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

COPYRIGHT

Signature of Author.....

Milto Miltiadou

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
DBH	Diameter at Breast Height
DEM	Digital Elevation Model
DTM	Digital Terrain Model (DTM)
FN	False Negative
FP	False Positive
FW	Full-Waveform
GB	Gigabyte
K-NN	K-Nearest Neighbour
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
TP	True Positive
TN	True Negative
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

Forest monitoring involves checking and observing the changes in the structure of the forests and their foliage over the years. It has a significant value in both sustainable and commercial forests, because it contributes to managing biodiversity, maintaining forest health and optimising wood trade procedures as explained below:

- **Biodiversity** plays a substantial role in ecosystem resilience [1] while various human activities affect biological communities by altering their composition and leading species to extinction [2]. For example, in Australian native forests many arboreal mammals and birds rely on hollow trees for shelters [3] . Hollow trees are trees that have hollows, which are semi-enclosed cavities on trunks and branches. They are formed by natural forces, like bacteria, fungi and insects and may take tens to hundreds of years to become suitable for animal/bird shelters. Unfortunately recent studies shown that it is likely to be a shortage of hollows available for colonisation in the near future [4] [5]. Therefore monitoring and protecting hollow trees has a positive impact in preserving biodiversity.

- **Forest Health**: Protecting vegetation from pests and diseases. An example of pests are the Brushtail Possums, which were initially brought to New Zealand for fur trade, but they have escaped and become a thread to native forests and vegetation [6]. In addition, anthropogenic factors have a negative impact to nature. For instance, acid rain is responsible for the freezing decease at red bruces because it reduces the membrane-associated calcium, which is important for tolerating cold [7]. Those changes in nature need to be monitor in order to preserve a healthy and resilience ecosystem.

- **Wood Trade**: Measuring stem volume and basal areas of trees contributes to forest planning and management [8]. For example, measuring stocking and wood quality would help into estimating the cost of harvesting the trees in relation to the stocking [9].

Traditionally, forest monitoring involves field work such as travelling into the area of interest and taking manual measurements. Regarding the need to monitor hollows, tree climbing with ladders and ropes gives very accurate results but is dangerous, expensive, time consuming, and cannot easily scale into surveying large forested areas [10] [11]. Therefore, automated ways of monitoring forests are essential and this is why Remote Sensing has a significantly positive impact in forestry.

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.

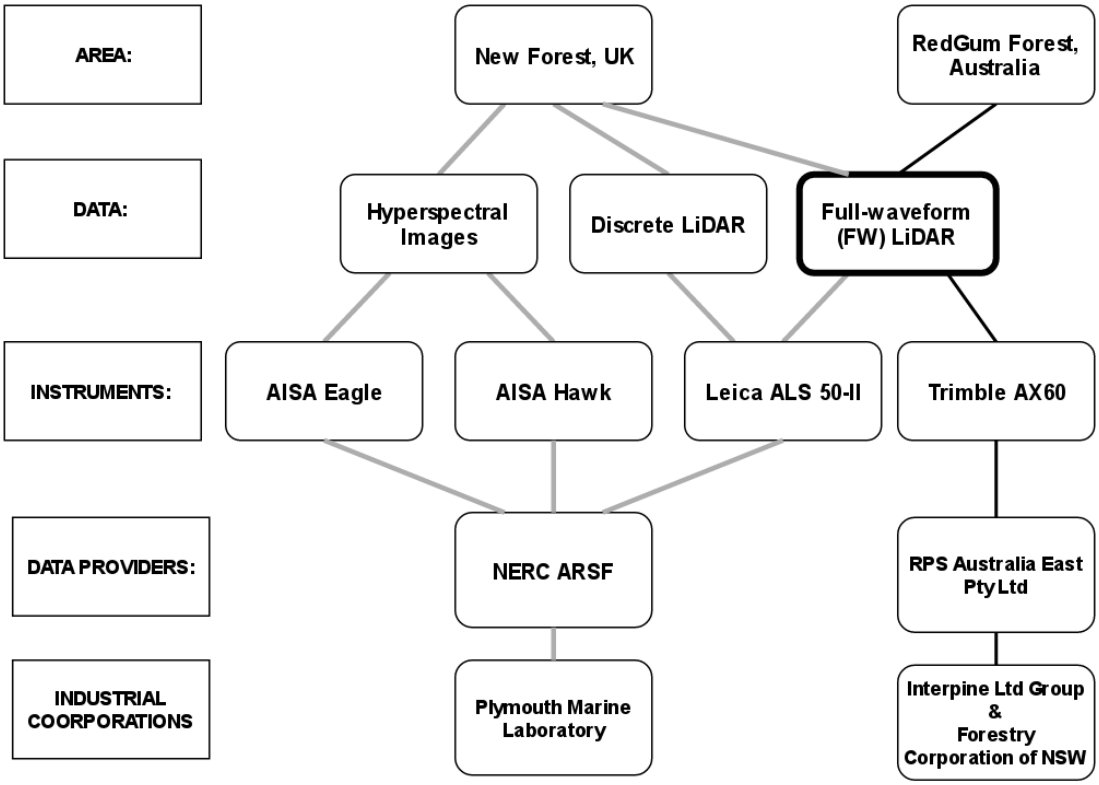
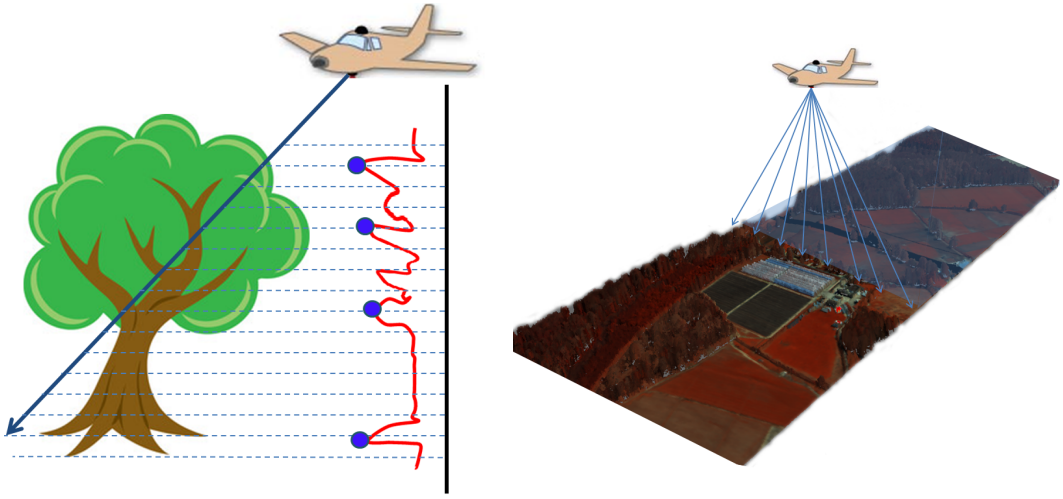


Figure 2-1: Data and Instruments



(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 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, which avoids recording the world position of every sample by 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 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.

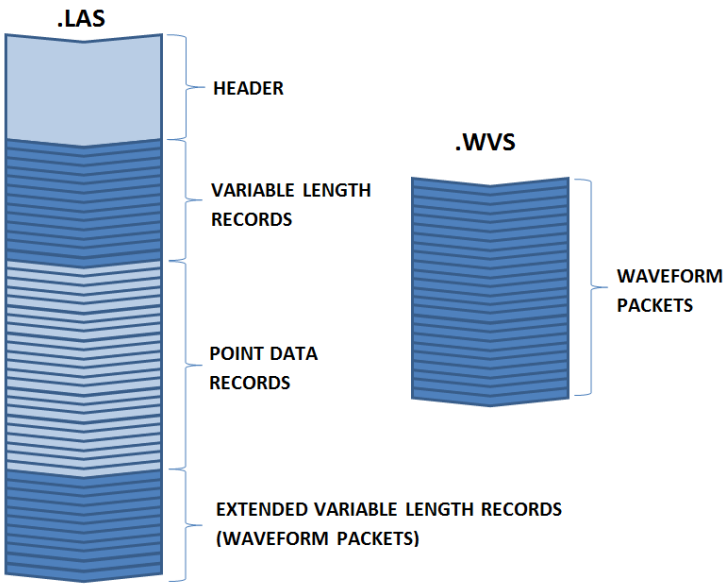


Figure 2-3: How the FW LiDAR data are stored into a binary file, according to the LAS1.3 file format specification

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.

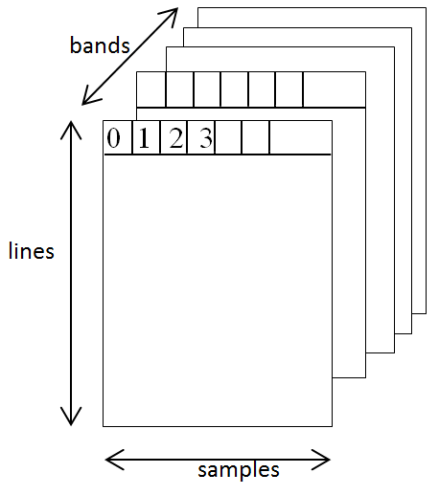


Figure 2-4: This figure shows the order of the hyperspectral pixels saved into the the binary .bil file.

- 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 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 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 FW LiDAR that will benefits biodiversity management.	8
8	Research whether terrain classification can be improved by the inference of high quality 3D information.	8

Table 3.1: Values of divisible sides

3.3 Overview

To address the limitations of existing workflows for using with FW data we developed the open source software DASOS (named after $\delta\alpha\sigma o\varsigma$, which means 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 extract a 3D surfaces from the scanned area. This research further introduces new data structures for accelerating the surface extraction processed.

- 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 feature vectors characterising objects like trees. 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.

To sum up, FW LiDAR has great potential to improving automated surveying accuracy and consequently reduce the expensive fieldwork conducted in forestry. 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, I occasionally receive emails with questions or suggestion about the software and according to github the visitors are slowly increasing; during the last forty days, there were six unique visitors and one cloner (Figure 3-1).

Figure 3-1: Github statistics from the 4th of March till the 14th of April

Chapter 4

The open source software DASOS and the Voxelisation Approach

As mentioned in Section 3.1, there are very few uses of FW LiDAR data because of the quantity of the recorded information. For that reason, DASOS was developed (Section 4.3) as an open source software, to help foresters without computer science background to use FW LiDAR data while simultaneously advancing the research goals of this thesis. In this section:

- An overview of related software packages is given and we explain how DASOS differs from those packages (Section 4.1).
- The main method of interpreting the data within DASOS (the voxelisation approach) is described (Subsection 4.2).
- All the functionalities of DASOS are listed (Section 4.3)
- and, finally, a summary is provided (Section 4.4).

4.1 State-of-Art FW LiDAR Software Packages

The most common approach for interpreting FW LiDAR is the Gaussian decomposition of the waveforms for peak-points extraction. Each waveform is modelled as a set of Gaussian pulses and for every Gaussian peak, a single return (equivalent to a discrete LiDAR point) is extracted [28]. Neunschwander et al used this approach for Landcover classification [29] while Reitberger et al applied it for distinguishing deciduous trees from coniferous trees [30]. Chauve et al further proposed an approach of improving the Gaussian model in order to increase the density of the points extracted from the

data and consequently improve point based classifications of FW LiDAR data [17]. The following tools are able to extract discrete points from the waveforms and visualise small areas of interest:

- **Pulsewaves**: visualises a small number of waveforms using different transparencies according to the intensities of the wave-samples and is able to generate discrete point clouds [25].
  Link: <https://rapidlasso.com/pulsewaves/>

- **FullAnalyze**: supports echo decomposition. Regarding visualisations, the user can select single trees from the Graphical User Interface (GUI) and, for each wave-sample, a sphere with radius proportional to its amplitude is created and visualised [31].
  Link: <http://fullanalyze.sourceforge.net/>

- **SPDlib**: exports discrete LiDAR from the waveforms and visualises either the samples that are above a threshold level as points or the extracted discrete point cloud. It also colours them according to their intensity value [32].
  Link: <http://www.spdlib.org/>

Echo decomposition and extraction of peak points identifies significant features and further enables the interpretation of the data within existing workflows and software that support discrete LiDAR data. For example, the discrete LiDAR can be analysed using:

- **Lag**: a visualisation tool for analysing and inspecting discrete LiDAR point clouds.
  Link: <http://arsf.github.io/lag/>

- **Quick Terrain Modeller** : a 3D discrete LiDAR points visualiser, that can generate Digital Elevation Models (DEM) and Digital Terrain Models (DTM).
  Link: <http://appliedimagery.com/>

- **LAStools** : a tool set that classifies noise, visualises point clouds, clips data.
  Link <https://rapidlasso.com/lastools/>

The DASOS approach to interpreting FW LiDAR data is fundamentally different from the aforementioned software packages. On the one hand, converting FW LiDAR into discrete peaks eases their usage, since existing workflows support discrete LiDAR. On the other hand, FW LiDAR contains information about pulse width that is usually

not preserved after peak point extraction. Also the comparison of point clouds depends on the density of the emitted pulses; problems arise with the sinusoidal scannng pattern of, for example, the Leica system, resulting in higher numbers of samples at the edges of the swath and lower in the middle. For these reasons, in DASOS, this information is accumulated from multiple shots into a voxel array, building up a 3D density volume. The correlation between multiple pulses in a voxel representation produces a more accurate and complete representation, which confers greater noise resistance and it further opens up possibilities of vertical interpretation of the data. The idea of voxelising FW LiDAR data is explained in the following section 4.2.

4.2 Voxelisation for Interpreting FW LiDAR data

Voxelisation of FW LiDAR data was first introduced by Persson et al., who used it to visualise waveforms using different transparencies [26] and it has been adopted as the future of FW LiDAR data with the literature moving toward that direction. In 2016-17, Cao et al used it for tree species identification [33], Hancock et al improved canopy height models of vegetation [34] and Sunmall et al characterised forest canopy from a voxelised vertical profile [20]. This innovative approach of voxelising the FW LiDAR data is an integral part of this thesis and it is used for both visualisations and classifications [35] [27].

The FW LiDAR data are voxelised by inserting the wave samples into a 3D regular grid and constructing a 3D discrete density volume. According to Persson et al, each wave sample is associated with the 3D cell, named voxel, that it lies inside. If multiple samples lie inside a voxel then the sample with the highest intensity is chosen [26]. In order to reduce noise, there are two differences between this approach and the way FW LiDAR data are voxelised in DASOS.

