waveform LiDAR Visualisation

Summarising previous aforemonetioned related work (Section **??**), traditional ways of interpreting the full-waveform LiDAR data suggest echo decomposition for detecting peak points and interpreting the point clouds extracted [28]. Both SPDlib [32] and FullAnalyse [31] visualises either the peak extracted points or the raw waveform samples. On the one hand, SPDlib visulises the samples as points with intensity above a given threshold, while FullAnalyse generates a sphere with radius directly correlated to that intensity of each wave sample. Similarly, Pulsewaves visualises a number of waveforms with different transparency according to their intensity [25]. On the one hand, visualising all the wave samples makes understanding of data difficult due to the high noise. On the other hand, peak point extraction identifies significant features but the FW LiDAR data also contains information about echo widths. These information can be accumulated from multiple shots into a voxel array, building up a 3D discrete density volume [35].

Voxelisation of FW LiDAR data was introduced by [26] who used it to visualise small scanned areas (15mx15m). The waveforms samples were inserted into a 3D Voxelised space and the voxels were visualised using different transparencies according to their intensity. Similarly, as explained at Section **??**, we adopt voxelisation for surface reconstruction and applied it on larger areas. Once the 3D density volume is generated, numerical implicitisation is used to represent the scanned area. Nevertheless, visualising numerical/implicit objects is not straight forward, since they contain no discrete values (Section 5.3). This problem can either be address by ray-tracing [38] or polygonisation [44]. In this thesis, the polygonisation direction is taken and a simple approach is explained in Section 5.4. This chapter introduces new ways of interpreting real voxelised data and tests how well six data structures and algorithms perform on surface reconstruction.

6.2.2 Optimising Volumetric Iso-surface Extraction

Even though volumetric visualisation has only been recently used for FW LiDAR systems, there are many applications in medical visualisation [45] [46] and visual effects [41] [47]. Research work exists on optimising both ray-tracing and surface reconstruction and it can be categorised into three groups: surface-tracking, parallelisation and data structures. Those approaches are discussed below along with their benefits and limitations with respect to voxelised FW LiDAR data.

Surface-tracking was applied at Rodrigues de Araujo and Pires Jorge [48] and Hartmann [49]. Starting from a seed point, the surface is expanded according to the local curvature of the implicit object. This method is considered to be faster and more efficient in comparison to the Marching Cubes algorithm since huge empty spaces are ignored. It further opens up possibilities for finer surface reconstruction at areas with high gradient changes. Nevertheless, surface-tracking algorithms cannot be applied with real laser scanning data because these data are neither manifold nor closed. For example, in a forest scene, a tree canopy may be detached from the ground due to missing information about its trunk. Therefore, by tracking the surface, the algorithm may converged at a single tree instead of the entire forest.

Hansen and Hinker proposed parallelising the polygonisation process of BlobTree trees on Single Instruction, Multiple Data (SIMD) machines [50]. The Instruction is a series of commands to be executed. The longer the series of the commands is, the greater the speed up is. BlobTree trees represent implicit objects as a combination of primitives and operations [51]. While the depth of the tree increases, the length of the parallelised instruction increases as well and therefore good speed up is achieved. Nevertheless the function at the implicit representation of the FW LiDAR data at [35] is executed at constant time, making it harder to achieve speed up using SIMD machines. Further, according to the C++ Coding Standards when optimisation is required is better to seek an algorithmic approach first because it is simpler to maintain and less likely to contain bugs [52].

Hierarchical data structures, like octrees, improves the performance of the isosurface extraction because of the huge amount of empty voxels that can be ignored during polygonisation [53]. The literature in the data structures direction aims to either simplify/improve the output mesh, optimise traversal time of hierarchical data structures or eliminate hierarchy. For example, the extraction of locally finer details either with dual grids [54] or edge-trees [55] reduces the amount of vertices produced. In addition, a net of linked surface nodes improved anti-aliasing and reduces artifacts of 3D Magnetic Resonance Imaging (MRI) [56]. Regarding efficiency of accessing data, franctional cascading slightly improved time complexity of range queries [57]. Sparse Voxel Octrees improved efficiency by having a pointer pointing to children and packing children coherently in memory [47]. Hadwiger et al. used a 3D virtual memory to keep voxels coherent on GPU and avoid traversal [46]. Nevertheless, due to the adjacency of neighbouring voxels, data are saved for empty voxels yielding into much wasted memory. OpenVDB library arranges blocks of grids into a B+ hierarchical data structure for increased cache coherency and lower tree depth [58]. The bricks stuctured used at GigaVoxels is similar in terms of blocks, named bricks, and it's been used for efficient GPU ray-casting [41]. For eliminating tree traversal time, Warren and Salmon introduced hash octrees for N-bosy simulation of particles [59]. Similarly, voxel hashing was proposed for overheading the traversal time of hierarchical structures and real time surface reconstruction using depth cameras online [60]. Most of those data structure optimisations are based on GPU processing, but they are still very relevant.

6.3 Overview

This thesis compares six approaches for handling and polygonising voxelised full-waveform LiDAR data. The first three approaches use data structures from the literature and the scan line Marching Cubes algorithm. An explanation of their functionalities is given at Table 6.1. The last three approaches are more complicated because they take into consideration the chunks of empty voxels and ignore them during surface reconstruction. A brief summary of them is given in Table 6.2 and an in-depth explanation is given in Sections 6.4 6.5 6.6 . Please note that the "1D Array" is the original implementation, while each one of the other five approaches tackles at least one of the aforementioned challenges (Section 6.1).

1D Array	Voxel Hashing	Octree
Influenced by [46], all the data are saved into an 1D array to guarantee coherent memory, even though much memory is wasted in regards of empty voxels.	The intensities of the voxels are saved into a simple hash table with key value relevant to their position into the volume. Similary to [60], this approach overheads traversing time of hierarchical structures and on top of that it reduces memory allocation because empty voxels are not stored.	This is a hierarchical octree with traversal time to be essential. Please note that this is a scan line test and therefore it does not take into consideration empty chunks of memory.

Table 6.1: Brief Description of the Three Scan-Line Tests

Integral Volumes	Octree Max and Min	Integral Tree
This data structure is an extension of 'Integral Images' to 3D. It was firstly presented at the CGVC conference as part of this thesis. Using Integral Volumes, the sum of any cuboid area is calculated in constant time. By repeatedly dividing the space into cuboids, big empty spaces are quickly identified and ignored during the surface reconstruction. (Section 6.4)	In this approach, the values are saved into an octree, but the surface reconstruction is build along the tree. This is slightly different than a traditional octree, because at each branch node its max and min values are saved. This way, areas that are completely full or empty are identified during traversal before reaching the leaves of the trees. (Section 6.5)	It is a combination of octree and integral volumes; the sum of a given branch is given at constant time. That was an attempt to combine the idea of 'Integral Images' and octrees. Nevertheless, traversal time and backtracking for finding neighbouring voxels still exists. (Section 6.6)

Table 6.2: Description of the Three Optimisation Attempts

6.4 Integral Volumes

The 'Integral Volumes' optimisation is based on the idea of Integral Images, which is an image representation where each pixel value is replaced by the sum of all the pixels that belong to the rectangle defined by the lower left corner of the image and the pixel of interest. An integral image is constructed in linear time and the sum of every rectangular area is calculated in constant time, as shown in figure 6-1 [61]

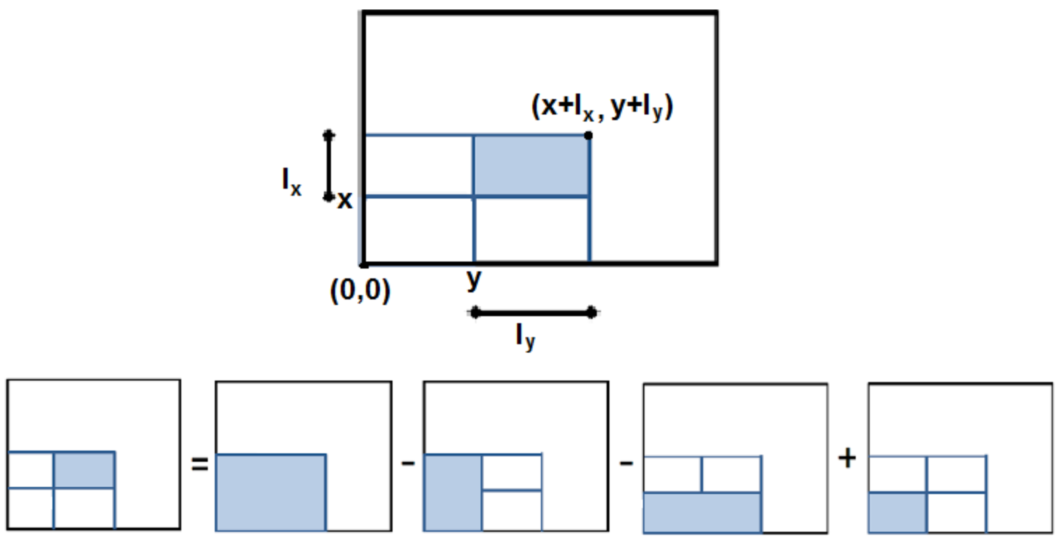


Figure 6-1: Once the Integral Image is constructed, the sum of any rectangular area is calculated in constant time.

In this paper, we extend 'Integral Images' to 'Integral Volumes' and use them to quickly identify and ignore big chunks of empty voxels during polygonisation. The following section explains the mathematics behind 'Integral Volumes', while sections 6.4.2 and 6.4.3 give an in depth description about the algorithms invented.

6.4.1 Extending Integral Images to Integral Volumes

As shown in Figure 6-1,the area of interest is defined by the pixels (x, y) and $(x+l_x, y+l_y)$ and the sum S is given by:

$$\begin{aligned} S =& T(x+l_x, y+l_y) - T(x+l_x, y-1) - \\ & T(x-1, y+l_y) + T(x-1, y-1) \end{aligned} \tag{6.1}$$

where S is the sum of rectangular area of interest, $T(x, y)$ is the value of the integral image at (x, y) and l_x, l_y define the length of the rectangle in the x and y axis respectively.

Extending integral images to 3D, the value of the voxel (x, y, z) in a 3D integral volume becomes equal to the sum of all the values that belong to the box defined by the (x, y, z) and $(0, 0, 0)$ included. Therefore the sum (S) of the box defined by (x, y, z) and $(x + l_x, y + l_y, z + l_z)$ included is given by:

$$
\begin{aligned}
S =& T(x - l_x, y + l_y, z + l_z) - T(x - 1, y + l_y, z + l_z) - \\
& T(x + l_x, y - 1, z + l_z) - T(x + l_x, y + l_y, z - 1) + \\
& T(x - 1, y - 1, z + l_z) + T(x - 1, y + l_y, z - 1) + \\
& T(x + l_x, y - 1, z - 1) - T(x - 1, y - 1, z - 1)
\end{aligned}
\tag{6.2}
$$

where $T(x, y, z)$ is the value of the voxel (x, y, z) in the 3D integral volume. S is the sum of voxels inside the box, $T(x, y, z)$ is the value of the voxel (x, y, z) in the 3D integral volume. and l_x, l_y, l_z define the length of the box in the x, y and z axis respectively.

6.4.2 Optimisation Algorithm

As mentioned before, using 'Integral volumes' empty areas are quickly identified and ignored during polygonisation. An iterative algorithm is introduced here. This algorithm continuously splits the volume and checks whether the sub-volumes and its neighbouring voxels are empty using the 'Integral Volumes'. Please note that all the values below the threshold boundary of the object must be zero and all the non-empty voxels must contain a positive value.