At first a threshold is used to remove low level noise, because when the length of a recorded waveform is longer than the distance between the first hit point and the ground, the system captures low signals for the remaining sampling time period after the pulse has been absorbed by the ground, which results pure noise. For that reason, the samples whose intensity is lower than a user-defined noise level/threshold are discarded.

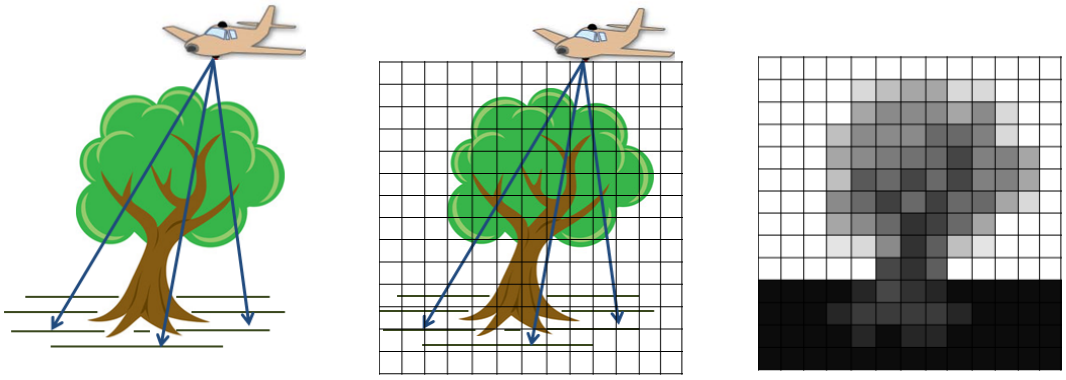
As before, each wave sample is associated with the voxel that it lies inside. The second difference is how DASOS overcomes the uneven number of samples per voxels, which is primarily caused by the differing angle at which the LiDAR shot passes through voxels directly below the sensor and those off to the sides. The intensity of each sample is the laser intensity returned during the corresponding time interval. For example, if 5 samples are inside a voxel and the waveform is digitised at 2ns, then the laser

intensity associated with that voxel corresponds to a 10ns waveform sampling length. For comparison purposes, it's essential to keep the waveform length consistent across the voxels. To overcome this issue in DASOS, the average intensity of the samples that lie inside each voxel is taken, instead of choosing the one with the highest intensity [26]. This way, the likelihood that the 3D volume will be affected by outliers and high noise is reduced. The following equation shows how the intensity value of a voxel is calculated:

$$I_v = \frac{\sum_{i=1}^{n} I_i}{n} \tag{4.1}$$

where I_v is the accumulated intensity of voxel v, n is number of samples associated with that voxel and I_i is the intensity of the sample i.

To sum up, during voxelisation, the area of interest is divided into voxels. The samples of the FW LiDAR data are inserted inside this 3D discrete density volume and normalised such that equally sized waveform length is saved inside each voxel. The result is a 3D discrete density volume of the scanned area. Figure 4-1 depicts this process in 2D.



(a) The sensor from the plane emits multiple pulses and collects information from the returned laser intensity.

(b) The area of interest is divided into equally sized cubes, named voxels, generating this way a discrete volume.

(c) The accumulated intensities of wave samples into the volume build up the voxelised representation of the scanned area.

Figure 4-1: The above images depict the voxelisation process of the FW LiDAR data in 2D. Please note that the voxelisation output in Figure 4-1c shows how ideally the result would look. But in reality, a number of trees may be disconnected from the ground due to missing information about their trunk. [1]

[1] The tree and plane images are taken from: http://images.clipartpanda.com/tree-clip-artKij4jKriq.jpeg & http://gmv.cast.uark.edu/wpcontent/uploads/2013/01/ALS_scematic.jpg

4.3 The functionalities of DASOS

So far, an overview of existing software packages supporting FW LiDAR was given (Section 4.1) and it was explained how DASOS differs from them by voxelising the waveforms (Section 4.2). In this section, the three main functionalities of DASOS are described in Tables 4.1, 4.2 and 4.3.

Each functionality is linked to a number of thesis sections, which describe the algorithms implemented and the related applications. In a few words, the 3D visualisations are useful in forestry for reducing fieldwork and improving planning of field trips (e.g. checking whether a road passes through a fieldplot area). The 2D metrics allow simultaneous interpretation of FW LiDAR data and hyperspectral imagery. They could also be used in GIS software. In this thesis, they are used for generating tree coverage maps. Last but not least the vector features that enable 3D feature detection and they are used for detecting dead standing trees.

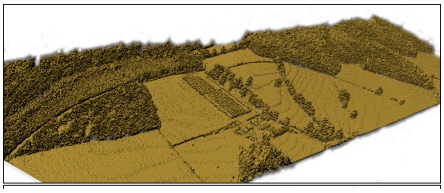
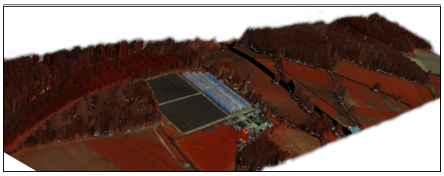
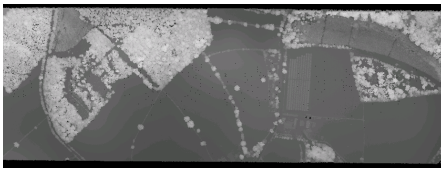
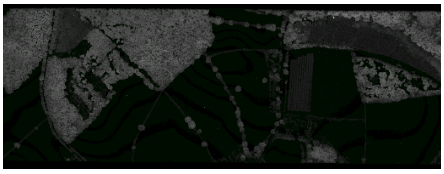
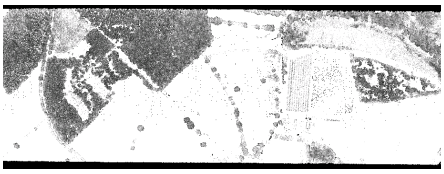
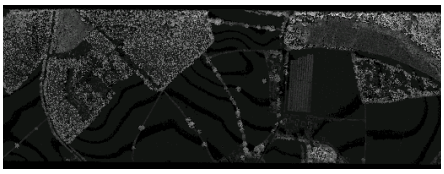
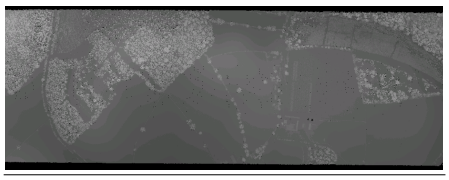
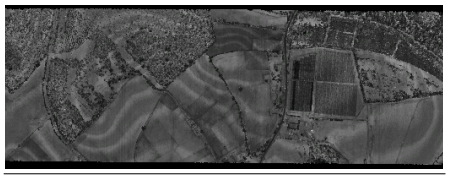
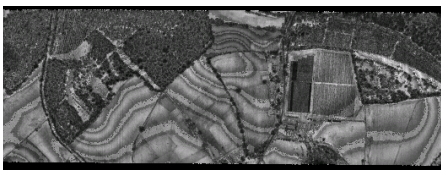
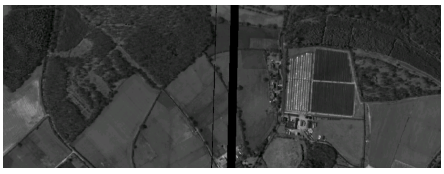
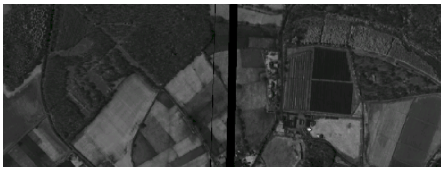
First Functionality: 3D Polygon Mesh Generation			
Input	**Description**	**Output Example**	**Output Format**
LAS1.3	**3D Polygon Mesh** Constructed from the volumetric representation (algorithms and user-defined parameters are explained in Section 5 while optimisation approaches are discussed in Section 6)		.obj
LAS1.3 and level 1 (.bil & .igm	**3D Coloured Polygon Mesh** Projecting 3 user-defined hyperspectral bands on the mesh (Section 7)		.obj & .png

Table 4.1: The 1st functionality of DASOS that generates 3D polygonal meshes.

Second Functionality: Generation of 2D metrics aligned with hyperspectral imagery			
In Section 7 a selection of the following metrics are used for generating tree coverage maps			
Input	**Metric Description** (L for LiDAR metrics & H for hyperspectral metrics)	**Output Example**	**Output Format**
LAS1.3	**L0 - Digital Elevation Model - DEM:** The distance between the top non-empty voxel and the lower boundaries of the volume.		.asc
LAS1.3	**L1 - Thickness:** The distance between the first and last non empty voxels in every column of the 3D volume.		.asc
LAS1.3	**L2 - Density:** Number of non-empty voxel over all voxels within the range from the first to last non-empty voxels.		.asc
LAS1.3	**L3 - First Patch:** The number of non-empty adjacent voxels, starting from the top non-empty voxel in that column.		.asc
LAS1.3	**L4: Last Patch:** The number of non-empty adjacent voxels, starting from the lower non-empty voxel in that column.		.asc

LAS1.3	**L5 - Edge detection:** The average height difference of neighbouring pixels		.asc
LAS1.3	**L6: Lowest Return** The height of the lowest non empty voxel		.asc
LAS1.3	**L7: Maximum Intensity** The maximum intensity of each column		.asc
LAS1.3	**L8: Average Intensity** The average intensity per column		.asc
LAS1.3 and level 1 (.bil & .igm)	**H0 : Mean** The mean of the hyperspectral spectrum.		.asc
LAS1.3 and level 1 (.bil & .igm)	**H1: Standard Deviation** [1] The standard deviation of the hyperspectral spectrum at each pixel.		.asc

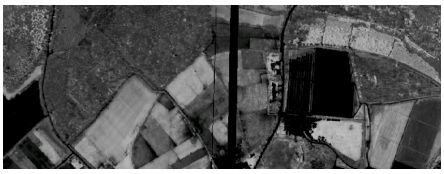
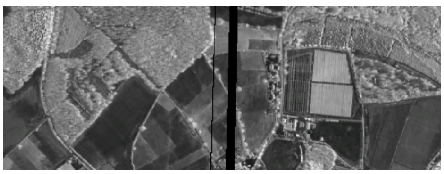
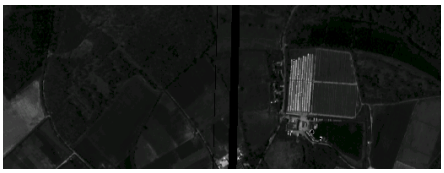
LAS1.3 and level 1 (.bil & .igm)	**H2: NDVI** The Normalised Difference Vegetation Index indicates whether green vegetation exists or not and it is derived from the electromagnetic spectrum as follow: $NDVI = \frac{NIR - VIS}{NIR + VIS}$ (4.2) where the NIR is the near-infrared region of the spectrum (700-2500nm) and VIS is the Visible/Visual spectrum (430-770) [36].		.asc
LAS1.3 and level 1 (.bil & .igm)	**H3: Spectral Signature** [1] The squared spectral difference between each pixels' spectrum and the generalised vegetation signature retrieved from USGS Digital Spectral Library [37].		.asc
LAS1.3 and level 1 (.bil & .igm)	**H4: Band** A single user defined hyperspectral band.	 	.asc .asc

Table 4.2: The 2nd functionality of DASOS that generates 2D metrics in ASCII format.

Third Functionality: 3D Priors / Signatures			
The list of feature vectors that characterise objects or local areas in 3D. In Section 8, the list of feature vectors are used to described dead trees and they are compared across the voxelised space for detecting dead standing trees			
Input	**Description**	**Output Example**	**Output Format**
LAS1.3	**List of feature vectors** with raw intensities	Please Look at Figure 4-2	.csv
LAS1.3	**List of feature vectors** with processed intensities	Please Look at Figure 4-3	.csv

Table 4.3: The three functionalities of DASOS

Index	centroid_x	centroid_y	V0_0_0	V0_0_1	V0_0_2	V0_0_3	V0_0_4	V0_1_0	V0_1_1	V0_1_2	V0_1_3	...
0	251836.109	6048994.5	7	14	10	26	0	0	9	10.25	11.875	...
1	251843.906	6048980.5	0	0	0	0	0	0	0	0	0	...
2	251846.312	6048979	9	60.75	70.75	13	8	0	0	0	7.667	...
3	251849.312	6049022.5	48.556	93.222	20.5	0	7	0	0	0	0	...
4	251851.703	6048988	100.2	53.222	10.5	7.143	0	0	0	0	47.25	...
5	251852.906	6048975	0	0	0	0	0	26.875	0	10.444	13.182	...
6	251857.109	6048974	0	0	0	0	0	45.667	93	16.333	7.25	...
7	251858.312	6049010.5	0	0	0	0	0	0	0	8	6	...
8	251860.703	6048984	0	45.75	8	7.333	0	0	0	0	6.8	...
9	251861.312	6049000	0	0	0	0	0	0	0	0	0	...

Figure 4-2: Example of an exported .csv file that contains a list of feature vectors with raw voxel intensities

Index	centroid_x	centroid_y	Height_Middle_Column	Height_Mean	Height_Median	Height_Std	Sum_Int_Diff_X	...
0	251836.109	6048994.5	36	35.5	36	0.943	95.125	...
1	251843.906	6048980.5	19.8	20.1	20.4	0.671	0	...
2	251846.312	6048979	16.8	16	15.6	1.02	169.167	...
3	251849.312	6049022.5	36	35.7	36.6	0.964	169.278	...
4	251851.703	6048988	17.4	16.2	16.2	0.346	408.065	...
5	251852.906	6048975	27	26.4	26.4	0.917	68.537	...
6	251857.109	6048974	17.4	17.4	18	0.849	162.25	...
7	251858.312	6049010.5	40.8	40	39.6	1.02	251.36	...
8	251860.703	6048984	17.4	16.6	16.2	0.663	67.883	...
9	251861.312	6049000	19.8	20.1	20.4	0.671	0	...

Figure 4-3: Example of exported .csv file that contains a list of feature vectors with processed information about voxels' intensities.

[1] The marked metrics of Table 4.2 were implemented specifically for the tree coverage maps [27] and they are not available on the released version of DASOS.

The complete user guide of DASOS is given in Appendix A. The user guide gives an in-depth explanation on how to use the available commands and variables to generate 3D coloured polygonal meshes, 2D metrics algined with hyperspectral imagery and lists of feature vectors. It also provides information on how to download DASOS with its source code and on what relevant forums exists for providing support.