Algorithm 1 Integral Volumes Optimisation Algorithm

```
1: Push the entire Volume as a cuboid inside a Stack
2: while stack is not empty do
3:     Cuboid-A ← next cuboid from the Stack
4:     if Cuboid-A and neighbours are empty then
5:         discard Cuboid-A
6:     else if Cuboid-A consists of only one cube then
7:         polygonise Cuboid-A
8:     else
9:         divide Cuboid-A
10:        push the two new Cuboids into stack
```

Here it is worth highlighting that, on line 3 of the algorithm it is checked if the neighbouring cubes of a cuboid are empty, because the voxels of the 3D density volume and the cubes in marching cubes algorithm are aligned with an offset (Figure 5-1a). If volumes with non-empty neighbouring voxels are ignored, then holes appear on the output polygon mesh.

*** NEILL could draw a red box on image to highlight the difference and menetion it in the caption as well

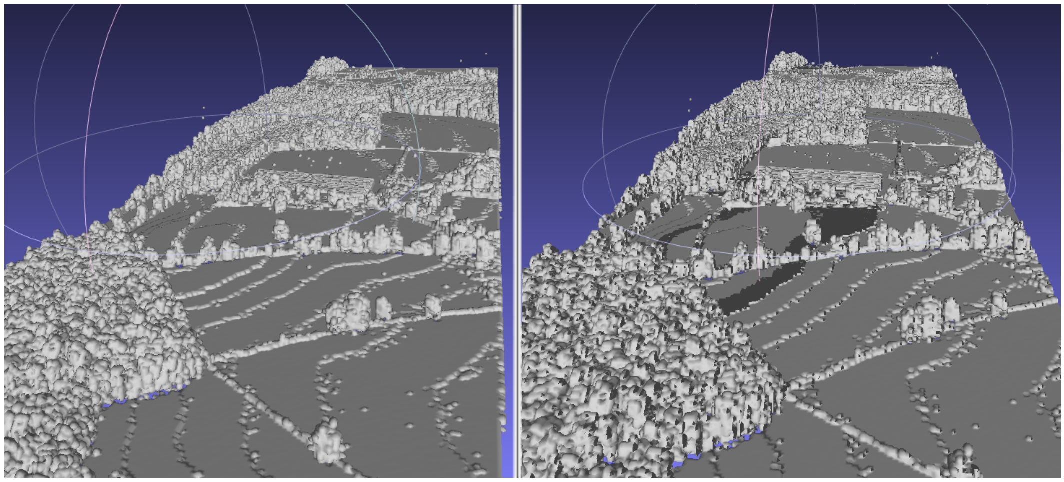


Figure 6-2: Comparison between including and ignoring neighborouing voxels; holes appears when ignored.

6.4.3 Coding Details for Faster Implementation

Implementation details contributes to the efficiency and speed up of the algorithm. Significant improvements are achieved by reducing recursions, big memory allocations and if statements, since memory jumps are time expensive. As shown in algorithm 1, a while loop is used to avoid recursion. In this section it's given an explanation on how the stack controls memory consumption and how bitwise operations reduces if-statement usage.

Regarding memory consumption, a stack was chosen over a queue, to decrease the amount of cubes saved into the data structure simultaneously. A queue is a first in first out data structure, while a stack accesses data in a last in first out order. In every iteration, it is ideal to interpret the smallest saved cube, such that the possibility of being polygonised is higher and the possibility of storing another cube is less. A queue guarantees cubes with approximately the same size, since the big cubes will be added first and sequentially being divided first. In contrast, a stack guarantees the smallest possible number of cubes saved. The larger cubes are stored in the bottom of the stack while the smaller ones are interpreted first because they are always the last one divided and inserted into the stack. For that reason, a stack guarantees the lowest memory usage.

Furthermore, in algorithm 1 an issue exists: how to quickly identify the side to be divided next? Ideally, the usage of if-statements should be low because they contains many time expensive memory jumps. For that reason, bitwise operations were embedded into the program to reduce their usage. A cube is defined with its position, its size, the next side to be divided s and its divisible sides D. The parameter s takes the values 1, 2, 3 for the x, y, z sides respectively. The parameter D is an integer consisting of the sum of three numbers (1 or 0) + (2 or 0) + (4 or 0) indicating whether the sides x, y, z are divisible or not (table 6.3). The parameter D takes the value between $[0, 7]$ and covering all the possible cases of divisible sides as shown in tables 6.4 and 6.5. For example if x and z are the divisible sides, then $D = 1 + 0 + 4 = 5$. By the end, the bitwise operations and the faster implementations of the Integral Volumes optimisations is shown at algorithm 2.

	Decimal Numbers		Binary Numbers	
Side	Divisible	Not Divisible	Divisible	Not Divisible
X	1	0	0001	0000
Y	2	0	0010	0000
Z	4	0	0100	0000

Table 6.3: Values of divisible sides

X	1	-	1	-	1	-	1	-
Y	2	2	-	-	2	2	-	-
Z	4	4	4	4	-	-	-	-
D	7	6	5	4	3	2	1	0

Table 6.4: How to calculate the value of D, which represents the divisible sides of a cuboid

X	0001	-	0001	-	0001	-	0001	-
Y	0010	0010	-	-	0010	0010	-	-
Z	0100	0100	0100	0100	-	-	-	-
D	0111	0110	0101	0100	0011	0010	0001	0000

Table 6.5: How to calculate the value of divisible sides (D) in binary representation

Algorithm 2 Integral Volumes Optimisation Algorithm

Push the entire Volume as a cuboid inside a Stack
while stack is not empty **do**
  Cuboid-A $\leftarrow$ next cuboid from the Stack
  if Cuboid-A and neighbours are empty **then**
    discard Cuboid-A
  else if D is equal to 0 **then**
    polygonise Cuboid-A
  else if (D bitwise add 2^s) shift right $(s-1)$ **then**
    divide side s of Cuboid-A
    if the new length of side s is equal to 1 **then**
      $D \leftarrow D$ bitwise add $(7-2^s)$
    $s \leftarrow (s+1) \bmod 3$
    push both new Cuboids into stack
  else
    $s \leftarrow (s+1) \bmod 3$
    push Cuboid-A back into the stack

6.5 Octree Max and Min

'Integral Volumes' quickly identify and ignore empty spaces during polygonisation (tackles the 1st, 2nd and 4th problem of the original algorithm – Section 6.1), but it allocates memory for the entire volume (the 3rd problem). For that reason, the 'Octree Max and Min' data structure has been implemented.

The 'Octree Max and Min' data structure avoids storing empty voxels and it also identifies empty areas during polygonisation. The polygonisation is built on the traversal of the octree, as explained in Algorithm 3. Similarly to 'Integral Volumes', a stack is used to avoid recursion and reduce memory jumps. When the 'Integral volumes' are used, it is checked whether the neighbours of a cuboid and itself are empty or not. If they are both empty then the cuboid is ignored. The neighbouring voxels have to be checked in order to avoid generating holes on the polygonal mesh. Similalry, it is essential to check neighbouring voxels when a branch of the 'Octree Max and Min' data structure is ignored. Nevertheless, because the branches of the octree are always a cube, it is not trivial to check whether they are empty or not. For that reason, if a branch is empty then we loop through its edges and polygonise them according to look up table of the the Marching Cubes algorithm.

Embedding the polygonisation of volumetric data into an octree has been done before [53]. Nevertheless, the 'Octree Max and Mean' data structure differs in two ways:

Algorithm 3 Embedding the Marching Cubes Algorithm into an octree structure

```
Push the Root as a Node into a Stack
while stack is not empty do
    Node-N ← next Node from the Stack
    if Node-N is a Leaf then
        polygonise Leaf
    else if Node-N has no children OR max value of Node-N < isolevel
OR min value of Node-N > isolevel then
        Polygonise edges of cubic with root node-N
    else
        push the children of Node-N into the Stack
```

- The max and min values of each branch are stored into the corresponding node to speed up polygonisation. This enables checking whether the leaves of a branch lie either only inside or only outside the implicit object[1]. If they do, then no iso-surface is crossing that branch and it can be discarded (after polygonising its edges).
- A new algorithm is proposed and implemented for finding neighbouring voxels. This algorithm reduces comparisons and jumps in memory. An in-depth explanation of this algorithm is given at Section 6.5.1.

6.5.1 Finding Neighbours

Every time a voxel/leaf is polygonised, seven of its neighbours are checked to decide whether a surface is passing through that area or not. At hierarchical data structures, the nearest common ancestor is tracked upwards and the branch, with root the common ancestor, is traversed to reach the neighbour. The article [62] uses recursion that terminates once a common ancestor between a leaf and its neighbour is identified. According to Scharack [63], finding neighbours in linear octree[2] is done in constant time. Nevertheless, linear octrees are full octrees. Therefore, if used in our case all the empty voxels would have to be stored as well. Lohner suggested vectorising the space during post-processing for finding the shortest distance between un-constructed points [64]. However, the 3D voxelised FW LiDAR is a regular grid and during polygonisation the shorter distance to travel is one voxel. For that reason, simpler approaches with less initialisation time, like [63], could perform equally well. Castro et al. [65] assume that with hierarchical octrees it is not possible to start searching neighbours from leaves and

[1] Explanation about implicit/algebraic objects is given at Section 5.3

[2] Linear octrees are octrees whose leaf nodes are stored into a linear array.

suggest using hashed octrees to do that. In contrast, it is possible to start from the leaves and find the common ancestor using parentship as described at [62].

To avoid recursion and reduce comparison, this thesis introduces a new way of finding the common ancestor using logarithms of 2. The Algorithm 4 explains the proposed method. As shown in Figure 6-3, there are occasions where it is cheaper to start searching a neighbour from the root instead of the leaf. For example Node-F is the (+1) neighbour of Node-E. If we start looking for it from the leaves then we need to travel through 6 nodes, but if we start from the root we only need to travel 5 nodes. Logarithms helps us decide which route to take, while reducing comparisons since it is not required to check whether branches has common faces while travelling upwards [62].

Algorithm 4 Finding the number of steps required to go upward in order to find the common ancestor of a Leaf(x) of interest and its (+1) neighbour

$c \leftarrow \text{ceil}(log_2 x)$
$c_1 \leftarrow \text{ceil}(log_2(x+1))$
while $c = c_1$ **do**
    $x = x - 2^{(c-1)}$
    $c \leftarrow \text{ceil}(log_2 x)$
    $c_1 \leftarrow \text{ceil}(log_2(x+1))$
if $D_{max}/2 < c_1$ **then**
    Start from Root to find Neighbour Branch $+1$
else
    Backtrack c_1 parents to find the common ancestor
    Find neighbour

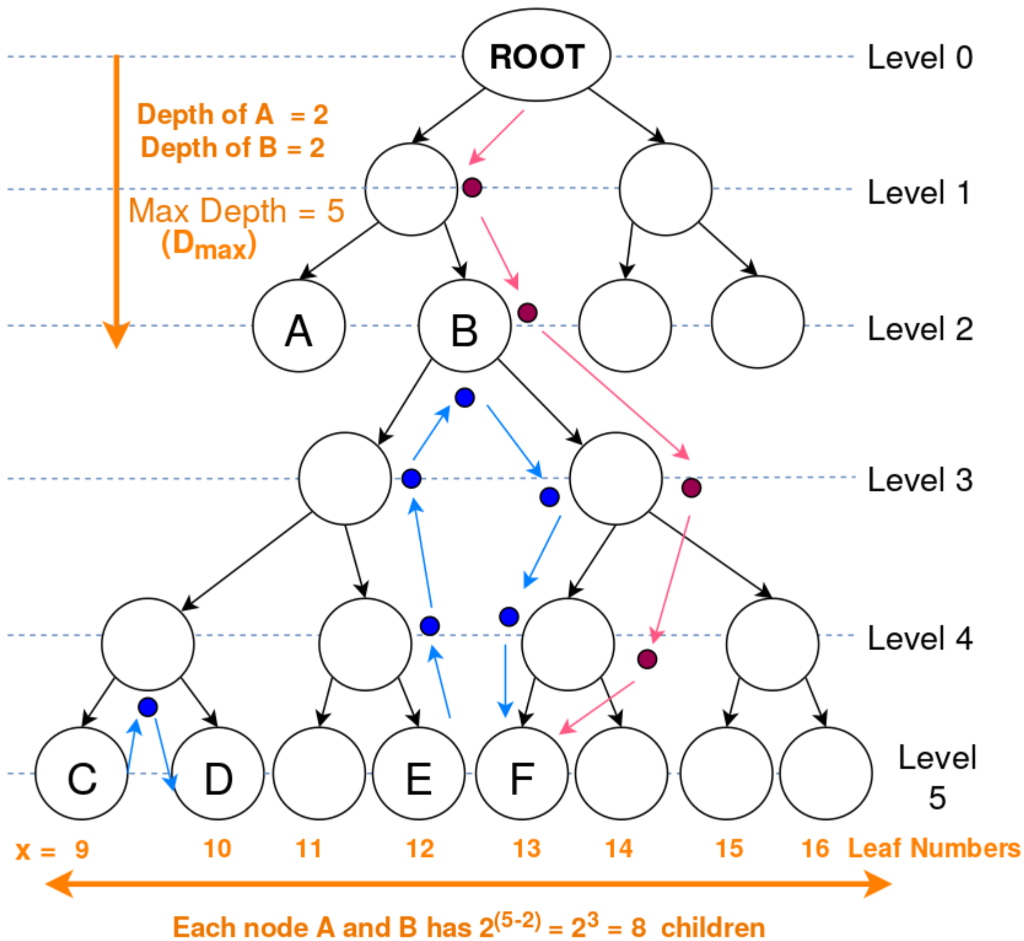


Figure 6-3: This diagram depicts the parameters used for finding neighbouring voxels.