4.4 Summary and Discussion

Along with supporting the research in this thesis, the open source software DASOS was developed to encourage foresters to use FW LiDAR data. The main way of interpreting FW LiDAR data in DASOS is fundamentally different from the state-of-art available software packages. In a few words, the FW LiDAR data are voxelised by inserting the wave samples into a 3D discrete density volume, which preserves an extra parameter (the echo width) in comparison to point extraction algorithms. It also accumulates intensity values from multiple shots and stores them into a 3D regular grid, resolving this way the problem with the sinusoidal footprint / uneven scanning pattern of the Leica system.

There are three main functionalities of DASOS: the construction of 3D polygon meshes, the generation of 2D metrics aligned with hyperspectral images andcharacterisation of objects using feature vectors. The visualisation outputs are also state-of-art since previous visualisations talk about points [32] or spheres [31], while DASOS is able to create closed polygon representation. In addition, the integration of various sensors allows simultaneous interpretation of their data and, in section 7, it is shown that this confers better results for generating tree coverage maps. The last feature of DASOS allow local inspection of data and they are used in section 8 for dead standing tree detection in native Australian forests.

Finally, it worth mentioning that there a few individuals/organisation that showed interest in using DASOS and, in the future, usage of it, it derivatives or the concepts employed in it are expected to increase in remote forest surveys (i.e. for commercial forest stocking estimation or for infected trees detection and treatment).

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 4.3) was developed along with this thesis to ease the usage of the data. DASOS voxelises (Section 4.2) 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 volumetric data (e.g. the 3D voxelised FW LiDAR data or MRI scans.).

The first approach is direct rendering, which continuously generates 2D images according to the position and rotation of the view point. This creates an interactive viewing of the scene/object, but an image needs to be generated every time the position or rotation of the view point is changed. A direct rendering approach is ray-tracing. Ray-tracing generates images by casting rays from the view point, passing through each pixel of the image to be generated and continuing until there is an intersection with an object from the scene (Figure 5-1). Intensity values are assigned to the pixels according to the nearest intersections between the ray and the scene [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].

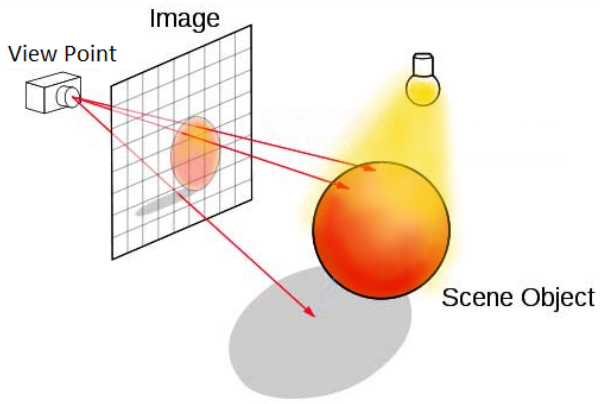


Figure 5-1: Ray-tracing illustration

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 4.2).

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 an 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 4.2), 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 5-6 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 algebraic representation 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).

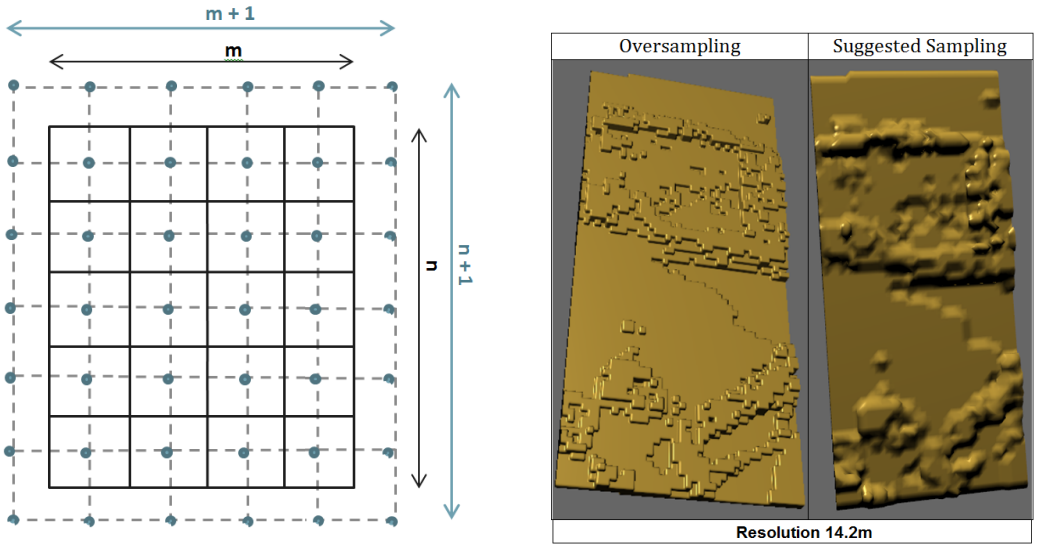
The Marching Cubes algorithm is an algorithm that extracts an iso-surface from the implicit volumetric field as a polygonal mesh using a look up table. Let's assume that $f(X)$ defines an implicit object. At first the space is divided into cubes. Each cube is defined by eight corner points and each corner point lies either inside or outside the iso-surface. 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 5-2; 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 5-2b 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

[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.

and once the resolution is increased then blur will disappear.



(a) Suggested Sampling

(b) Effects of Oversampling

Figure 5-2: The suggested sampling during polygonisation using the Marching Cubes Algorithm

5.5 Results

The output of DASOS is a polygonal mesh exported into a .obj file, which is a standard graphics format. The .obj files can be loaded into various animation software tools like Maya and Meshlab (Figure 5-3). Figure 5-4 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 A-1).

Furthermore, there are three main user-defined parameters and Figure 5-6 shows how the results are affected once modified:

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 4.2). 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 MakerBot. There are some difficulties as the meshes are not manifold[3] (Figure 5-7). Simplification of the mesh would have eased the processing of the .obj file in MakerBot.

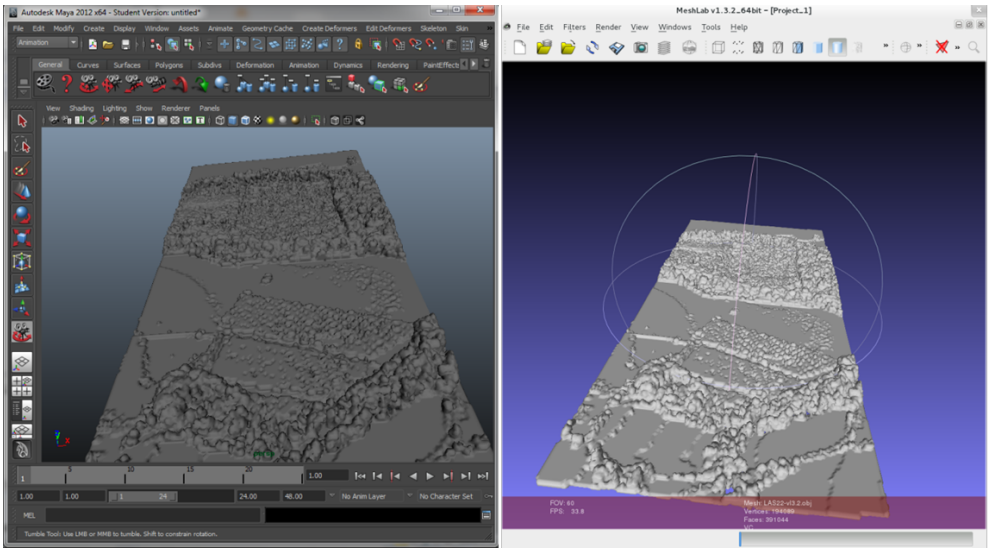


Figure 5-3: Visualising the output of DASOS into animation software packages (Maya and Meshlab)

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

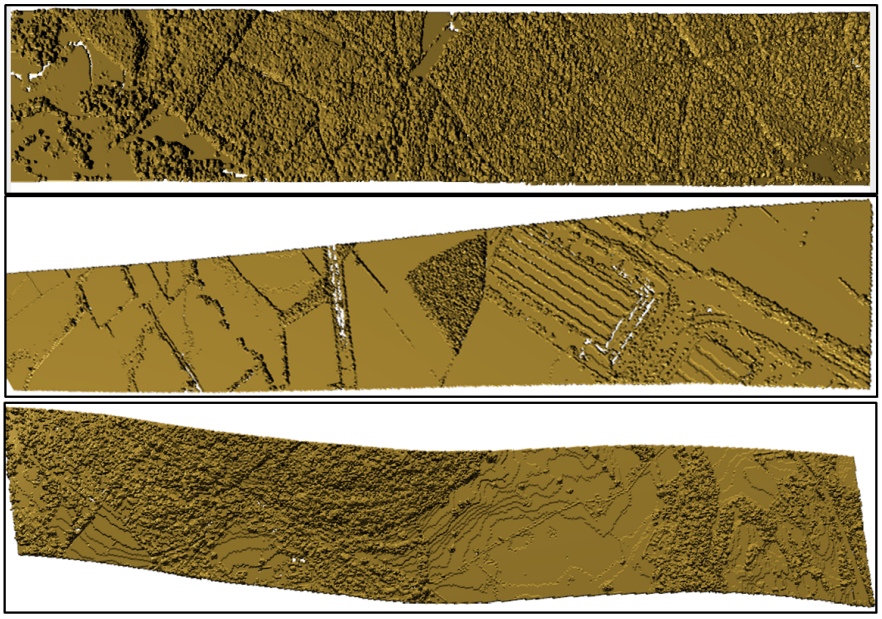


Figure 5-4: Polygonising NERC-ARF FW LiDAR data captured at different areas (New Forest, Milton Keynes and Eaves Wood)

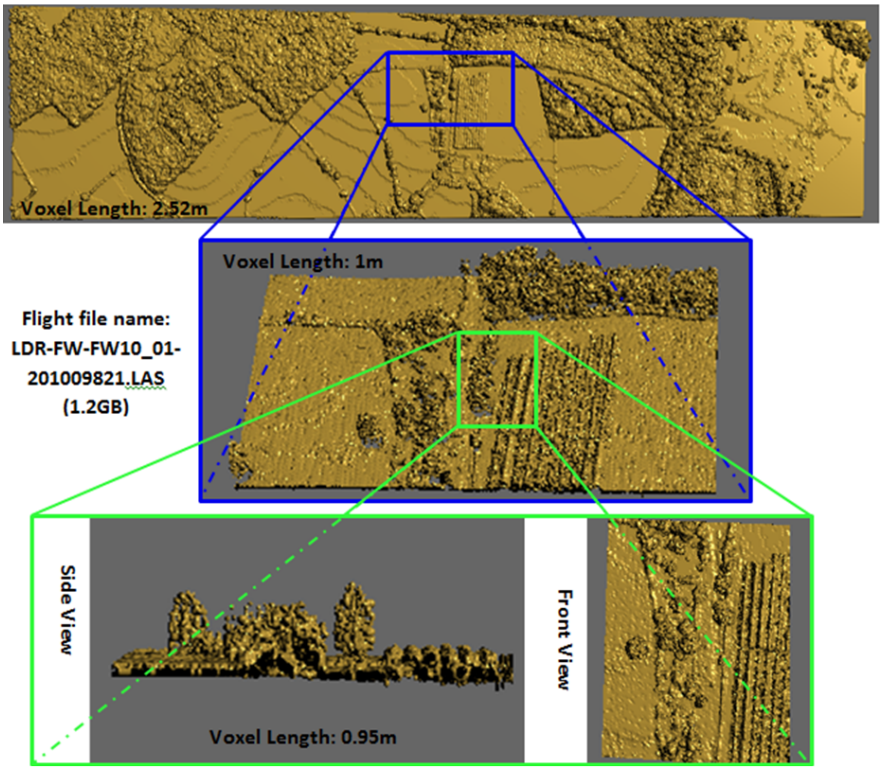


Figure 5-5: Selecting Region of Interest

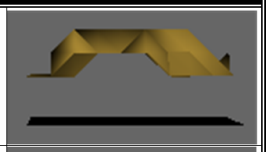
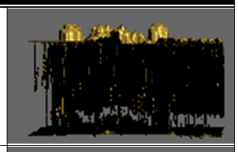
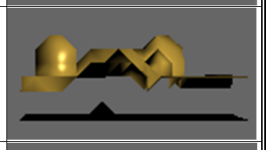
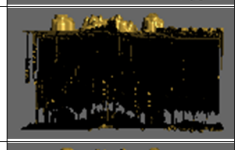
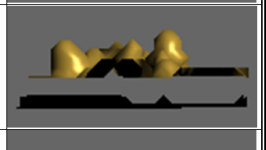
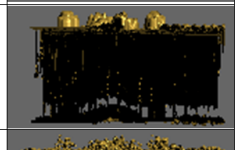
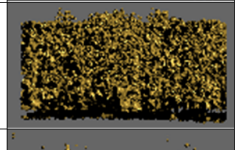
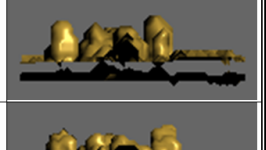
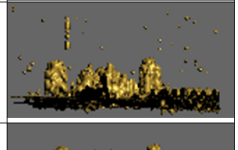
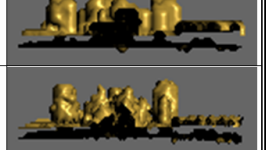
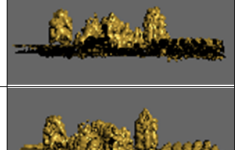
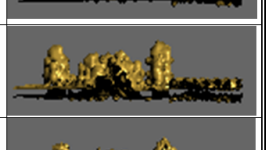
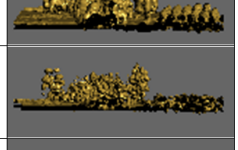
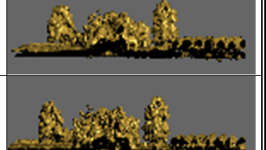
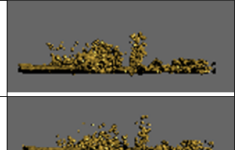
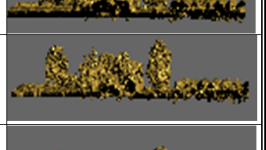
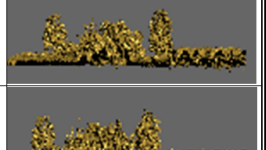
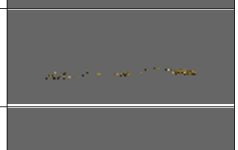
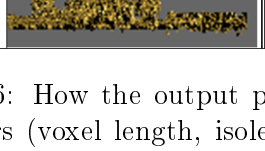
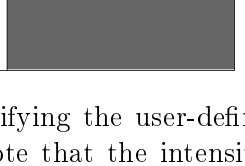
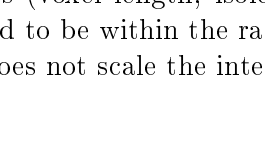
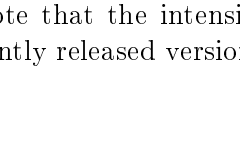
Voxel Length	Visualisation with different voxel lengths	Iso-level *	Visualisations with various isolevels	Noise Level	Visualisations with various noise levels
16.67 m		60		0	
10.0m		45		5	
7.14m		30		10	
5.7m		15		15	
4.44m		0		17	
3.33m		-15		20	
2.5m		-30		25	
2.0m		-45		30	
1.43m		-60		40	
1.2m		-75		60	
1.0m		-85		75	
0.8m		-95		100	
0.67m		-100		135	

Figure 5-6: 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.