6.6 Integral Tree

6.6.1 Main Idea

The 'Integral Tree' is a new term that describes the attempt to preserve some properties of the 'Integral Images' while using a non-full tree structure. Every 'Integral Tree' consists of two elements: an integral 1D-array and a tree. All the values of every non-empty and non-connecting node are saved into an 1D-array, in a way such that the condition of the 'Integral Tree' is (*** NELL needs to be clarified) fulfilled: all the values of every branch B are adjacent inside the 1D-array. Afterwards the array is converted to integral; the sum of every n continuous values is calculated in constant time. Additionally, the root node of each branch B contains two parameters $(*p, k)$. The number k is the number of nodes, which contain values, of the branch B (e.g. for an octree, it is all its leaf nodes) and the pointer $*p$ points to the first one in the 1D-array (Figure 6-4).

*** NELL:: the *p is a pointer)

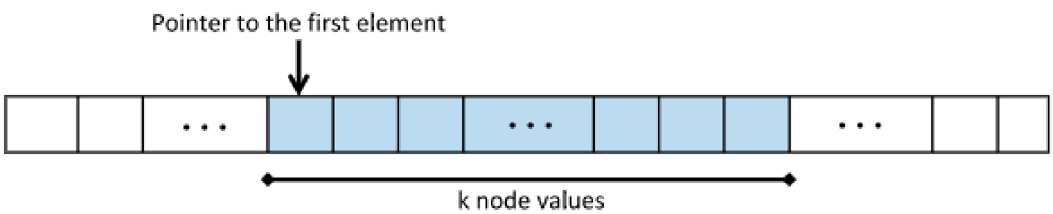


Figure 6-4: Ordering of tree elements

The aforementioned rules can be applied to any tree structures including binary trees, quadtrees and octrees. To better perceive how this data structure works, let's assume that there is a number of 2D spatially distributed values. Figure 6-5 depicts how they can be saved into an 'Integral Quad Tree' in order to fulfil the adjacency condition of the 'Integral Tree'. Also, Section 6.6.2 gives an example of an 'Integral Binary Tree'.

6.6.2 Integral Binary Tree Example

An example of applying the idea of 'Integral Tree' into a binary tree is given for clarification (Figure 6-6). Firstly, the values of the binary tree are sorted into the 1D-array A as {15, 12, 10, 13, 14, 17, 16, 18, 19} in order to fulfil the adjacency condition. Secondly, the array A is modified as {15, 27, 37, 50, 64, 81, 97, 115, 134} in order to

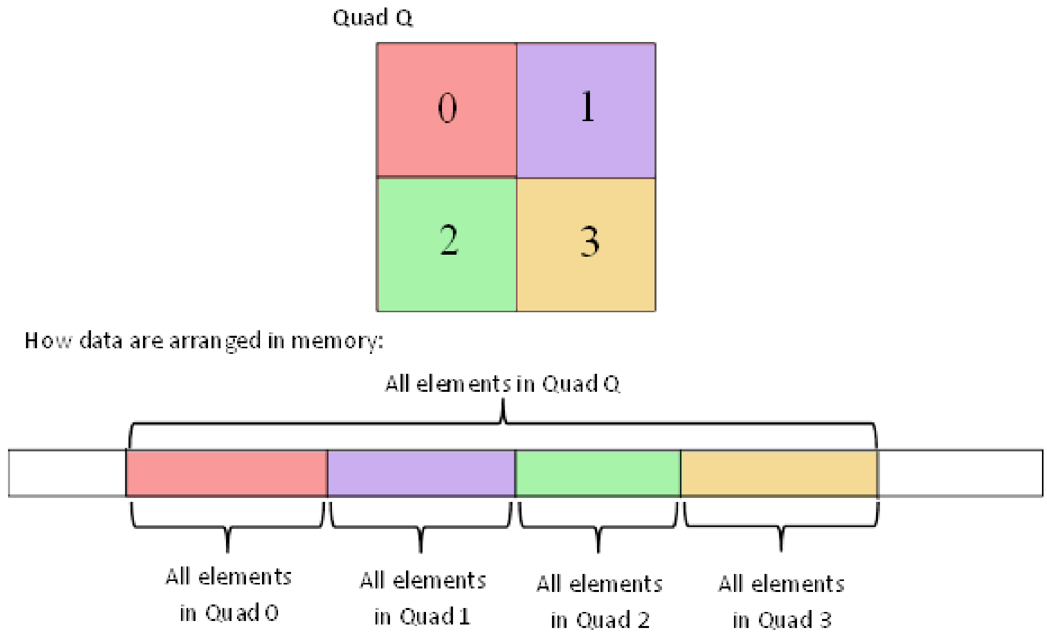


Figure 6-5: Illustration of how to save the values of an 'Integral Quad Tree' into the 1D-array, in order to preserve the condition of 'Integral Trees'

*p	0	1	2	3	4	5	6	7	8
1-D Array (1st step)	15	12	10	13	14	17	16	18	19
1-D Array (2nd step)	15	27	37	50	64	81	97	115	134

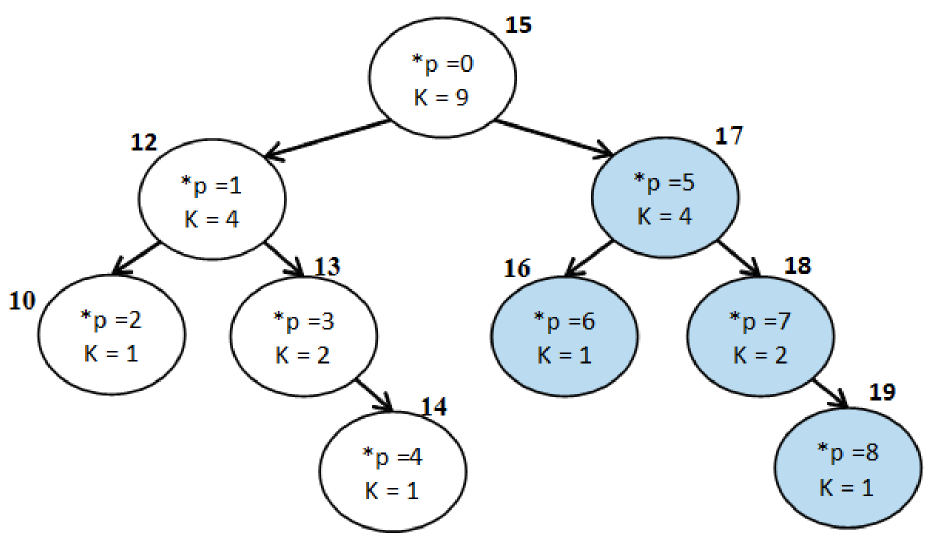


Figure 6-6: Example of 'Integral Binary Tree'

become integral using the following equation:

$$A[i] = A[i] + A[i-1] \tag{6.3}$$

Then the sum S of a branch, with $(*p, k)$ parameters, is calculated at constant time as follow:

$$S = A[*p + k - 1] - A[*p - 1] \tag{6.4}$$

For instance the sum of the blue branch on Figure 6-6 is $A[5+4-1] - A[5-1] = A[8] - A[4] = 134 - 64 = 70$, which is correct since $17 + 16 + 18 + 19 = 70$.

6.6.3 Integral Octree for Surface Reconstruction

For an 'Integral Octree', all the values saved into the integral 1D-array are the values of the leaf nodes since the rest are connecting nodes. For the surface reconstruction, an 'Integral Octree' is implemented and the same polygonisation algorithm as 'Octree Max and Min' are used (Algorithm 3 and Algorithm 4). The only difference is the comparison at Line 6 of Algorithm 3; instead of checking the max and min values, the sum of the branch is checked instead. If the sum is smaller than the iso-surface value then no surface is crossing that area and the branch is discarded.

6.7 Data Structures Summary

To briefly sum up, the following six data structures has been implemented their performance has been tested for reconstructing polygonal meshes from voxelised FW LiDAR data:

1. **1D-Array**: Simple array that keeps data coherent in memory for quick access.
2. **Voxel Hashing**: A hashed table is used for storing the intensity values of the voxels [60].
3. **Octree**: Simple hierarchical structure with a scan-line implementation.
4. **Integral Volumes**: Extension of 'Integral Images' that allows finding the sum of any cuboid area in constant time. It is a new algorithm and it is used for quickly identifying and ignoring empty areas during polygonisation.
5. **Octree Max/Min**: The polygonisation is embedded into an hierarchical data structure [53]. The max and min values of each branch are stored to identify and

ignore branches that either only contain low level noise or are completely inside the implicit object. Logarithms are further introduced for faster neighbouring finding.

6. **Integral Octree**: An attempt to preserve properties from both 'Octree Max/Min' and 'Integral Volumes'.

Each one of the aforementioned data structure has different properties and attempts to address at least one of the problems mentioned in Section 6.1. The first three implementations are scaline algorithms, which means that polygonisation is linear and all the voxels, including the empty ones, are checked for generating triangles primitives. Some data structures are taken from the literature to test how well they perform on this specific datasets while others are new and presented into this thesis. Table 6.6 summarises their properties and the problems each data structure attempts to resolve.

***Neill: Include citations in table for existing scanline algorithms from literature

	Scan-line algorithm: loops through all voxels (1)	Identifies and ignores empty areas during poly-gonisation (2)	Avoids storing empty voxels in memory (3)	Works on non-manifold objects (4)	Requires Cubic Boundaries of the voxelised data	New data structure, introduced for this thesis
1D-Array	✓	-	-	✓	-	-
Voxel Hashing	✓	-	-	✓	-	-
Octree	✓	-	✓	✓	✓	-
Integral Volumes	-	✓	-	✓	-	✓
Octree Max/Min	-	✓	✓	✓	✓	✓[3]
Integral Octree	-	✓	✓	✓	✓	✓

Table 6.6: Summarising the addressed challenges and the properties of all the data structures implemented.The numbers of the first four columns correspond to the challenges described in Section 6.1

[3]Integrating polygonisation into an octree has been done before, but there only a few modifications to a normal octree; the max and min values stored into the branch and the introduction of logarithms for finding neighouring voxels.

6.8 Results and Testing

The implemented algorithms are beneficial in different aspects: speeding up execution or decreasing memory usage. The performance has been tested within two groups of test cases:

- The results of the first group are given in Table 6.7 and visualised in the Charts depicted in Figures 6-7, 6-8, 6-9 and 6-10
- The results of the second group are given in Table 6.8. The related charts are inside Table 6.9.