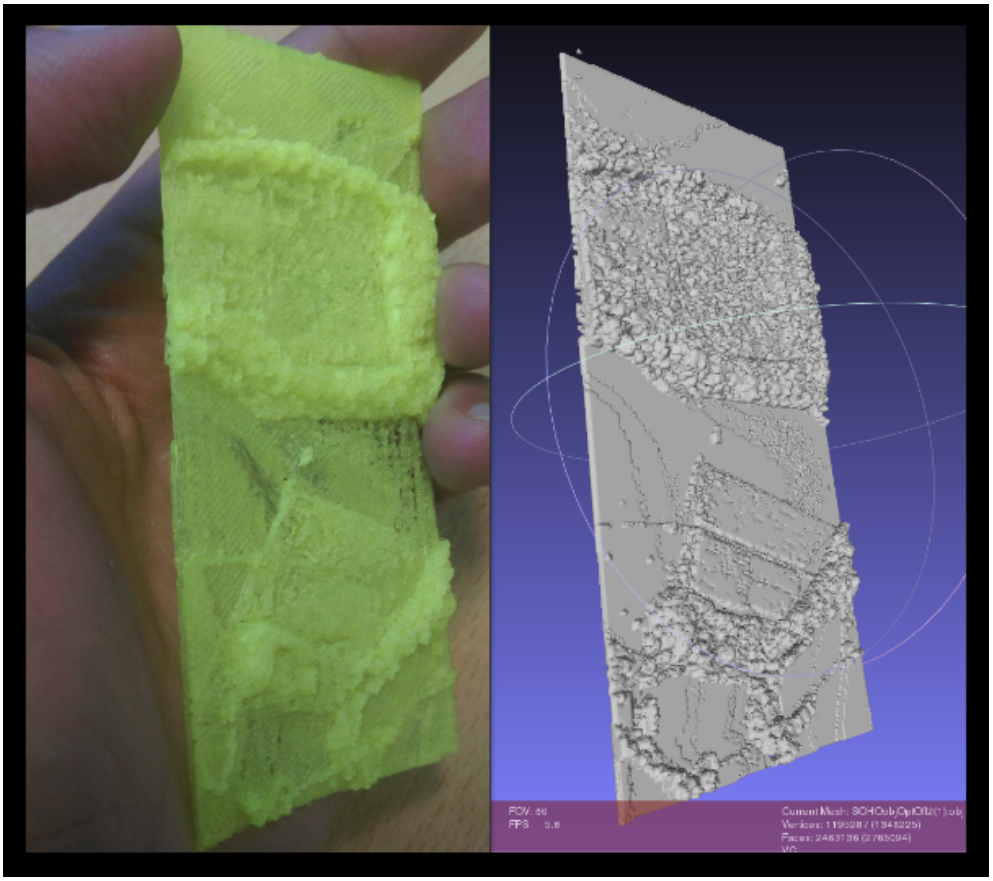


Figure 5-7: 3D printing of New Forest FW LiDAR data

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 4.1), 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] visualise either the peak extracted points or the raw waveform samples. On the one hand, SPDlib visulises the samples with intensity above a given threshold as points, while FullAnalyse generates a sphere per sample, with its radius directly correlated to the 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 4.2, 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 iso-surface extraction (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]. As the depth of the tree increases, the length of the parallelised instruction increases as well and therefore a good speed up is achieved. Nevertheless the function for the implicit representation of the FW LiDAR data at [35] executes in 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 reduced 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 resulting in 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-body simulation of particles [59]. Similarly, voxel hashing was proposed for reducing the overheads of 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]

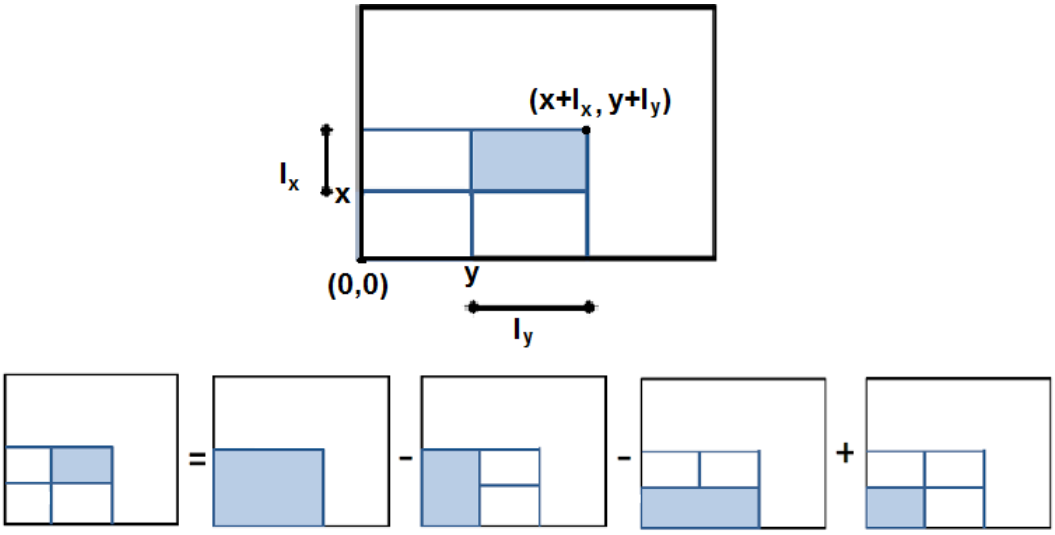


Figure 6-1: Once the Integral Image is constructed, the sum of any rectangular area is calculated in constant time.

In this thesis, we extend 'Integral Images' to 'Integral Volumes' and use them to quickly identify aion using depth cameras online [60]. Most of those data structure optimisations are based on GPU processing, but they are still very relevant.nd 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 in-

tegral 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 5-2a). If volumes with non-empty neighbouring voxels are ignored, then holes appear on the output polygon mesh.

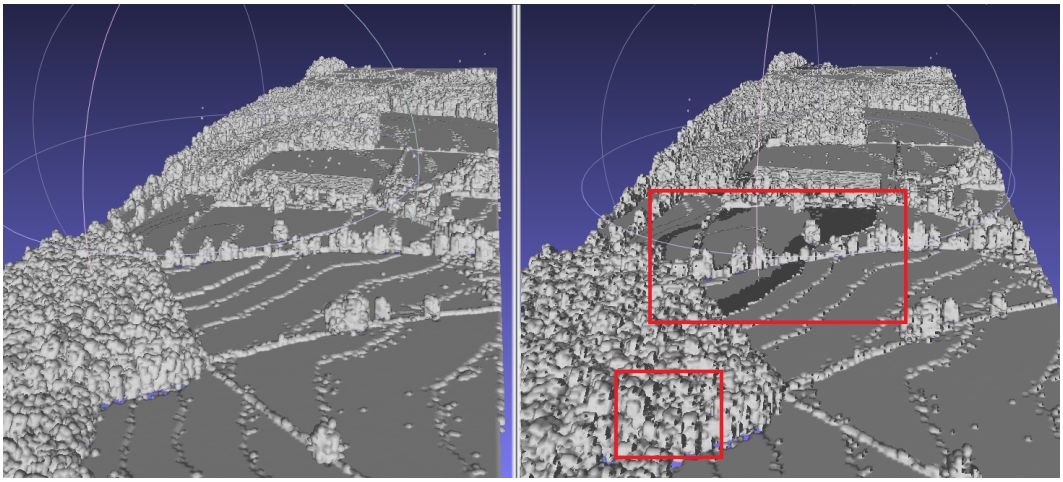


Figure 6-2: Comparison between including and ignoring neighbouring voxels; holes appears when ignored. Inside the red boxes, there are two affected areas.

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

```
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 D is equal to 0 then
7:         polygonise Cuboid-A
8:     else if (D bitwise add 2^s) shift right (s − 1) then
9:         divide side s of Cuboid-A
10:        if the new length of side s is equal to 1 then
11:            D ← D bitwise add (7 − 2^s)
12:        s ← (s + 1) mod 3
13:        push both new Cuboids into stack
14:    else
15:        s ← (s + 1) mod 3
16:        push Cuboid-A back into the stack
```

6.5 Octree Max and Min

'Integral Volumes' quickly identifies and ignores 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. As in the 'Integral volumes', it is essential to check neighbouring voxels when a branch of the 'Octree Max and Min' data structure could be ignored. However, 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:

- 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

Algorithm 3 Embedding the Marching Cubes Algorithm into an octree structure

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

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. In hierarchical data structures, the nearest common ancestor is tracked upwards and the branch, with its root as 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 octrees[2] is done in constant time. Nevertheless, linear octrees are full octrees. Therefore, if used in our application, 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 shortest 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 find neighbours from leaves and 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].

[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.

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 upward steps required to reach 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

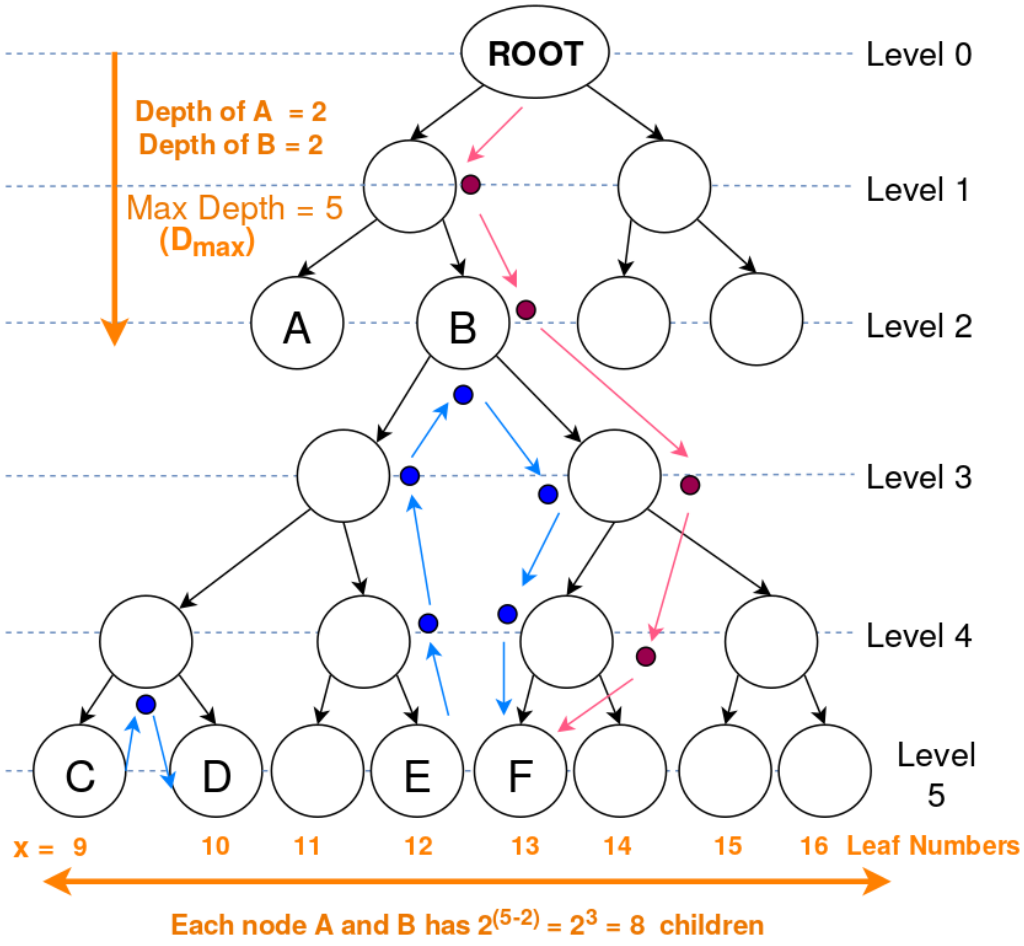


Figure 6-3: This diagram depicts the parameters used for finding neighbouring voxels.

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 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)

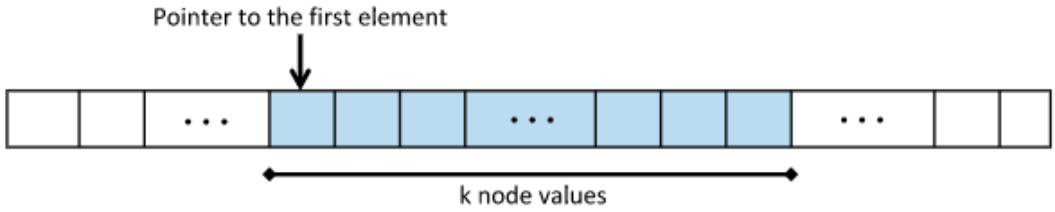


Figure 6-4: Ordering of tree elements

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 become integral using the following equation:

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

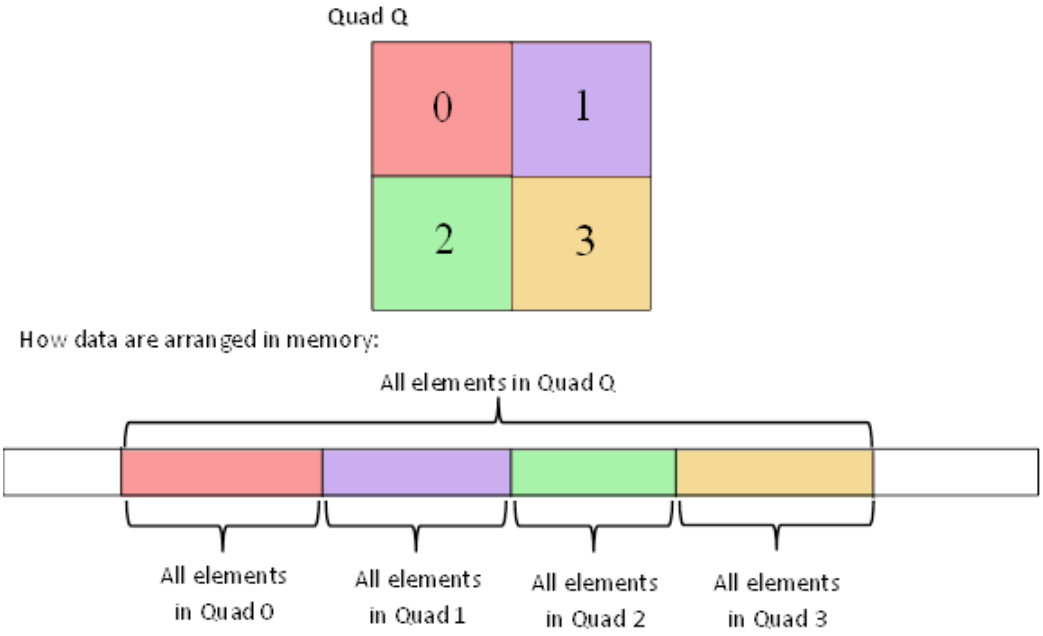


Figure 6-5: Illustration of how to save the values of an 'Integral Quad Tree' into the 1D-array, in order to preserve the condition of 'Integral Trees'

*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

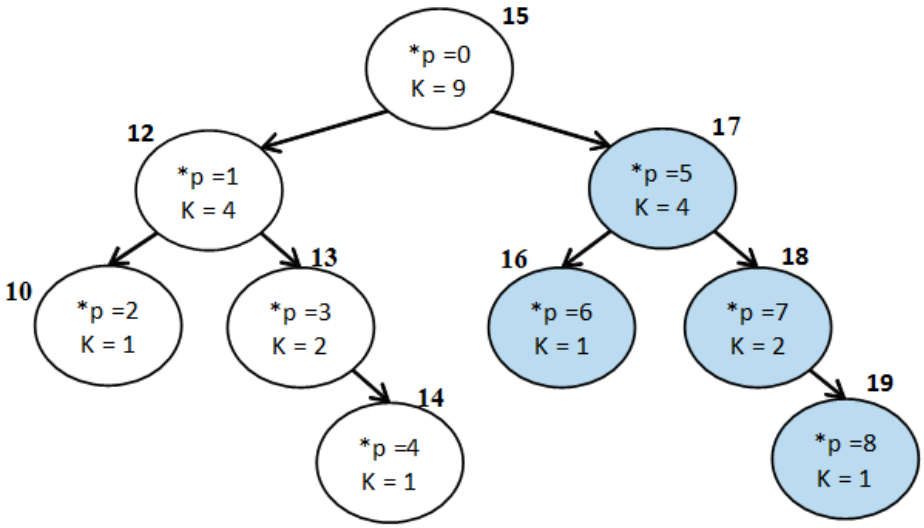


Figure 6-6: Example of 'Integral Binary Tree'

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 scanline 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.

	Scan-line algorithm: loops through all voxels (1)	Identifies and ignores empty areas during polygonisation (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

6.8 Results and Experiments

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:

[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.

- 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 have only one difference, with the rest of the parameters being held 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 first flightline is from the New Forest (LDR-FW-FW10_01-201009822.LAS), the second flightline is from the Dennys Wood (LDR-FW10_01-201018713.LAS) and the third one from Eaves Wood (LDR-FW-GB12_04-2014-083-13.LAS). The second group checks whether there are significant performance differences when the algorithms are applied on different flightlines.

Except for 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.

			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

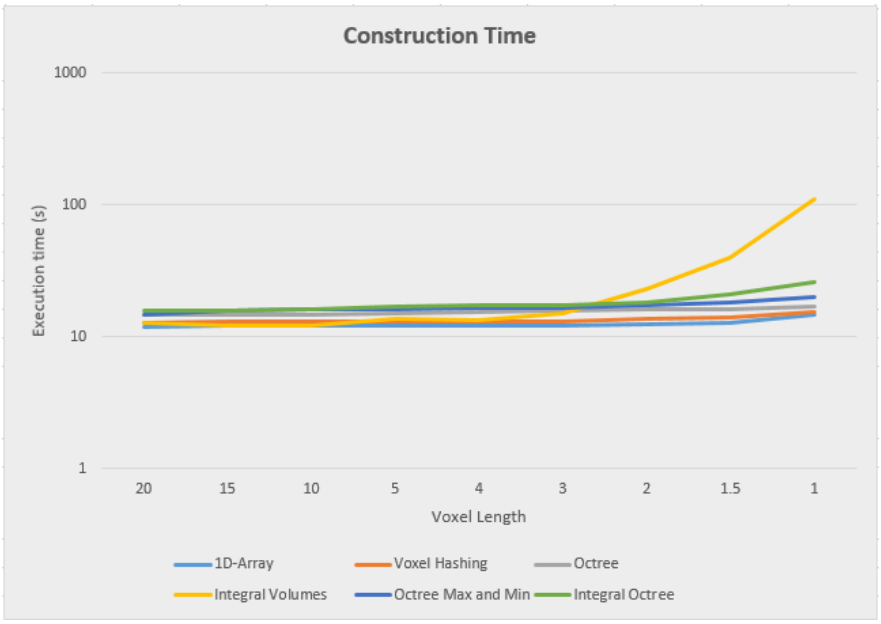


Figure 6-7: Time required to build each data structure by voxelising the FW LiDAR samples and inserting them inside the 3D volume (Table 6.7). Please note that the y-axis is in logarithmic scale.

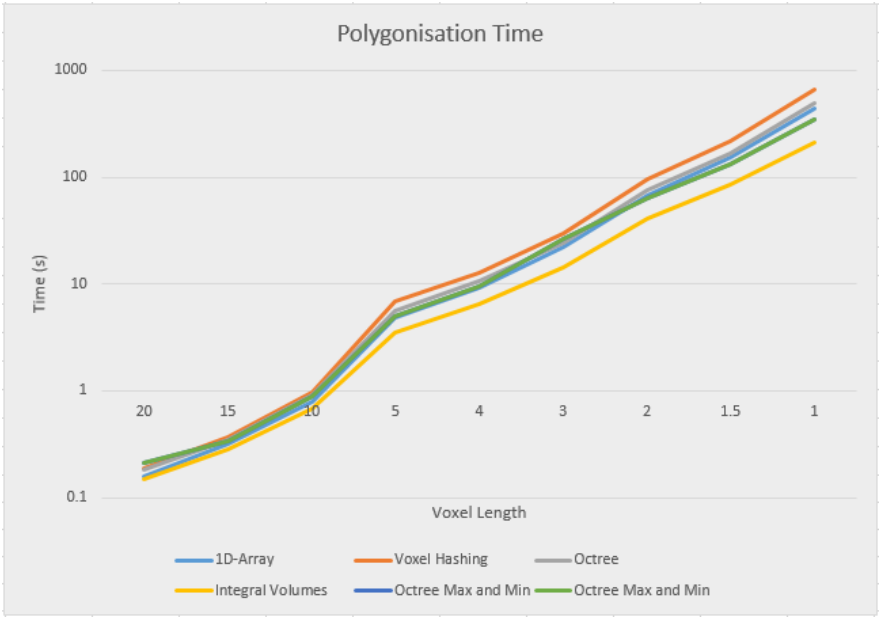


Figure 6-8: Time required to reconstruct the surface from the voxelised FW LiDAR data, after the data are voxelised (Table 6.7). Please note that the y-axis is in logarithmic scale.

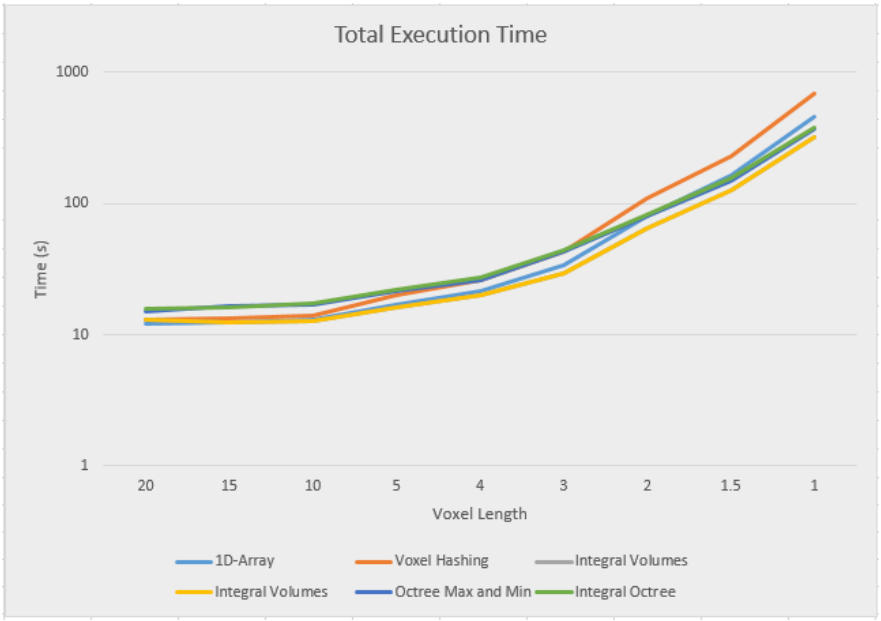


Figure 6-9: The sum of the time required to construct a data structure and the time required to generate a polygonal mesh (Table 6.7). The fastest one is the 'Integral Volumes'. Please note that the y-axis is in logarithmic scale.

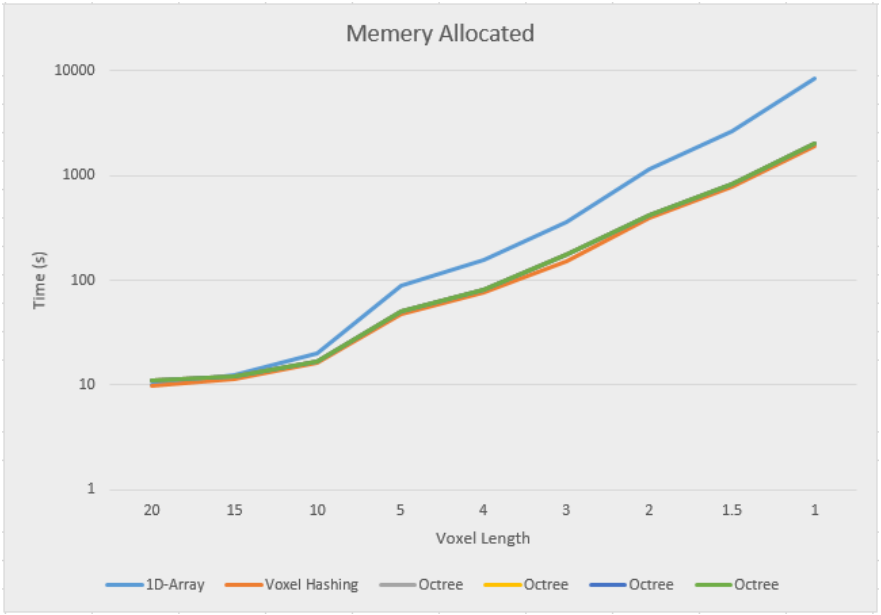
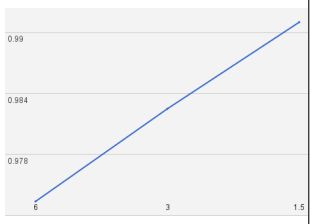
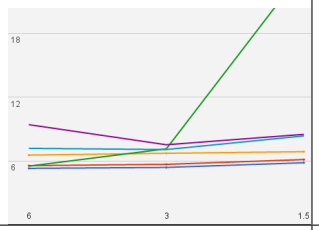
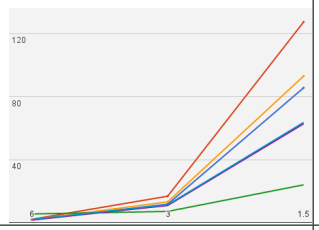
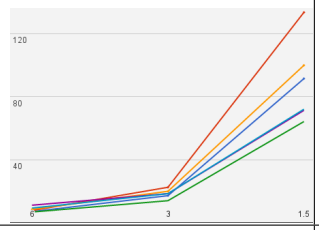
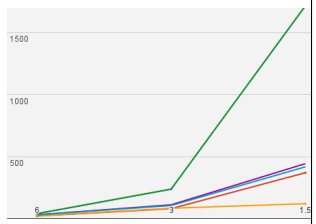
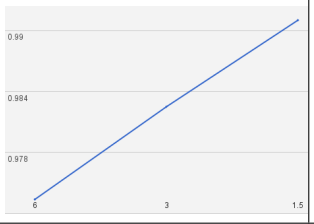
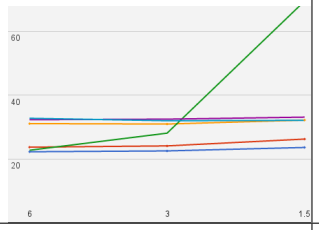
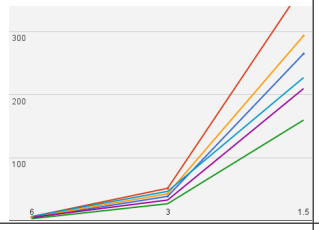
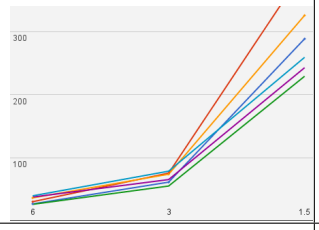
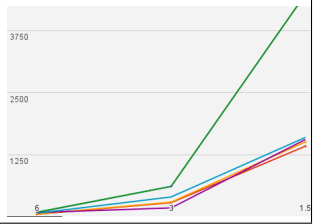
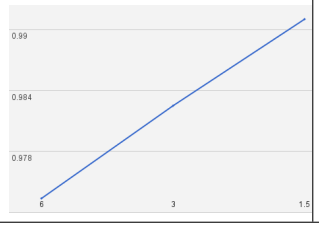
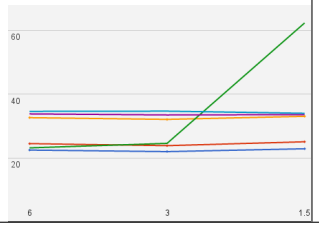
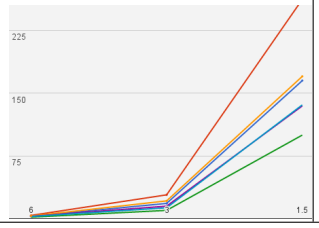
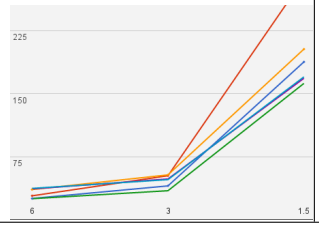
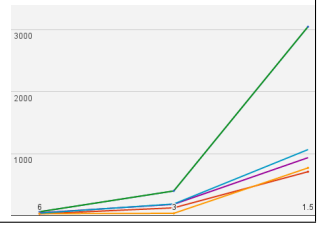


Figure 6-10: Maximum memory consumption at run time. '1D-Array' and 'Integral Volumes' consume the highest memory, which is approximately the same (Table 6.7). Please note that the y-axis is in logarithmic scale.