This section clarifies the various parameters of testing, while the following Section 6.9 discussed the results and explains the behaviour of the algorithms in respect to the results.

The two test cases has only one difference, while the rest of the parameters are the same. The difference is that the first one uses one flightline (the LDR-FW-FW10_01-201009821.LAS from New Forest) and focuses on its performance using a finer resolution range. In contrast, the second uses three filghtlines from the NERC-ARF datasets. The 1st flightline is from the New Forest (LDR-FW-FW10_01-201009822.LAS), the 2nd flightline is from the Dennys Wood (LDR-FW10_01-201018713.LAS) and the 3rd one from Eaves Wood (LDR-FW-GB12_04-2014-083-13.LAS). The 2nd group checks whether there are significant performance differences when the algorithms are applied on different fligtlines.

Except from the flightlines used, the rest of the parameters are the same in both test cases. In order to understand the size of the data, the voxel length, the number of voxels in the x,y,z axes and the percentage of empty voxels are stated. The smaller the voxel length is, the more voxels exist because the boundaries of the voxels in meters are constant and when the voxel length decreases the resolution of the volume increases. Additionally, for every resolution the execution time and maximum memory consumption are measured. Execution time is further divided into data structure construction (including reading the LAS file) and polygonisation.

*** Neill::: Consider adding log(execution time) in diagrams and sum of voxels per test in the results

			1D-Array				**Voxels Hashing**				**Octree**			
Specifications			Time (s)			Memory	Time (s)			Memory	Time (s)			Memory
Length (m)	No. of Voxels	Empty	Con	Pol	Total	MByte	Con	Pol	Total	MByte	Con	Pol	Total	MByte
20	29x115x23	93.20%	12.04	0.16	12.21	10.17	12.84	0.19	13.02	9.78	14.58	0.18	14.76	11.07
15	39x157x30	94.32%	12.06	0.32	12.38	12.50	12.96	0.37	13.33	11.44	14.91	0.35	15.26	12.00
10	58x235x45	95.08%	12.07	0.8	12.87	20.09	12.95	0.96	13.92	16.19	14.92	0.91	15.82	16.69
5	116x476x89	96.38%	12.08	4.85	16.92	88.35	13.01	6.95	19.96	47.66	15.26	5.55	20.81	50.50
4	145x597x111	96.81%	12.24	9.21	21.45	158.94	13.08	12.83	25.91	76.70	15.58	10.61	26.19	80.31
3	194x800x148	97.42%	12.19	21.9	34.09	362.23	13.23	29.94	43.16	153.27	15.67	24.14	39.81	178.27
2	290x1199x222	98.21%	12.45	67.65	80.10	1153.13	13.69	95.85	109.54	389.34	16.16	75.29	91.45	417.98
1.5	387x1602x295	98.70%	12.83	151.48	164.31	2666.67	13.96	216.35	230.31	788.00	16.26	166.23	182.49	839.35
1	80x2405x443	99.24%	14.62	443.5	458.1	8556.78	15.43	672.07	687.5	1912.57	16.91	491.88	508.79	2056.805

			Integral Volumes				**Octree Max/Min**				**Integral Octree**			
Specifications			Time (s)			Memory	Time (s)			Memory	Time (s)			Memory
Length (m)	No. of Voxels	Empty	Con	Pol	Total	MByte	Con	Pol	Total	MByte	Con	Pol	Total	MByte
20	29x115x23	93.20%	12.9	0.15	13.05	10.38	14.65	0.21	14.86	18.32	15.67	0.23	15.9	18.27
15	39x157x30	94.32%	12.11	0.28	12.39	12.80	16.01	0.34	16.35	19.80	15.76	0.37	16.13	20.16
10	58x235x45	95.08%	12.17	0.68	12.85	20.43	16.12	0.89	17.01	25.68	16.32	0.92	17.24	25.93
5	116x476x89	96.38%	13.62	3.56	16.02	88.84	16.31	4.99	21.3	67.50	16.98	5.03	22.01	68.94
4	145x597x111	96.81%	13.32	6.48	19.81	159.08	16.62	9.45	26.07	110.24	17.45	9.67	27.12	117.25
3	194x800x148	97.42%	15.15	14.37	29.52	363.95	16.74	26.16	42.9	218.92	17.51	26.35	43.86	231.67
2	290x1199x222	98.21%	23.11	40.80	63.91	1154.02	17.21	63.02	80.23	595.01	18.14	64.08	82.22	720.01
1.5	387x1602x295	98.70%	39.64	86.54	126.18	2667.67	18.37	131.21	149.58	898.8	21.22	133.46	154.68	1068.43
1	80x2405x443	99.24%	111.38	210.94	322.32	8559.66	19.91	348.97	368.88	2087.71	25.83	352.31	378.14	2223.14

Table 6.7: Results: Execution time and memory consumption, Con=Construction, Pol= Polygonisation, MByte=Max Memory

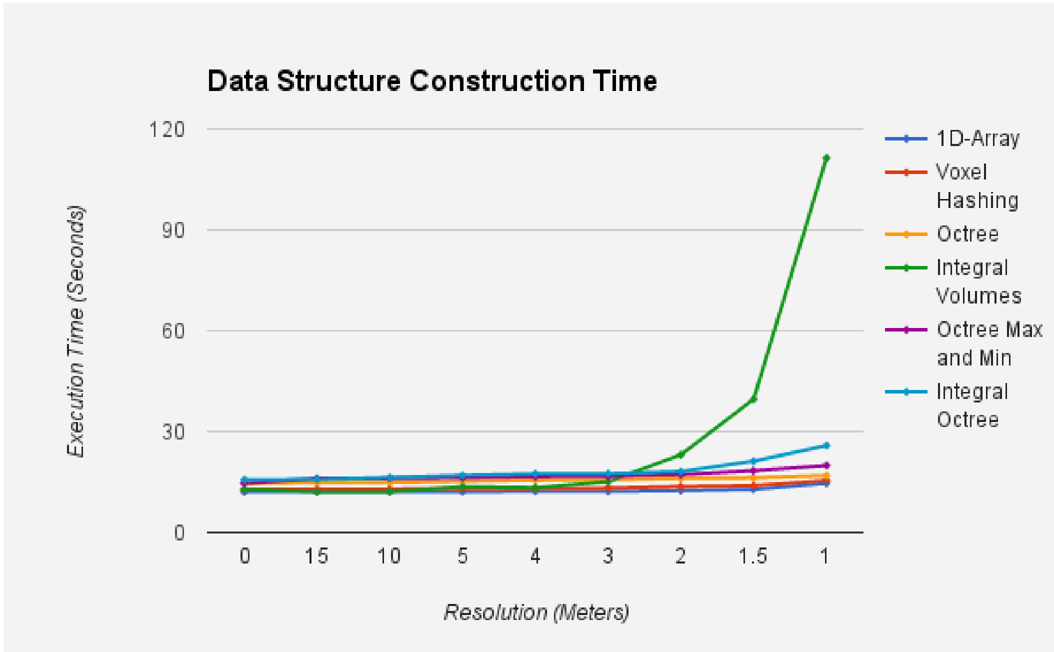


Figure 6-7: Time required to build each data structure by voxelising the FW LiDAR samples and inserting them inside the 3D volume (Table 6.7).

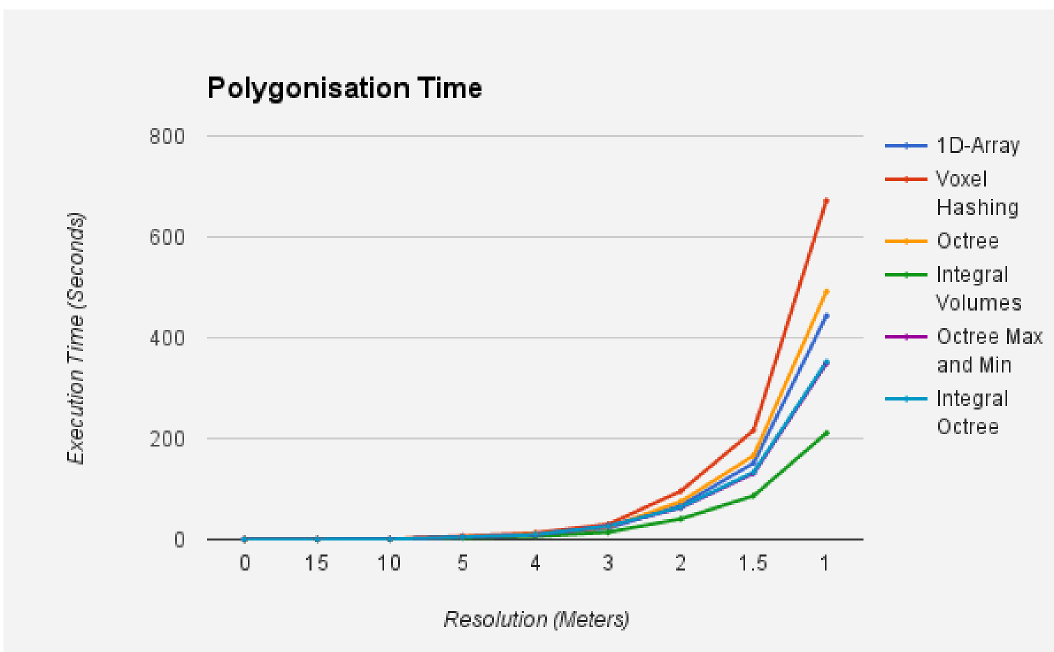


Figure 6-8: Time required to reconstruct the surface from the voxelised FW LiDAR data, after the data are voxelised (Table 6.7).

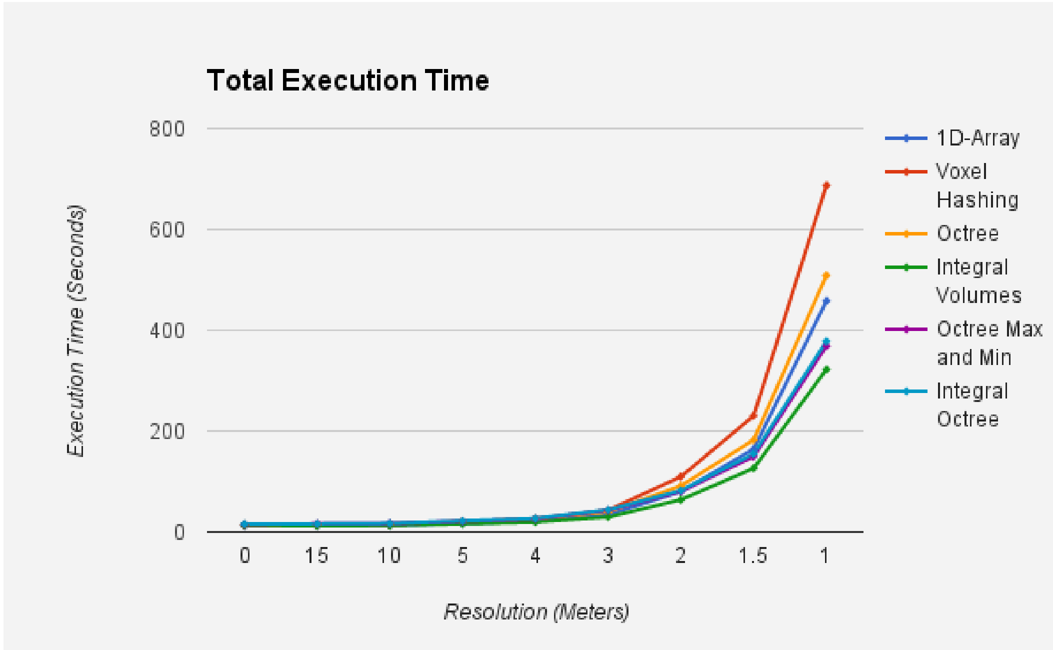


Figure 6-9: The sum of the time required to construct a data structure and the time required to generate a polygonal mesh (Table 6.7). The fastest one is the 'Integral Volumes'.