			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					

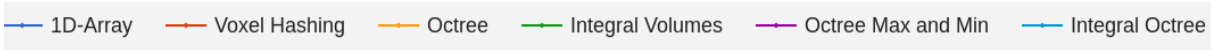


Table 6.9: Chart diagrams generated using the results from Table 6.8. The 1st flightline is from the New Forest Dataset (LDR-FW-FW10_01-201009822.LAS). The 2nd flightline is from the Dennys Wood dataset (LDR-FW10_01-201018713.LAS) and the 3rd one from Eaves Wood (LDR-FW-GB12_04-2014-083-13.LAS). Please note that the y-axis of this table are not in logarithmic scale.

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 require traversal for reaching elements, but it's very likely to have more memory jumps, considering that the implementation of the octree structures keeps the children of every branch coherent in memory for faster interpretation. Additionally, during the expansion of hashed tables, many reallocation occurs.

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 Octree' 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 Hashing' and octrees.

Chapter 7

Alignment with Hyperspectral Imagery

7.1 Introduction

In this chapter, the hyperspectral images are introduced to improve the visual output of the polygonal meshes derived from the FW LiDAR data (Chapter 5). The combination of NERC-ARF LiDAR and hyperspectal data from New Forest (Figure 2-1) for generating tree coverage maps is investigated.

Please note that definitions (i.e. bands and level-1) are used in this chapter and if you are not familiar with these terms it is highly recommended to read again Section 2.4, which explains everything about hyperspectral imagery.

7.2 Previous Work

Regarding the integration of FW LiDAR and hyperspectral data in remote forest surveying, there are diverse opinions on whether the integration of multi-sensor data improves remote forest surveying. Clark et al. attempted to estimate forest biomass but no better results were observed after the integration [66], while the outcomes of Anderson et al. for observing tree species abundances structures were improved after the integration of the data [67].

Buddenbaum et al. [68], and Heinzel and Koch [69], used a combination of multi-sensor data for tree classifications. Buddenbaum et al. use fusion of data to generate RGB images from a combination of FW LiDAR and hyperspectral features, although the fusion reduces the dimensionality of a classifier [68]. In that study, three different classifiers were implemented and the Support Vector Machines (SVMs) returned the

best results. SVMs were also used in [69] to handle the high dimensionality of the metrics (464 metrics). In that research a combination of FW LiDAR, discrete LiDAR, hyperspectral and colour infrared (CIR) images are used. Each of the 125 hyperspectral bands was directly used as a feature in the classifier, contributing to the high dimensionality.

7.3 Spatial Representation of Hyperspectral Pixels for Quick Search

For the New Forest Dataset (Figure 2-1), there are both FW LiDAR and hyperspectral data. The data are collected from two independent instruments and, while they are flown together, each instrument has a slightly different view point, a different resolution/sensing mechanism and data collection parameters (e.g. integration time). Therefore the data collected are not aligned to each other (e.g. for each waveform there is no trivial-correspondence to a spectral measurement). To integrate the data geo-spatially, alignment of the data is required. As mentioned at Section 2.4, in order to preserve the highest possible quality and reduce blurring that occurs during geo-rectification, data in the original sensing geometry (level-1) are used.

In Anderson et al. [67], an inverse distance weighted algorithm is used to rasterise the hyperspectral images and the pixel size is constant, 15.8m, while in this study an approach similar to Warren et al. [24] is used and the resolution is changeable. The main concept of our geo-rectification algorithm is to be able to find the nearest hyperspectral pixel to a point in the fastest possible way. For that reason, a spatial representation of the hyperspectral pixels is created by importing the pixels into a 2D grid, similar to [24]. The cell size of the grid in meters is constant, but the dimensions of the grid in number of squares (n_x, n_y) can be varied according to the chosen average number of pixels per square (A_{ps}):

$$n_x = \sqrt{\frac{n_s^2}{A_{ps}}} \qquad n_y = \sqrt{\frac{n_l^2}{A_{ps}}} \tag{7.1}$$

Where n_s is the number of samples and n_l is the number of lines of the hyperspectral cube (figure 2-4).

Furthermore, Warren et al. [24] uses a tree-like structure, here a structure similar to hash tables is used to spatially represent the pixels and speeding up searching. As shown in Figure 7-1, for each cell there is a bucket containing all the points that lie inside it. The hash function takes as input the unique key of the cell and returns the

memory location of the corresponding bucket. The key of a cell with coordinates (x_s, y_s) is is equal to $(x_s + y_s * n_x)$ where n_x is the number of pixels in the x-axis.

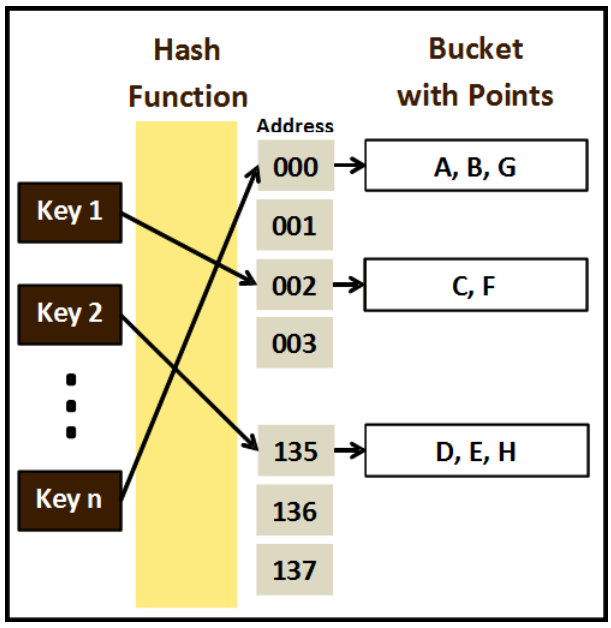


Figure 7-1: The hash table of the spatial representation of the hyperspectral pixels; each bucket contains all the pixels in a square and has a unique key derived from the coordinates of its square. The hash function takes as input the key and returns the address in memory of the corresponding bucket.

The next step is for a point (x_v, y_v, z_v) to find the pixel whose geolocation is the closest to it. First we project the point into 2D by dropping the z coordinate and then we find the square (x_s, y_s) that the projected point $\mathbf{v}(x_v, y_v)$ lies inside, as follow:

$$x_s = \frac{x_v - X_{min}}{X_{max} - X_{min}} * n_x \tag{7.2}$$

$$x_s = \frac{y_v - Y_{min}}{Y_{max} - Y_{min}} * n_y \tag{7.3}$$

Where X_{max}, X_{min}, Y_{max}, Y_{min} are the geospatial boundaries of all the hyperspectral image and n_x, n_x are the number of pixels in the x and y axis accordingly.

From the square (x_s, y_s) we can get the set of pixels that lie inside the same square with the point of our interest. Let's assume that the geospatial locations of these pixels are the vectors $\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}, ..., \mathbf{g_n}$ respectively. Then, by looping through that set of pixels, we can find the pixel i that is most likely to be the closest pixel to the point

$\mathbf{v}(x_v, y_v)$:

$$i = \arg\min |\mathbf{v} - \mathbf{g_i}|^2. \tag{7.4}$$

Finally, there is the case of the closest point to be within an adjacent square and this occurs when the point is very close to the edges of the square. Even though this was not implemented in DASOS when the paper [27] was published, it can be done by checking the distance between the edges of the square and the point. If this distance is smaller than the distance between pixel i then we can loop through the points of the corresponding adjacent square and check whether there is another pixel closer to point $\mathbf{v}$ than pixel i. Similarly, by checking the distance between the point $\mathbf{v}$ and the corners of the square (x_s, y_s), the case of the closest point to exist inside a diagonally-adjacent square is also covered.

7.4 Projecting hyperspectral images into polygon meshes generated using FW LiDAR data

This section focuses on projecting the (level-1) hyperspectral images onto the polygonal meshes reconstructed from the FW LiDAR data as explained in Section 5.4. As shown in Figure 7-2, the result is a coloured polygon mesh. That mesh is saved into two files:

- the .obj file that contains the 3D geometry and
- the .png file that contains the 2D texture image

The (level-1) hyperspectral images look deformed because the pixel size is not consistent. (Figure 7-2 shows that inconsistency). DASOS resolves this problem by adjusting the texture coordinates of the polygonal mesh according to the geolocation of the pixels. The texture coordinates (u, v) of each vertex lies inside the range $[0, 1]$ and if they are multiplied by the height/width of the texture, then the position of the corresponding pixel of the texture is given. In order to calculate the texture coordinates of each vertex (x_v, y_v, z_v), the spatial representation of the hyperspectral pixels (explained at Section 7.3) is used for quickly detecting the pixel (x_p, y_p), whose geolocation is the closest to a vertex. By dividing the pixel position with the number of samples (n_s) and lines (n_l) in the hyperspectral image, the texture coordinates (u, v) of each vertex $v(x_v, y_v)$ are calculated:

$$u = \frac{x_p}{n_s} \qquad v = \frac{y_p}{n_l} \tag{7.5}$$

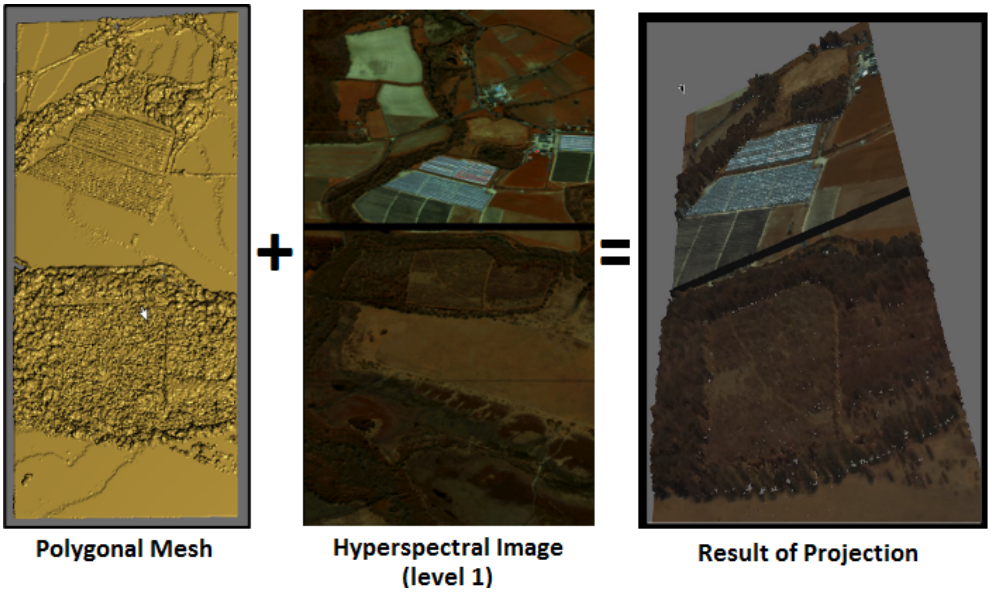


Figure 7-2: Projecting hyperspectral images into the polygonal meshes

Regarding the outputs, the texture coordinates of the polygonal mesh are added into the .obj file, while the 2D texture is simply an image generated from three user-selected bands for the RGB colours. The width of the image is equal to the number of hyperspectral samples per line while its height is equal to the number of lines.

7.4.1 Results

The results of the projection are coloured polygonal meshes. Each coloured polygonal mesh is exported into two files:

1. the .obj file contains the 3D geometry with all the information about the vertices, edges, faces, normals and texture coordinates, and

2. the .png is the 2D texture (an RGB image) and it is aligned with the texture coordinates of the polygonal mesh.

Figure 7-3 shows how the visual output is affected by projecting hyperspectral data from different sensors and by changing the selected bands.

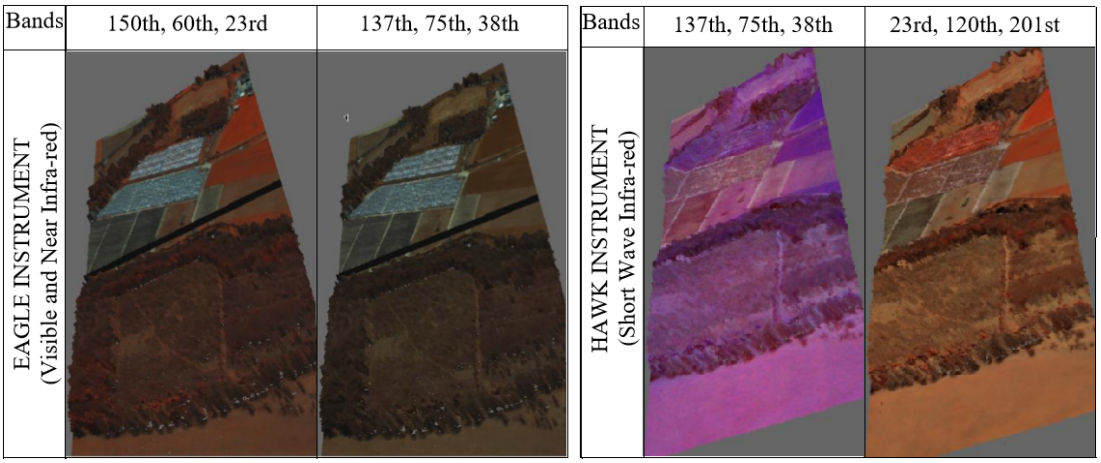


Figure 7-3: Results of Alignment; the left table shows the results of projecting hyperspectral images from the Eagle instrument onto the polygonal meshes generated using FW LiDAR data and the right hand side table shows results using the Hawk instrument.

7.5 Tree Coverage Maps

As mentioned before, there are diverse opinions on whether or not integration of remotely sensed data improves forest monitoring [66] [67]. For that reason, a simple pixelwise classifier was implemented to test how the integration of NERC-ARF data, using metrics generated from DASOS, performs for generating tree coverage maps.