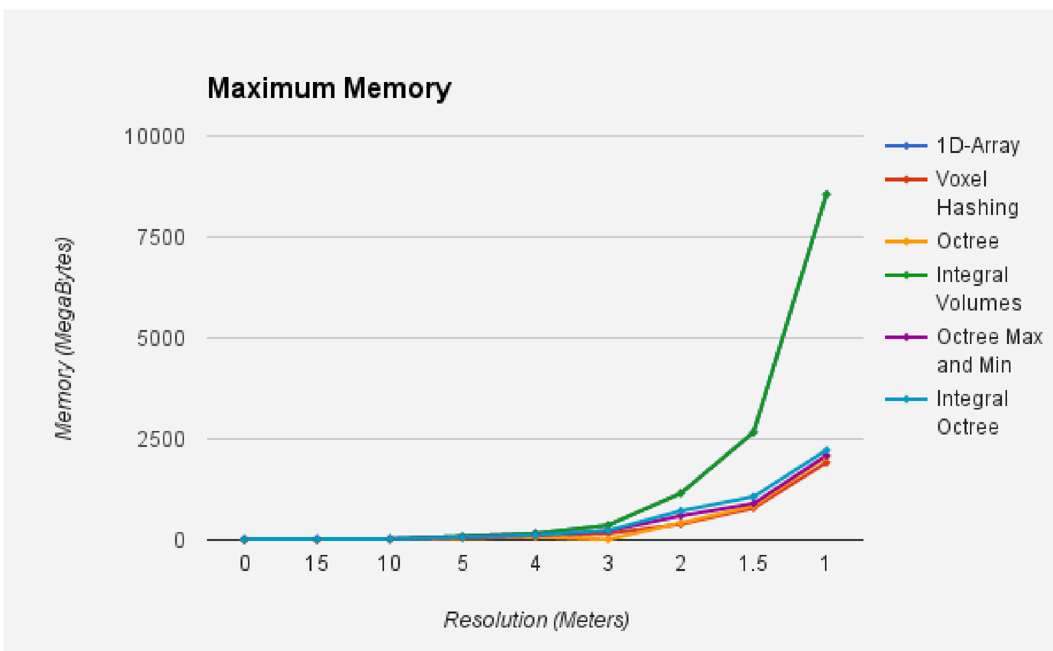
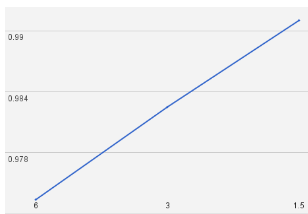
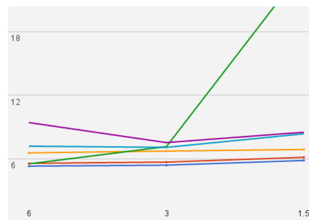
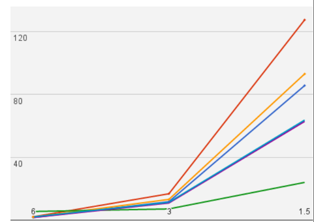
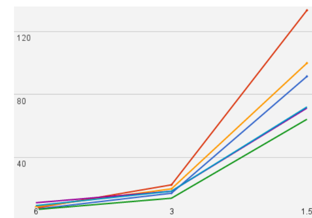
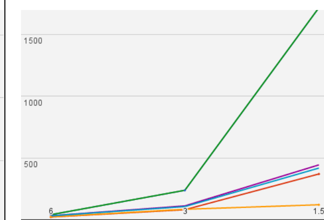
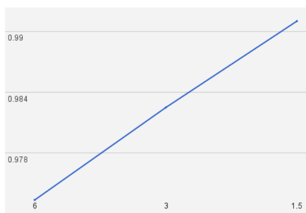
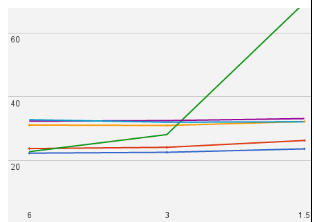
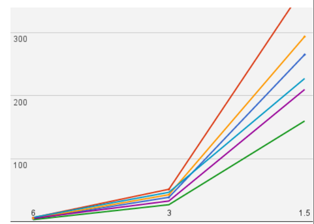
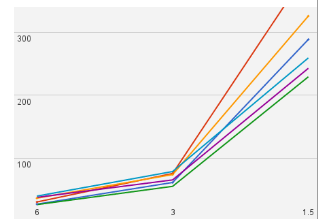
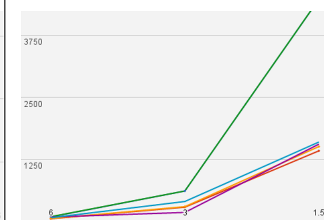
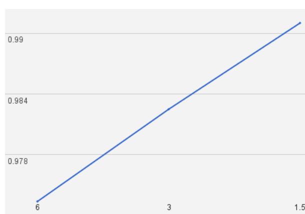
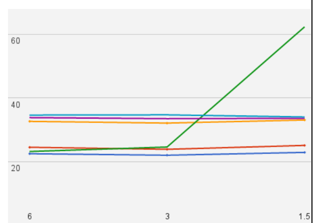
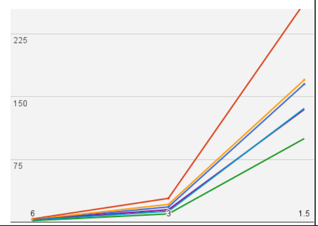
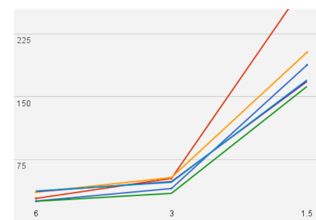
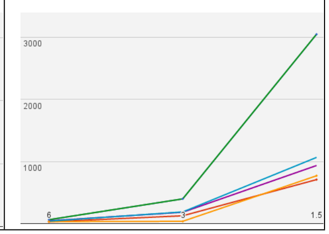


Figure 6-10: Maximum memory consumption at run time. '1D-Array' and 'Integral Volumes' consume the highest memory, which is approximately the same (Table 6.7).

			1D-Array				Voxels Hashing				Octree			
Specifications			Time (s)			Memory	Time (s)			Memory	Time (s)			Memory
Length (m)	No. of Voxels	Empty	Con	Pol	Total	MByte	Con	Pol	Total	MByte	Con	Pol	Total	MByte
6	96x250x76	97.34%	5.29	1.70	6.99	40.01	5.55	2.13	7.68	21.13	6.54	1.91	8.45	22.55
3	191x561x149	98.25%	5.38	11.73	17.11	237.39	5.67	16.76	22.43	80.45	6.71	13.18	19.89	83.95
1.5	381x1122x296	99.10%	5.82	85.51	91.33	1713.74	6.12	127.23	133.35	369.61	6.86	92.91	99.77	120.57
6	100x760x64	94.43%	22.21	4.38	26.59	84.55	23.65	6.40	30.05	48.10	31.04	5.07	36.11	52.23
3	199x1525x124	96.74%	22.48	38.57	61.05	608.29	24.05	51.31	75.36	281.26	30.9	42.70	73.6	292.18
1.5	398x3063x248	98.50%	69.42	159.5	228.92	4478.66	33.06	209.41	242.47	1553.92	32.05	226.85	258.9	1596.43
6	382x90x108	96.60%	22.43	2.75	25.18	62.50	24.45	3.87	28.32	29.67	32.58	3.19	35.77	32.16
3	763x178x213	97.52%	21.95	18.20	40.15	397.73	23.81	28.42	52.23	126.06	32.05	21.20	53.25	37.37
1.5	1526x355x424	98.38%	22.84	164.73	187.57	3044.09	25.03	261.64	286.67	707.75	33.00	169.8	202.8	769.43
			Integral Volumes				**Octree Max/Min**				**Integral Octree**			
Specifications			Time (s)			Memory	Time (s)			Memory	Time (s)			Memory
Length (m)	No. of Voxels	Empty	Con	Pol	Total	MByte	Con	Pol	Total	MByte	Con	Pol	Total	MByte
6	96x250x76	97.34%	5.50	1.23	6.73	40.05	9.40	1.67	11.07	31.50	7.17	2.07	9.24	30.88
3	191x561x149	98.25%	7.13	6.80	13.93	237.75	7.51	10.89	18.40	111.52	7.06	11.33	18.39	105.68
1.5	381x1122x296	99.10%	23.98	40.13	64.11	1714.71	8.49	62.73	71.22	443.09	8.34	63.60	71.94	417.36
6	100x760x64	94.43%	22.69	3.19	25.88	89.9	32.26	5.17	37.43	82.86	32.70	6.70	39.40	68.93
3	199x1525x124	96.74%	28.04	26.86	54.90	608.79	32.43	32.70	65.13	176.47	31.94	46.50	78.44	396.45
1.5	398x3063x248	98.50%	69.42	159.50	228.92	4478.66	33.06	209.41	242.47	653.92	32.05	226.85	258.9	1546.43
6	382x90x108	96.60%	23.12	1.80	24.92	63.02	33.76	2.77	36.53	45.84	34.56	2.62	37.18	40.33
3	763x178x213	97.52%	24.53	9.87	34.40	398.16	33.43	14.92	48.35	183.02	34.63	12.96	47.59	187.89
1.5	1526x355x424	98.38%	62.25	99.75	162.00	3045.41	33.54	134.56	168.10	934.25	33.96	135.79	169.75	1064.62

Table 6.8: Execution time and memory consumption results from 3 different flightlines.

	Empty Voxels (Percentage)	Construction Time (Seconds)	Polygonisation Time (Seconds)	Total Time(Seconds)	Maximum Memory (MBytes)
1					
2					
3					

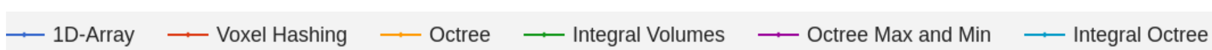


Table 6.9: Chart diagrams generated using the results from Table 6.8. The 1st flightline is from the New Forest Dataset (LDR-FW-FW10_01-201009822.LAS). The 2nd flightline is from the Dennys Wood dataset (LDR-FW10_01-201018713.LAS) and the 3rd one from Eaves Wood (LDR-FW-GB12_04-2014-083-13.LAS).

6.9 Discussion

Overall, ‘Integral Volumes’, the main new approach proposed, is the fastest one but it consumes as much memory as the original ‘1D-Array’. It’s performance is better than the ‘Octree Max and Min’ because:

- Elements are accessed in constant time while traversing a tree requires at least O(log n) time, when the tree is balanced, and up to O(n) for unbalanced trees.
- The size of the volume is the original cuboid while any octree data structure requires a cubic space that is a power of two. This results into extending the boundaries of the 3D voxelised FW LiDAR, including big empty areas and building deeper and unbalanced trees (increased traversal time).
- Neighbours finding is faster than octrees since no backtracking is required.
- Checking whether a surface is crossing the edges of an empty area is much faster using the ‘Integral Volumes’ because the sum of any volume is calculated in constant time. Therefore checking whether the neighbours are empty as well is trivial. While for the ‘Octree Max/Min’ and ‘Integral Tree’ data structures, it’s required to loop through all the voxels at the edges of an empty branch to avoid generating holes.

Regarding the ‘Voxel Hashing’, faster results were expected than the ‘Octree’ because it doesn’t required traversal for reaching elements, but it’s very likely to have more jumps in memory, considering that the implementation of the octree structures, keeps the children of every branch coherent in memory for faster interpretation.

Furthermore, ‘Octree Max and Min’ and the ‘Integral Octree’ have similar results. In the tests, the isolevel was set lower than the noise threshold and for that reason the empty branches were the ones discarded at the tests (Line 6 of Algorithm 3). If the isolevel was lower than the noise threshold, then the low level noise would have affected the ‘Octree Max and Min’ less than the ‘Integral Octree’; the ‘Octree Max and Min’ check whether the max value is below the threshold, while the ’Integral Octree’ the sum of the leaves. Additionally, ‘Integral Octree’ consumes more memory for saving the leaves into an 1D-array, but even though ‘Integral Ocree’ generally performed worse than the ‘Octree Max/Min’ in the tests of Table 6.7 and 6.8, it should be beneficial in multi-resolution direct volumetric rendering and blurring the volume for noise removal.

To sum up, ‘Integral Volumes’ is a new and simple algorithm presented in this thesis and it is the fastest one for the surface reconstruction of voxelised FW LiDAR in comparison to ‘Voxel Hasing’ and octrees.

Chapter 7

Alignment with Hyperspectral Imagery

Chapter 8

Detection of Dead Standing Eucalyptus For Managing Biodiversity in Native Australian Forest

Chapter 9

Comparison with Discrete Data

Furthermore, DASOS allows the user to choose whether the waveform samples or the discrete returns are inserted into the 3D density volume. Each sample or each return has a hit point and an intensity value. So, in both case the space is divided into 3D voxels and the intensity of each return or sample is inserted into the voxel it lies inside.

In general the results of discrete returns contain less information compared to the results from the FW LiDAR, even though the FW LiDAR contain information from about half of the emitted pulses (Section 3). As shown on the 1st example of table 3 the polygon mesh generated from the FW LiDAR contains more details comparing to the one created from the discrete LiDAR. The forest on the top is more detailed, the warehouses in the middle have a clearer shape and the fence on the right lower corner is continuous while in the discrete data it is disconnected and merged with the aliasing.

FW LiDAR polygons, compared to the discrete LiDAR ones, contain more geometry below the outlined surface of the trees. On the one hand this is positive because they include much information about the tree branches but on the other hand the complexity of the objects generated is high. A potential use of the polygon representations is in movie productions: instead of creating a 3D virtual city or forest from scratch, the area of interest can be scanned and then polygonised using our system.

But for efficiency purposes in both animation and rendering, polygonal objects should be closed and their faces should be connected. Hence, in movie productions, polygons generated from the FW LiDAR will require more post-processing in comparison with object generated from the discrete LiDAR.

Example 2 in table 3 shows the differences in the geometry complexity of the discrete and FW polygons using the x-ray shader of Meshlab. The brighter the surface appears the more geometry exists below the top surface. The brightness difference between area 1 and area 2 appears less in the discrete polygon.

Nevertheless, the trees in area 2 are much taller than in area 1, therefore more geometry should have existed in area 2 and sequentially be brighter. But the two areas are only well-distinguished in the FW LiDAR. On average the FW polygon is brighter than the discrete polygon, which implies higher geometry complexity in the FW polygon.

The comparison example 3 is rendered using the Radiance Scaling shader of Meshlab (Vergne et al, 2010). The shader highlights the details of the mesh, making the comparison easier. Not only the FW polygons are more detailed but also holes appear on the discrete polygons. The resolution of the voxels of those examples is 1.7m3 is, the bigger the holes are, while the full-waveform can be polygonised at a resolution of 1m3 without any significant holes. Figure 4 shows an example of rendering the same flightline of examples 3 at the resolution of 1m3 LiDAR data.

The last two examples (4 and 5) compare the side views of small regions. On the one hand the top of the trees are better-shaped in the discrete data. This may occur either because the discrete data contain information from double pulses than the FW data (Section 3) or because the noise threshold of the waveforms is not accurate and the top of the trees appear noisier on the FW LiDAR data. On the other hand more details appear close to the ground on the FW LiDAR data.

*** left during copying :s (and the higher the resolution, using FW)

Chapter 10

Overall Results

Chapter 11

Conclusions

11.1 Contributions

Bibliography

[1] T. Elmqvist, C. Folke, M. Nyström, G. Peterson, J. Bengtsson, B. Walker, and J. Norberg, "Response diversity, ecosystem change, and resilience," *Frontiers in Ecology and the Environment*, vol. 1, no. 9, pp. 488–494, 2003.

[2] D. U. Hooper, F. S. Chapin Iii, J. J. Ewel, A. Hector, P. Inchausti, S. Lavorel, and B. Schmid, "Effects of biodiversity on ecosystem functioning: a consensus of current knowledge," *Ecological monographs*, vol. 75, no. 1, pp. 3–35, 2005.

[3] D. B. Lindenmayer and J. T. Wood, "Long-term patterns in the decay, collapse, and abundance of trees with hollows in the mountain ash (eucalyptus regnans) forests of victoria, southeastern australia," *Canadian Journal of Forest Research*, vol. 40, no. 1, pp. 48–54, 2010.

[4] R. L. Goldingay, "Characteristics of tree hollows used by australian birds and bats," *Wildlife Research*, vol. 36, no. 5, pp. 394–409, 2009.

[5] P. Gibbons and D. Lindenmayer, *Tree Hollows and Wildlife Conservation in Australia*. CSIRO Publishing, 2002.

[6] "Animal pests: Poss." http://www.doc.govt.nz/conservation/threats-and-impacts/animal-pests/animal-pests-a-z/possums/. Accessed: 19th of September 2014.

[7] D. H. DeHayes, P. G. Schaberg, G. J. Hawley, and G. R. Strimbeck, "Acid rain impacts on calcium nutrition and forest health alteration of membrane-associated calcium leads to membrane destabilization and foliar injury in red spruce," *BioScience*, vol. 49, no. 10, pp. 789–800, 1999.

[8] J. Holmgren, "Prediction of tree height, basal area and stem volume in forest stands using airborne laser scanningce," *Scandinavian Journal of Forest Researcht*, vol. 19, no. 6, pp. 543–553, 2004.

[9] S. G. Aracil and R. B. A. Herries, D.L, "Evaluation of an additional lidar metric in forest inventory," *Proceedings of Silvilaser*, 2015.

[10] M. J. Harper, M. A. McCarthy, R. Van Der Ree, and J. C. Fox, "Overcoming bias in ground-based surveys of hollow-bearing trees using double-sampling," *Forest Ecology and Management*, vol. 190, no. 2, pp. 291–300, 2004.

[11] L. Rayner, M. Ellis, and J. E. Taylor, "Double sampling to assess the accuracy of ground-based surveys of tree hollows in eucalypt woodlands," *Forest Ecology and Management*, vol. 36, no. 3, pp. 252–260, 2011.

[12] R. B. Smith, *Introduction to Hyperspectral Imaging*. MicroImages, 2014.

[13] W. Wanger, A. Ullrich, T. Melzer, C. Briese, and K. Kraus, "From single-pulse to ful-waveform airborne laser scanners," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, pp. 100–112, 2004.

[14] A. Wehr and U. Lohr, "Airborne laser scanning - an introduction and overview," *ISPRS Journal of Photogrammerty and Remote Sensing*, vol. 54, pp. 68–82, 1999.

[15] C. Mallet and F. Bretar, "Full-waveform topographic lidar: State-of-the-art," *ISPRS Journal of Photogrametry and Remote Sensing*, vol. 64, pp. 1–16, 2009.

[16] K. Anderson, S. Hancock, M. Disney, and K. Gaston, "Is waveform worth it? a comparison of lidar approaches for vegetation and landscape characterization," *Remote Sensing in Ecology and Conservation*, 2015.

[17] A. Chauve, C. Mallet, F. Bretar, S. Durrieu, M. Deseilligny, and W. Puech, "Processing full-waveform lidar data: Modelling raw signals," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2007.

[18] *LAS Specification version 1.3-R1*. Bethesda, Maryland: American Society for Photogrammetry and Remote Sensing, 2010.

[19] M. Warren, *Full Waveform Upgrade*. NERC ARSF wiki, 2012.

[20] M. J. Sumnall, R. A. Hill, and S. A. Hinsley, "Comparison of small-footprint discrete return and full waveform airborne lidar data for estimating multiple forest variables," *Remote Sensing of Environment*, vol. 173, pp. 214–223, 2016.

[21] K. H. R. A. . Z. A. Lehner, H., "Consideration of laser pulse fluctuations and automatic gain control in radiometric calibration of airborne laser scanning data.," *Proceedings of 6th ISPRS Student Consortium and WG VI/5 Summer School*, 2011.

[22] I. Korpela, H. O. Ørka, H. V. Hyyppä, J., and T. Tokola, "Range and agc normalization in airborne discrete-return lidar intensity data for forest canopies," vol. 65, no. 4, pp. 369–379, 2010.

[23] M. Isenburg, *LAStools - efficient tools for LiDAR processing.* rapidlasso.

[24] M. Warren, B. Taylor, M. Grant, and J. D. Shutler, "Data processing of remorely sensed airborne hyperspectral data using the airborne processing library (apl)," *ScienceDirect, Computers and Geosciences*, vol. 64, 2014.

[25] M. Isenburg, "Pulsewaves: An open, vendor-neutral, stand-alone, las-compatible full waveform lidar standard.," 2012.

[26] A. Persson, U. Soderman, J. Topel, and S. Ahlberg, *Visualisation and Analysis of full-waveform airborne laser scanner data.* V/3 Workshop, Laser scanning 2005, 2005.

[27] M. Miltiadou, M. A. Warren, M. Grant, and M. Brown, "Alignment of hyperspectral imagery and full-waveform lidar data for visualisation and classification purposes," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 7, p. 1257, 2015.

[28] W. Wanger, A. Ullrich, V. Ducic, T. Maizer, and N. Studnicka, "Gaussian decompositions and calibration of a novel small-footprint full-waveform digitising airborne laser scanner," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 60, pp. 100–112, 2006.

[29] A. Neuenschwander, L. Magruder, and M. Tyler, "Landcover classification of small-footprint full-waveform lidar data," *Jounal of Applied Remote Sensing*, vol. 3, no. 1, pp. 033544–033544.

[30] J. Reitberger, P. Krzystek, and U. Stilla, "Analysis of full waveform LiDAR data for tree species classification," *International Journal of Remote Sensing*, vol. 29, no. 5, pp. 1407–1431, 2008.

[31] A. Chauve, F. Bretar, S. Durrieu, M. Pierrot-Deseilligny, and W. Puech, "Fullanalyze: A research tool for handling, processing and analysing full-waveform lidar data," *IEEE International Geoscience and Remote Sensing Symposium*, 2009.

[32] P. Bunting, J. Armston, D. Clewley, and R. M. Lucas, "Sorted pulse data (spd) library—part ii: A processing framework for lidar data from pulsed laser systems in terrestrial environments," *Computers & Geosciences*, vol. 56, pp. 207–215, 2013.

[33] L. Cao, N. Coops, L. Innes, J. Dai, and H. Ruan, "Tree species classification in subtropical forests using small-footprint full-waveform lidar data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 49, pp. 39–51, 2016.

[34] S. Hancock, K. Anderson, M. Disney, and K. J. Gaston, "Measurement of fine-spatial-resolution 3d vegetation structure with airborne waveform lidar: Calibration and validation with voxelised terrestrial lidar.," *Remote Sensing of Environment*, vol. 188, pp. 37–50, 2017.

[35] M. Miltiadou, M. Grant, M. Brown, M. Warren, and E. Carolan, "Reconstruction of a 3d polygon representation from full-wavefrom lidar data," *RSPSoc Annual Conference 2014, New Sensors for a Changing World*, 2014.

[36] R. Crippen, "Calculating the vegetation index faster," *Remote Sensing of Environment*, vol. 34, no. 1, pp. 71–73, 1990.

[37] R. N. Clark, G. A. Swayze, R. Wise, K. E. Livo, T. Hoefen, R. F. Kokaly, and S. J. Sutley, "Usgs digital spectral library splib06a," *US Geological Survey, Digital Data Series*, vol. 231, 2007.

[38] P. Hanrahan, "Ray tracing algebraic surfaces," *ACM SIGGRAPH Computer Graphics*, vol. 17, no. 3., 1983.

[39] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The volumepro real-time ray-casting system," *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 251–260, 1999.

[40] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–55, 2008.

[41] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann, "Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering," *Proceedings of the 2009 symosium on Interactive 3D graphics and games*, pp. 15–22, 2009.