The metrics generated from both hyperspectral and FW LiDAR data are 2D aligned images (Table 4.3). In other words, the pixel (x, y) has the same geospatial coordinates in every metric. Further the resolution of the metrics depends on the resolution of the 3D voxelised FW LiDAR data (Section 4.2). If the dimensions of the volume are (x, y, z) then the dimensions of the metrics are (x, y). For the LiDAR metrics, each pixel is coloured according to the information derived from the corresponding column. Regarding the hyperspectral metrics, (level-1) data are used to preserve the highest possible quality. The method in Section 7.3 was used for finding the pixel from the hyperspectral data with the closest geospatial location to the centre of each column of the 3D voxelised FW LiDAR.

The metrics used for generating tree coverage maps are grouped into two categories (FW LiDAR and hyperspectral metrics):

- FW LiDAR: Height (L0), Thickness (L1), Density (L2) and First Patch (L3)
- Hyperspectral: Mean (H0), NDVI (H1), Standard Deviation (H2) and Spectral Signature (H3)

For more descriptive information and examples of the metrics please look at Table 4.3,

where all the functionalities of DASOS are listed.

7.5.1 Testing and Results

In this case, the total accuracy was increased with the integration of FW LiDAR data and hyperspectral images. A Naïve Bayesian classifier using a multi-variance Gaussian model is applied for distinguishing tree-covered areas from the ground. The main idea is for each pixel/column to find the class that is more likely to belong to Tree or Ground. Ground truth data were hand painted using 3D models generated with DASOS and were divided into training and testing data. There are three test cases and, for each test case, the following metrics are used:

- 1st test case uses the L0-L3 metrics that are generated from the FW LiDAR data.
- 2nd test case uses the H0-H3 metrics that are generated from the hyperspectral imagery.
- 3rd test case uses L0-L3 & H0-H3 which is a combination of metrics generated from either FW LiDAR data or hyperspectral imagery.

For each test case, an error matrix is generated to indicate the accuracy of the classification results as verified against the ground truth data (Tables 7.1, 7.2 and 7.3) [70]. Each row shows the number of pixels assigned to each class relative to their actual class. For example, the first row of Table 7.1 shows that 130445 pixels were classified as trees, where 125375 were actual trees and the rest 5070 were ground. From the error matrices, the classification accuracy of each test case was calculated and it is presented in Table 7.4.

Figure 7-4 depicts the coverage maps generated for each test case. Three areas are marked for comparison. In Area 1 there is low vegetation, in Area 2 there are short trees and in Area 3 warehouses. Area 1 was incorrectly classified when only the hyperspectral data were used; when the height information of the LiDAR data was included into the classifier, area 1 was correctly classified. Similarly, Area 2 was wrongly classified when using the only FW LiDAR metrics because the height of the trees was less than the average training samples. But since features from the hyperspectral data are not height dependant, the classification results of test case 1 (with hyperspectral metrics) was better at Area 2. Area 3 seems to confuse the first two classifiers in different ways, but the combination improved the results.

Finally, to demonstrate the usefulness of DASOS's polygonal meshes, the results of the tree coverage maps were projected into the polygon representations as shown in the following Figure 7-5.

		Ground truth data		
		Tree	Ground	Row Total
Results	Tree	125375	5070	130445
	Ground	45093	228495	273588
	Total	170468	233565	404033

Table 7.1: Error Matrix showing the pixel-wise classification results of the 1st test case that only uses hyperspectral data.

		Ground truth data		
		Tree	Ground	Row Total
Results	Tree	154768	39504	194272
	Ground	15700	194061	209761
	Total	170468	233565	404033

Table 7.2: Error Matrix showing the pixel-wise classification results of the 2nd test case that only uses FW LiDAR data.

		Ground truth data		
		Tree	Ground	Row Total
Results	Tree	152597	10548	163145
	Ground	17871	223017	240888
	Total	170468	233565	404033

Table 7.3: Error Matrix showing the pixel-wise classification results of the 3rd test case that uses both hyperspectral and FW LiDAR data.

	FW LiDAR	**Hyperspectral Imagery**	**Both**
Tree	73.55%	90.79%	89.52%
Ground	97.83%	83.09%	95.48%
Mean Average Precision	87.58%	86.34%	92.97%

Table 7.4: Classification accuracy of each test case

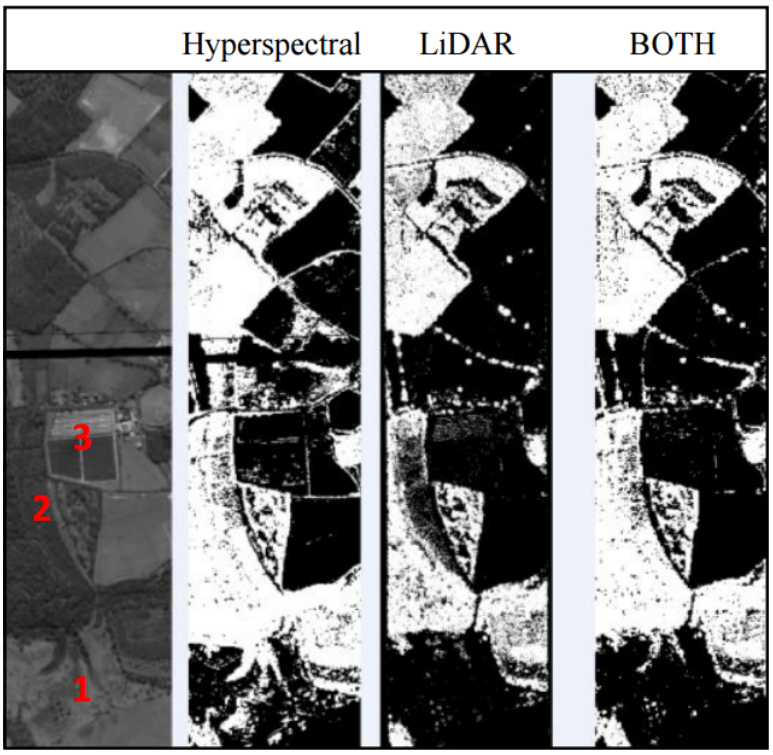


Figure 7-4: Visual Comparison of the results of the coverage maps

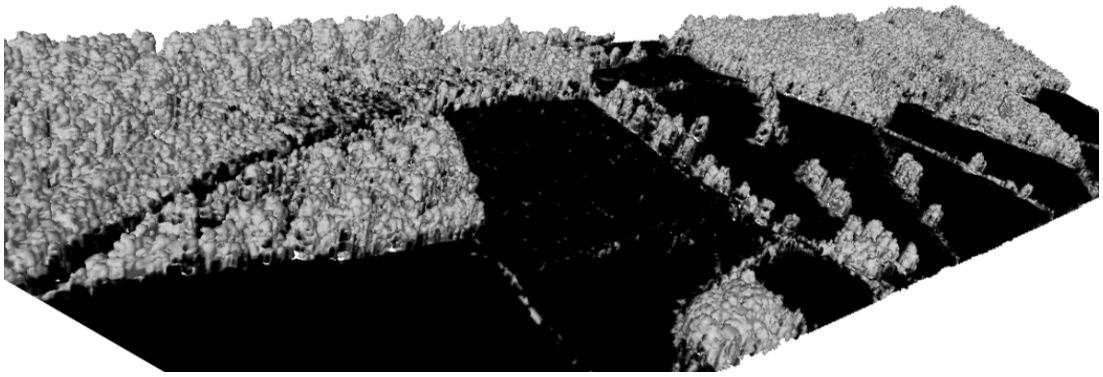


Figure 7-5: 3D Coverage model, generated by projecting the results of the tree coverage classification into a polygonal mesh.

7.6 Summary and Conclusions

In conclusion, this chapter describes an efficient way of aligning the FW LiDAR data and hyperspectral images using a spatial representation of hyperspectral pixels. The voxelisation of the FW LiDAR data also eases the generation of aligned metrics from both datasets. Furthermore, the resolution of the metrics is changeable and depends on the user-defined resolution of the voxelised FW LiDAR data. Additionally, since the closest pixel is always selected, regardless of the distance from the point of interest, the problem of having data at different resolution is automatically resolved.

Regarding the results, coloured polygonal meshes were generated using the alignment and the result demonstrated that the integration of this specific data has potential in remote forest surveying - aside from improving the visual appeal, it also improves automatic classification. This was shown using a simple classifier for generating tree coverage maps. The results were positive; the classification accuracy was improved by 5.39% when both datasets were used. A more sophisticated classifier would likely give even better results.

Chapter 8

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

8.1 Introduction

8.1.1 The Importance of Dead Wood

The value of dead trees from a biodiversity management perspective is large. Once a tree dies, its contribution to our ecosystem continues. The woody structure remains for centuries and it contributes to forest regeneration while providing resources for numerous surrounding organisms [71]. As an indication, more than 4000 species inhabit dead wood in Finland [72], where an estimate of 1000 species has been extinct [73]. These species do not only include animals and birds but also organisms, like fungi. Fungi contributes to wood decaying, formation of hollows and biodiversity, which is an important factor for a resilient ecosystem [74]. Observing the changes of fungal diversity on decaying wood has an increased interest in science [75] [76] [77] in order to ensure the continuous existence of decaying wood in forests.

In Australia, tree hollows play a significant role in managing biodiversity. Nearly all arboreal mammals rely on hollows with the exception of the Koala and perhaps Ringtail Possums that preferentially make a stick nest, but they use hollows as well. Additionally, a large number of Australian bird species rely on hollows for shelters [5]. Nevertheless, Australia has no real hollow creators unlike the northern hemisphere

(e.g. Woodpeckers), and therefore it relies predominantly on natural processes of limb breakage, insect and fungal attack when access points are provided through damage caused by wind, storms and fire.

This kind of hollows take hundreds of years to form and because of that it is more likely to exist on dead trees. In Australia, studies predict shortage of hollows for colonisation in the near future [3] [4]. Therefore automated detection of them plays a significant role in protecting those animals. As an indicator of the importance of hollows in managing biodiversity, a list of a few of the species that rely on hollows was provided by the Forestry Corporation of NSW. Those species are shown at Figure 8-1. According to the Department of the Environment of Australian Government and the Government of Western Australia, six of them are protected, threatened or close to extinct [78] [79]. Figure 8-1 shows the species from the provided list and the six protected species have a red border and their names are bold in the description.

For the aforementioned reasons, monitoring dead trees is essential for having a resilient ecosystem. Nevertheless, the distribution of dead trees significantly varies making detection of them difficult [80]. Remote sensing approaches has been introduce to automate the process of monitoring forest and further increase the spatial resolution of the monitored area. The following section gives an overview of the related work undertaken in Remote Sensing.

8.1.2 Related Work

Remote Sensing was introduced for automatically detecting dead trees, because field-work is time consuming considering their variance spread and the size of the relevant forests. From a classification perceptive, the task of identifying dead standing and dead fallen trees is different. Fallen trees are identified by detecting segments or line-like features on the terrain surface using LiDAR data [81] [82]. Regarding standing dead trees, their shape (reduced number of leaves or broken branches) [83] and light reflectance (less green light illuminated) [84] are important factors for identifying them.

Previous work on dead standing trees detection performs single tree crown delineation before health assessment [83] [85]. Tree-crown delineation is usually done by detecting local maxima from the canopy height model (CHM) and then segmenting trees with watershed algorithm [86]. Improvements has been achieved by introducing markers controlled watershed [87] and structural elements of tree crowns with different sizes [88]. Additionally, Popescu and Zhao analyse the vertical distribution of the LiDAR points in conjunction with the local maximum filtering of CHM [89].

In the case of Eucalyptus, single tree detection is a challenge on its own, due to their irregular structure and multiple trunk splits. In other words, each tree trunks splits

Figure 8-1: A number of species that rely on tree hollows of which the red ones / bold ones are close to extinction: Kookaburra, Sulphur Crested Cockatoo, **Corella**, Crimson Rosella, Eastern Rosella, Galah, Rainbow Lorikeet, Musk Lorikeet, Little Lorikeet , Red-winged Parrot, **Superb Parrot**, Cockatiel, Australian Ringneck (Parrot), Red-rumped Parrot, Powerful Owl, Sooty Ow, Barking Owl, **Masked Owl**, **Barn Owl**, White-throated Treecreeper, Hollow Owl, **Brush-tailed Possum** (mammal) [1]

[1]The images of the birds were taken from the following links (Retrieved on the 27th of April 2016): Kookaburra: <http://tenrandomfacts.com/blue-winged-kookaburra/>, Sulphur Crested Cockatoo: <http://aussiegal7.deviantart.com/art/Sulphur-Crested-Cockatoo-08-153341893>, Corella: <http://www.theparrotplace.co.nz/all-about-parrots/long-billed-corella/, Superb Parrot: <http://www.davidkphotography.com/?showimage=637>, Crimson Rosella: <http://25.media.tumblr.com/tumblr_m3mo89c40r1r4t9h1o1_1280.jpg>, Eastern Rosella: <http://2.bp.blogspot.com/-pYxw51WjSOY/UB-LEFgd2KI/AAAAAAAAAWg/9z60PUWE6TE/s1600/_GJS6601-as-Smart-Object-1.jpg>, Rainbow Lorikeet: <https://www.reddit.com/r/pics/comments/328fvc/a_rainbow_lorikeet_found_in_coastal_regions/>, Musk Lorikeet: <http://www.rymich.com/girraween/photos/animals/birds/medium/glossopsitta_concinna/glossopsitta_concinna_001.jpg>, Little Lorikeet: <http://www.pbase.com/sjmurray/psittacidae>, Red-winged Parrot: <https://www.pinterest.com/pin/395894623469889727/>, Cockatiel: <http://up.parsipet.ir/uploads/Cockatiels-for-sale.jpg>, Australian Ringneck (Parrot): <http://ontheroadmagazine.com.au/wp-content/uploads/2015/09/Twenty-eight-parrot-2-min.jpg>, Red-rumped Parrot: <http://parrotfacts.net/wp-content/uploads/Red-Rumped-Parrot-on-a-tree.jpg>, Powerful Owl: <http://farm1.staticflickr.com/219/495796536_f78dac04c1.jpg>, Sooty Owl: <ttp://www.mariewinn.com/marieblog/uploaded_images/screech2-738532.jpg, Barking Owl: <http://www.pcpimages.com/Nature-and-Wildlife/Birds/i-7JKSTp5/1/L/owl%20%281%20of%201%29-L.jpg>, Masked Owl: <http://www.survival.org.au/images/birds/masked_owl_2_600.jpg>, Galah: https://www.pinterest.com/pin/537546905498955709/>, White-throated Treecreeper: <https://geoffpark.files.wordpress.com/2011/09/female-white-throated-treecreeper.jpg>,

create a local maximum leading into over-segmentation when tree crowns are detected by local maxima filtering. Shendryk published a eucalyptus delineation algorithm that starts segmentation from bottom to top. In this paper, the trunks point cloud is separated from the leaves and individual trunks are identified before proceeding to crown segmentation [90]. Nevertheless, for that project only 17 flightlines of LiDAR data were collected. The density resolution starts from 12 points/m^2 and goes up to 36 points/m^2 around forested areas. For small research projects capturing this high resolution is acceptable, but for commercial use and larger areas, the density of data collected is above the optimal resolution for a cost effective versus quality acquisition [91]. The project of this thesis is much larger. The resolution of our acquired LiDAR data has an average of four pulses per square meter,which is considered an optimal resolution in relation to the cost. But because of the tree height (up to 43m according to the fieldwork), a small amount of pulse intensity reached the trunks and the recordered waveform do not include enough information for individual trunk detection. An example of this project's discrete LiDAR data is shown in Figure 8-2 and the missing information about the trunks is depicted.