[42] J. F. Blinn, *A Generalization of Algebraic Surface Drawing*, vol. 1. ACM Transactions on Graphics (TOG).

[43] A. Pasko and V. Savchenko, *Blending operations for the functionally based constructive geometry.* 1994.

[44] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” *ACM Siggraph Computer Graphics*, vol. 21, pp. 163–169, 1987.

[45] M. Levoy, “Volume rendering: Display of surfaces from volume data,” *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, 1998.

[46] M. Hadwiger, J. Beyer, W. K. Jeong, and H. Pfister, “Interactive volume exploration of patascale microscopy data streams using visualizatin-driven virtual memory approach,” *IEEE Transactions on Visualisation and Computer Graphics*, 2012.

[47] S. Laine and T. Karras, “Efficient sparse voxel octrees,” *Visualization and Computer Graphics, IEEE Transactions*, vol. 17, no. 8, pp. 1048–1059, 2011.

[48] B. Rodrigues de Araujo and J. A. Pires Jorge, *Adaptive polygonization of implicit surfaces*, vol. 29. Science Direct, Computer and Graphics, 2005.

[49] E. Hartmann, “A marching method for the triangulation of surfaces,” *The Visual computer 14, no*, vol. 14, no. 3, pp. 95–108, 1998.

[50] C. D. Hansen and P. Hinker, “Massively parallel isosurface extraction,” *Proceedings of the 3rd conference on Visualization ’92*, pp. 77–83, 1992.

[51] C. Galbraith, P. MacMurchy, and B. Wyvill, *BlobTree Trees.* IEEE Computer Graphics International, 2004.

[52] H. Sutter and A. Alexandrescu, *C++ Coding Standards: 101 Rules, Guidlines, and Best Practices.* United States: Addison-Wesley, 2004.

[53] J. Wilhelms and A. Van Gelder, “Octrees for faster isosurface generation,” vol. 24, no. 5, 1990.

[54] . W. J. Schaefer, S., “Dual marching cubes: Primal contouring of dual grids,” *Computer Graphics Forum*, vol. 24, no. 2, 2005.

[55] J. Wilhelms and A. Van Gelder, “Octrees for faster isosurface generation,” *ACM Transactions on Graphics (TOG)*, vol. 11, no. 3, pp. 201–227, 1992.

[56] S. F. Gibson, “Constrained elastic surface nets: Generating smooth surfaces from binary segmented data,” *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 888–898, 1998.

[57] B. Chazelle and L. J. Guibas, “Fractional cascading: I. a data structuring technique.,” *Algorithmica.*

[58] K. Museth, "Vdb: High-resolution sparse volumes with synamic topology," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, p. 27, 2013.

[59] M. S. Warren and J. K. Salmon, "A parallel hashed oct-tree n-body algorithm," *In Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, pp. 12–21, 1993.

[60] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 2, p. 169, 2013.

[61] F. C. Crow, "Summed-area tables for texture mapping," *ACM Computer Graphics*, vol. 18, no. 3, pp. 207–212, 1984.

[62] S. Hanan, "Neighbor finding in images represented by octrees," *Computer Vision, Graphics, and Image Processing*, vol. 46, no. 3, pp. 367–386, 1989.

[63] G. Schrack, "Finding neighbors of equal size linear quadtrees and octrees in constant time," *CVGIP: Image Understanding*, vol. 55, no. 3, pp. 221–230, 1992.

[64] R. Lohner, "Robust, vectorized search algorithms for interpolation on unstructured grids," *Journal of Computational Physics*, vol. 118, no. 2, pp. 380–387, 1995.

[65] R. Castro, T. Lewiner, H. Lopes, G. Tavares, and A. Bordignon, "Statistical optimisation of octree searches," *Computer Graphics Forum*, vol. 27, no. 6, pp. 1557–1566, 2008.

[66] M. L. Clark, D. A. Roberts, J. J. Ewel, and D. B. Clark, "Estimation of tropical rain forest aboveground biomas with small-foorprint lidar and hyperspectral sensors," *ScienceDirect, Remote Sensing of Enviroment*, vol. 115.

[67] J. E. Anderson, L. C. Plourde, M. E. Martin, B. H. Braswell, M. L. Smith, R. O. Dubayah, M. A. H. Dubayah, and J. B. Blair, "Integrating waveform lidar with hyperspectral imagery for inventory of a northern temperate forest," *Remote Sensing of Environment*, vol. 112, no. 4, pp. 1856–1870, 2008.

[68] H. Buddenbaum, S. Seeling, and J. Hill, "Fusion of full-waveform lidar and imaging spectroscopy remote sensing data for the characterization of forest stands," *International Journal of Remote Sensing*, vol. 32, no. 13, pp. 4511–4524, 2013.

[69] J. Heinzel and B. Koch, "Investigating multiple data sources for tree species classification in temperate forest and use for single tree delineation," *International*

Journal of Applied Earth Observation and Geoinformation, vol. 18, pp. 101–110, 2012.

[70] R. G. Congalton, "A review of assessing the accuracy of classifications of remotely sensed data," *Remote Sensing of Enviroment*, vol. 37, no. 1.

[71] J. F. Franklin, H. H. Shugart, and M. E. Harmon, "Tree death as an ecological process," *BioScience*, vol. 17, no. 8, pp. 550–556, 1987.

[72] J. Siitonen, "Forest management, coarse woody debris and saproxylic organisms: Fennoscandian boreal forests as an example," *Ecological bulletins*, pp. 11–41, 2001.

[73] I. Hanski, "Extinction debt and species credit in boreal forests: modelling the consequences of different approaches to biodiversity conservation," *Annales Zoologici Fennici*, pp. 271–280, 2000.

[74] G. Peterson, C. R. Allen, and C. S. Holling, "Ecological resilience, biodiversity, and scale.," *Ecosystems*, vol. 1, no. 1, pp. 6–18, 1998.

[75] N. Abrego and I. Salcedo, "How does fungal diversity change based on woody debris type? a case study in northern spain," *Ekologija*, vol. 57, no. 3, 2011.

[76] J. N. Stokland and K. H. Larsson, "Legacies from natural forest dynamics: Different effects of forest management on wood-inhabiting fungi in pine and spruce forests," *Forest Ecology and Management*, vol. 261, no. 11, pp. 1707–1721, 2011.

[77] D. Lonsdale, M. Pautasso, and O. Holdenrieder, "Wood-decaying fungi in the forest: conservation needs and management options," *European Journal of Forest Research*, vol. 127, no. 1, pp. 1–22, 2008.

[78] "List of extinct, threatened and near threatened australian birds," *Environment Protection and Biodiversity Conservation Act*, 1999.

[79] Government of Western Australia, "Oarks and wildlife the list of threatened and priority fauna list," tech. rep., November 2015.

[80] Y. Kim, Z. Yang, W. B. Cohen, D. Pflugmacher, C. L. Lauver, and J. L. Vankat, "Distinguishing between live and dead standing tree biomass on the north rim of grand canyon national park, usa using small-footprint lidar data.," *Remote Sensing of Environment*, vol. 113, no. 11, pp. 2499–2510, 2009.

[81] P. Polewski, W. Yao, M. Heurich, P. Krzystek, and U. Stilla, "Detection of fallen trees in als point clouds using a normalized cut approach trained by simulation,"

ISPRS Journal of Photogrammetry and Remote Sensing, vol. 105, pp. 252–271, 2015.

[82] W. Mücke, B. Deák, H. M. Schroiff, A., and N. Pfeifer, "Detection of fallen trees in forested areas using small footprint airborne laser scanning data," *Canadian Journal of Remote Sensing*, vol. 139, no. s1, pp. S32–S40, 2013.

[83] W. Yao, P. Krzystek, and M. Heurich, "Identifying standing dead trees in forest areas based on 3d single tree detection from full-waveform lidar data," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. I-7, pp. 359–364, 2012.

[84] J. Pasher and D. J. King, "Mapping dead wood distribution in a temperate hardwood forest using high resolution airborne imagery," *Forest Ecology and Management*, vol. 258, no. 7, pp. 1536–1548, 2009.

[85] I. Shendryk, M. Broich, M. G. Tulbure, A. McGrath, D. Keith, and S. V. Alexandrov, "Mapping individual tree health using full-waveform airborne laser scans and imaging spectroscopy: A case study for a floodplain eucalypt forest," *Remote Sensing of Environment*, vol. 187, pp. 202–217, 2016.

[86] S. C. Popescu, R. H. Wynne, and R. F. Nelson, "Measuring individual tree crown diameter with lidar and assessing its influence on estimating forest volume and biomass," *Canadian journal of remote sensing*, vol. 29, no. 5, pp. 564–577, 2003.

[87] L. Jing, B. Hu, J. Li, and T. Noland, "Automated delineation of individual tree crowns from lidar data by multi-scale analysis and segmentation," 2012.

[88] B. Hu, J. Li, L. Jing, and A. Judah, "Improving the efficiency and accuracy of individual tree crown delineation from high-density lidar data," 2014.

[89] S. C. Popescu and K. Zhao, "A voxel-based lidar method for estimating crown base height for deciduous and pine trees," *Remote sensing of environment*, vol. 112, no. 3, pp. 767–781, 2008.

[90] I. Shendryk, M. Broich, M. G. Tulbure, and S. V. Alexandrov, "Bottom-up delineation of individual trees from full-waveform airborne laser scans in a structurally complex eucalypt forest," *Remote Sensing of Environment*, vol. 173, pp. 69–83, 2016.

[91] J. L. Lovell, D. L. B. Jupp, G. J. Newnham, N. C. Coops, and D. S. Culvenor, "Simulation study for finding optimal lidar acquisition parameters for forest height retrieval.."

[92] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *Computer Vision and Pattern Recognition*, vol. 1, 2001.

[93] P. Dong, “Characterization of individual tree crowns using three-dimensional space signatures derived from lidar data.,” *International Journal of Remote Sensing*, vol. 30, no. 24, pp. 6621–6628, 2009.

[94] M. Sumnall, *Assessment of habitat condition and conservation status for lowland British woodland using earth observation techniques.* Bournemouth University: Unpublished PhD thesis, 2013.

Appendix A

DASOS user guide

Appendix B

Case Study: Field Work in New Forest

B.1 Introduction

This section is a case study containing field work from a non-professional perspective to better understand the challenges of working remotely with forests. Remotely sensed data contain a great amount of information but in order to build a good system for identifying trees and materials, an in depth knowledge of them is required [12]. For that reason, this case study was created; information about the New Forest, which is a forest in the south of United Kingdom, were collected and a small validation dataset was created. The dataset created includes the tree species and approximate heights of the trees in two areas of interest.

Before travelling to the New Forest, two areas of interest were selected. These areas were selected according to the following criteria:

- There were LiDAR data of the selected area to be able to compare what we can see in real life with the scanned data
- Areas that had a variation of tree species were selected. This was done according to the (non-validated) results of a thesis of Bournemouth University that classified the tree species of the New Forest [94]. This helped get a broader range of tree species.

The following sections give a detailed description of the information gathered during the trip. This includes the species and height maps generated, the different types of landscapes found and the challenges faced.

B.2 Validation Data Collected

The tree classes were initially defined by the provided Bournemouth thesis [94]. A colour was chosen for each tree class and, while being in the New Forest, the aim was to mark each tree on the paper map with the corresponding colour. Using QGIS (Quantum Geographic Information System) the classification results of the forest assessment, undertaken by Sumnall in 2013 [94], were coloured with the same colours to ease comparison.

At the aforementioned forest assessment, there were 26 classes from 14 different species; the remaining 12 classes were young versions of the 14 species. Here the classes are reduced to 14 by merging all the young trees into the tree species classes (in the 4 years gap between the 2010 assessment and and the visit to new Forest in 2014, the young trees would have aged). See table B.1 for the initial 14 classes. Nevertheless, more tree species existed in the areas of interest in New Forest than those 14 classes. The colours and symbols of the extra tree classes are shown on table B.2.

Tree	Colour
1. Beech	
2. Oak	
3. Silver Birch	
4. Sweet Chestnut	
5. Corsican Pine	
6. Coast Redwood	
7. Douglas Fir	
8. Grand Fir	
9. Japanese Larch	
10. Lawson Cypress	
11. Norway Spruce	
12. Scots Pine	
13. Western Hemlock	
14. Common Adler	

Table B.1: Colours of the initial 14 classes

During the visit, tree species maps were generated for a few square meters. The position of the trees were found relative to easily-spotted refereence points (e.g. road crossing) that were marked in advanced. That was done because, according to Dr. Ross Hill, no GPS can be accurate enough when trees are around since the satellite signal bounces off the leaves and reduces the positioning accuracy. In professional fieldwork, a total station is used but, for the purposes of this visit, it was not considered necessary. By the end of the case study, ground maps were coloured according to the tree species

Tree	Colour / Symbols
15. Ash	A
16. Hawthorn	Blue pen colour
17. Malus (Crabapple)	Highlighter
18. Holly Tree	
19. Trees that have been cut down	x
20. Trees that are mixed together	// (added on top of the normal colour)

Table B.2: Classes that were added during the trip

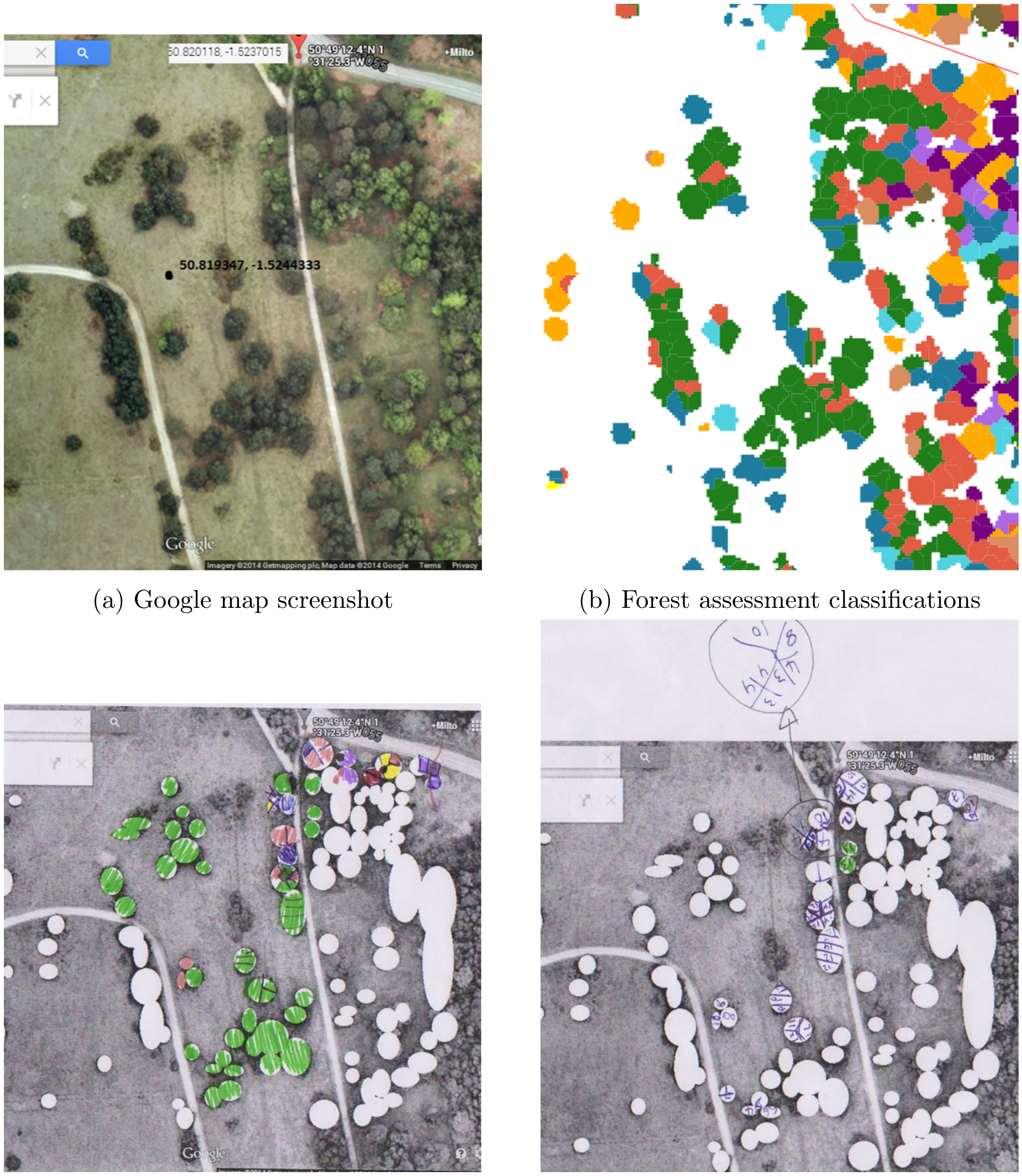
identified and estimates of the approximate heights of the trees were also noted down.

The following four maps were created for each selected area. The first two maps were created before the trip during preparation, while the last two contain the information collected during the field work.

- a screen shot of the area from Google map,
- the classification results from the forest assessment [94],
- the coloured tree species map and
- the approximated height map.

Comparing the validation dataset created with the classifications done at Bournemouth University (which were not validated), it is clear there are misclassifications. This is shown in Figures B-1 and B-2 and it is likely that is occurs due to the over-segmentation of trees. Those wrong classifications justify that validation and field work data are essential for building a good classifier.

The first area is included in the LAS file named LDR-FW-FW10_01-201018715.LAS and it lies inside the limits: X = (433453 - 433761), Y = (102193 - 102405) [British National Grid coordinates]. The four maps that relate to these areas are shown in Figure B-1.



(a) Google map screenshot

(b) Forest assessment classifications

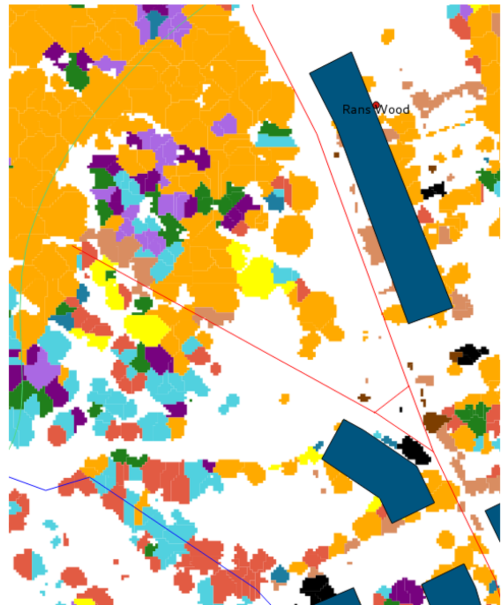
(c) Tree species map, from field work

(d) Approximate heights of the trees, from field work

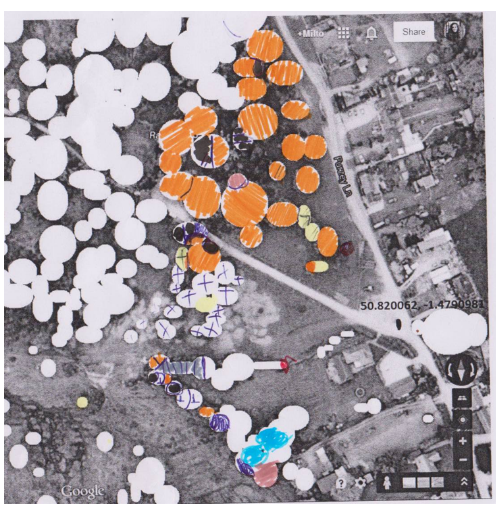
Figure B-1: The first area of interest and the related maps.

The second area is included in the LAS files named LDR-FW-FW10_01-201018719.LAS and LDR-FW-FW10_01-201018718.LAS and it lies inside the limits: X = (436442 - 436835), Y = (102334 - 102585) [British National Grid coordinates]. The four maps created for these areas are shown in Figure B-2.

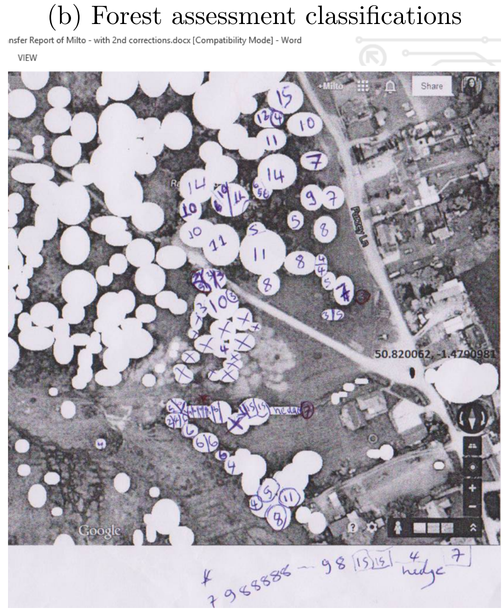
(a) Google map screenshot



(b) Forest assessment classifications



(c) Tree species map, from field work



(d) Approximate heights of the trees, from field work

Figure B-2: The second area of interest and the related maps.

- > Wrong classification due to over segmentation of the trees.

B.3 Landscape types

During the forest assessment in New Forest, not only validation data were collected, but also useful information about classifying the data. The following images show examples of the five landscape types that were found in New Forest:

1. Heather fields:

Figure B-3: Trees that have been cut down

2. Grass with a few scattered trees:

Figure B-4: Grass with a few scattered trees

3. Dense Forest:

Figure B-5: Dense forest

4. Bushes and Shrubs

Figure B-6: Trees that have been cut down

5. Lakes and rivers, which are more rarely found

Figure B-7: Lakes and rivers

Please note that the landscape types could significantly differ according to the scanned area. For example, the landscape of New Forest is flat while the landscape of Eaves Wood (another scanned forest in UK) is hilly. The landscape type should be taken into consideration during classifications.

B.3.1 Classification challenges

This case study brought further understanding of the challenges of creating validation data and writing a tree species classifier. These challenges are listed and explained below with some photos taken during field work:

1. Field work and remotely sensed data collection should happen around the same time to avoid changes that happens over time. In the New Forest case, the airborne data were collected in 2010 and many changes occurred in the intervening time - in the most extreme cases, some trees had been cut down.

2. Machine learning becomes more challenging as the number of classes increases. Regarding tree species classes, it is unrealistic to expect that all tree species will be identified. This point is underlined by the fact that the list of tree species used in the

Figure B-8: Trees that have been cut down

tree assessment held by Sumnall [94] didn't include a number of trees (e.g. holly trees and crabapple) that were widespread in New Forest.

3. There is much more than just trees in the forest, including mobile animals, that may confuse a classification if LiDAR returns hit rocks, animals, vehicles or buildings instead of branches, leaves and trunks. Any classification must account for inevitable errors due to non-target objects being in the scene.

Figure B-9: Animals in New Forest

4. Large validation datasets from a single area will not be sufficient, because trees of the same species are usually gathered together. For instance, the first selected area has many pine trees while the second one has many oak trees. Therefore, it is important to have many field plots spread well within the area of interest.

5. Further, some trees are entwined together which makes it difficult to identify from the data whether they are one or two trees. Examples are shown in Figure B-10; in the left image, the trunks of the two trees are very close to each other and, in the right image, a crabapple and an oak tree have grown together.

Figure B-10: Trees, which are mixed together

B.4 Conclusions and Discussion

To sum up, the trip to the New Forest was essential for better understanding the challenges of remote monitoring of forests. During the visit, a small validation dataset was generated; the species and height of trees that are inside the two areas of interest were noted down. Field work is a time consuming task and weeks are required for generating a big enough validation dataset, but it is essential for understanding the object of interest (trees) in relation to the scanned data. Challenges identified were also explained and this increased knowledge about forests should lead to implementing a better classifier.