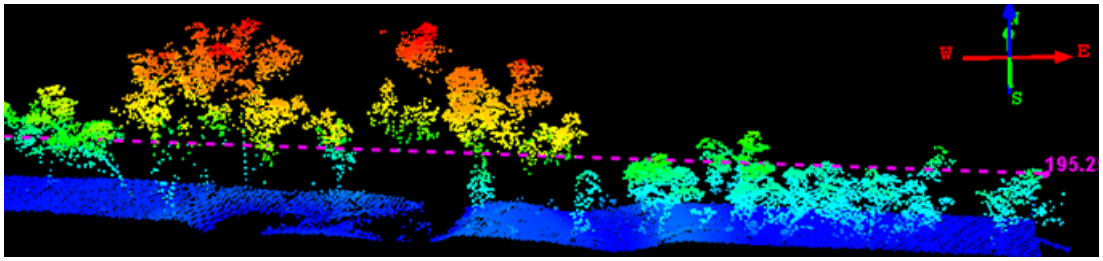


Figure 8-2: LiDAR point cloud showing that there are very limited points reflected from tree trunks.

The acquired data are full-waveform LiDAR data. Traditional ways of interpreting FW LiDAR data, suggests extraction of a denser points cloud using Gaussian decomposition [29] [30]. Nevertheless, in this project we uses the open source software DASOS. DASOS was influenced by Persson et al [26], who used voxelisation to visualise the waveforms . But, it does not only uses voxelisation for visualisations but also for extracting metrics useful in classification. It further normalises the intensities so that equal pulse length exists inside each voxel, making intensities more meaningful. It is further seems that the literature is moving towards voxelisation with promising results obtained at recent publication on tree species classification [33].

Here, it is introduced an approach for quick dead tree detection derived from the boost cascade approach [92] but extended into 3D. This approach further contains

Hollow Owl: <http://www.mariewinn.com/marieblog/uploaded_images/screech2-738532.jpg>

similarities of the 3D tree shape signatures proposed by Dong, 2009, for distinguishing Oaks from Douglas fir tree crowns [93].

8.2 Materials

8.2.1 Study Area

The study area (Figure 8-3) is a native River Red Gum (Eucalyptus camaldulensis) forest of size $542km^2$ in south-eastern Australia. The regeneration of the eucalyptus is extremely dependant in floods and therefore, their distribution in respect to density, health and age is highly variance [94]. Additionally, the height of Eucalyptus camaldulensis reaches up to $30 - 40m$ and their structural complexity is high with multiple trunk splits [95]. The size and structure of the forest, with a human as reference, is depicted in Figure 8-4, while examples of the variance shape of dead trees is shown in Figure 8-5.

8.2.2 Acquired full-waveform LiDAR data

Multiple-echo, full-waveform (FW) LiDAR data are supplied by RPS Australia East Pty Ltd. The data were acquired from 900m above ground level, using the Trimble AX60 Airborne LiDAR sensor, which was released in October 2013 [96]. The wavelength of the emitted laser was 1062nm, the maximum scan angle was 60 degrees, and the pulse rate was 400kHz. The acquisition was held from the 6th of March till the 31st of March 2015. The collected LiDAR were delivered into 206 flightlines, of which 13 are cross runs used for geometric correction. There is also a 30% of swath overlap. The point spacing along and across the track is 0.48m and the average point spacing is 4.3 points per square meter. Figure 8-6 shows an example of a dead tree in respect to the acquired discrete LiDAR point cloud. Detailed information about FW LiDAR related concepts are given in section 2.

8.2.3 Field Data

The field data were collected in July 2015 during the winter season of Australia and they include tree and canopy related measurements on circular plots. There are 33 plots with radius 35.68m and area 0.4ha allocated randomly inside the study area. On these plots, a total of 2386 trees were individually measured. Tree measurements include the geo-location, the trunk diameter at the standard height of 1.3m (breast heigh), height, species and health conditions (i.e. dead or alive). The geo-location of each tree is defined by the magnetic bearing from the centroid of the plot in degrees (range $[1, 360]$) and

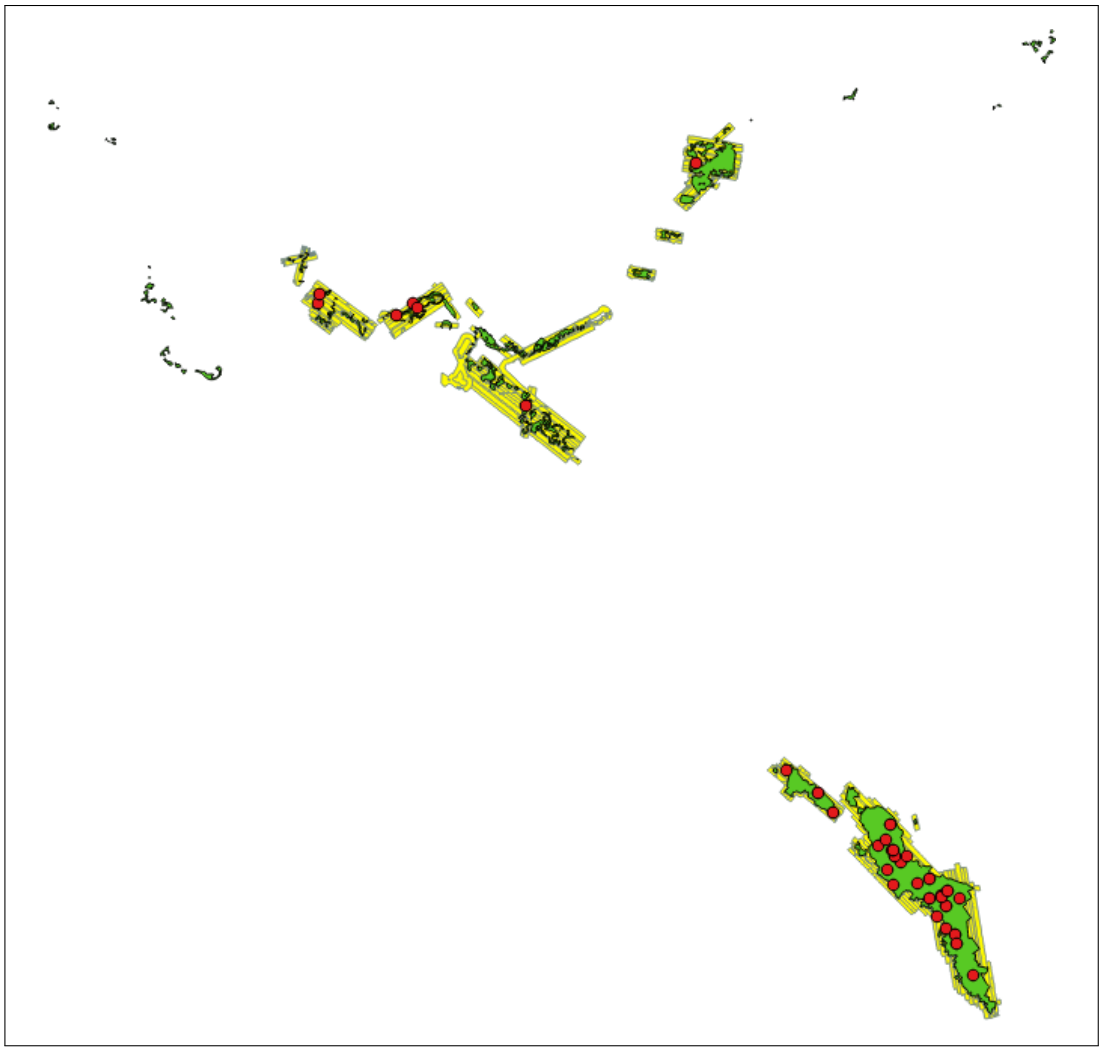


Figure 8-3: The study area is depicted by green ($542km^2$), the yellow strips are the LiDAR flightlines and the red dots are the position of the field plots.

the distance from the centroid in meters. The northing and easting coordinates of the geo-location of each tree were calculated in post-processing. Here is worth mentioning that a single tree may be recorded as multiple trees if there is a trunk split bellow the breast height of 1.3m. Furthermore, 91.59% are River Red Gum and the rest are Black Box (Eucalyptus largiflorens) and Wattle group (Acacia spp.).

Inside the field data, there are 260 dead trees recorded. Nevertheless, not all of those trees are considered useful for biodiversity. Dead trees with big Diameter at Breast Height (DBH) are more likely to contain hollows. Additionally, trees with DBH smaller than the footprint spacing of the LiDAR data are not identifiable from the FW

Figure 8-4: Structure of Red Gum Forest in south-eastern Australia.

Figure 8-5: Example of dead trees indicating their variance in shape.

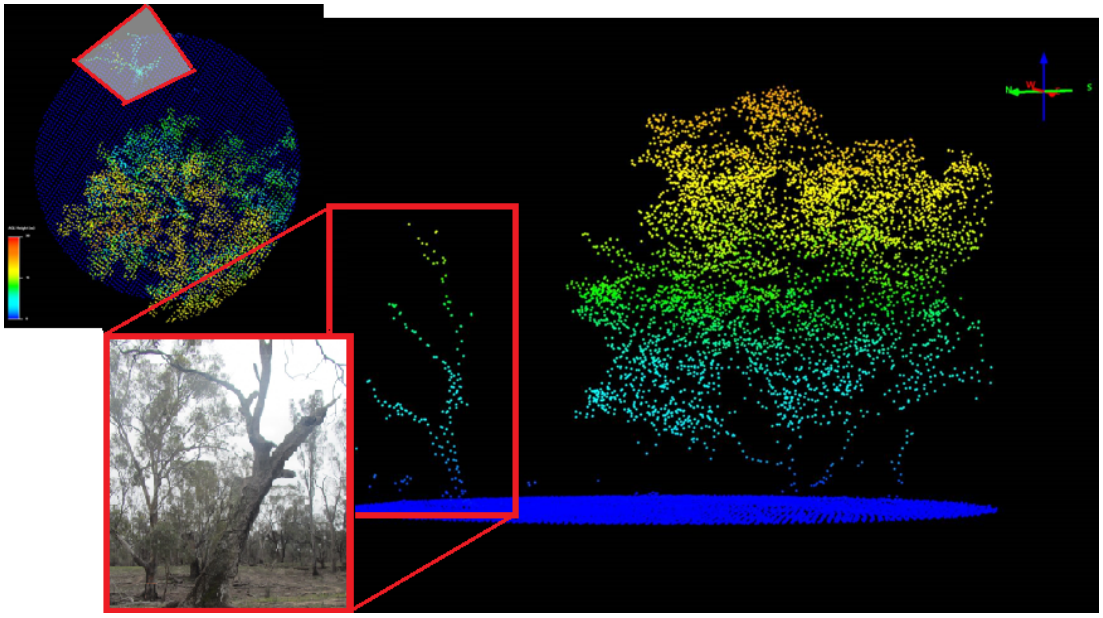


Figure 8-6: Example of a dead tree in relation to the discrete LiDAR point cloud.

LiDAR data. Table **??** shows the number of dead and alive trees in respect to their DBH.

DBH (cm)	**Dead Trees**	**Alive Trees**
>2000	0	1
1000-2000	7	21
600-1000	8	146
400-600	26	290
300-400	32	286
200-300	50	462
100-200	125	904
<100	11	16
Total	260	2126

Table 8.1: Number of trees according to their DBH

Please note that the aforementioned field data were provided by Forestry Corporation of NSW, Wauchope, Australia and Interpine Ltd Group, New Zealand. For this thesis, a case study for collecting field data was conducted in New Forest, UK. This helped to better understand classification challenges in forestry applications. More information about this study is provided in Appendix B.

8.3 Classification Challenges

This section focuses on the challenges faced while working on the detection of dead standing eucalyptuses. Table 8.2 underlines these challenges, categorised into three groups: the nature of the study area, the acquired data and the field data. All these challenges influence the quality of the classifier and the accuracy of the results.

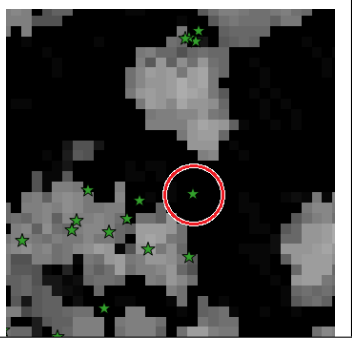
Study Area	**Acquired Data**	**Field Data**
• The study area is a native eucalyptus forest. Native forests contain trees of different ages and heights. The height of a dead tree could be within the range of [1.5,40] meters. • There is a high variance in the density of the forest. Sometimes the testing/training priors of the small dead trees may contain information from either nearby alive trees or ground. • A tree may have dead branches but still be alive. • Eucalyptus trees have irregular shapes and multiple trunk splits making tree delineation to require very dense acquired data.	• The pulse density of the acquired data does not allow bottom to top tree delineation. Crown detection from DEM (top) leads to over-segmentation due to the multiple trunk-splits. We, therefore, investigate the performance of object detection algorithms that do not require tree delineation. • An important factor of identifying dead trees is the light reflectance, but for this project this kind of data (i.e. coloured imagery) was not acquired. Therefore, the classifier is only trained on tree shapes. But the shape of the tree is not an independent factor of identifying dead trees, since a tree may not have leaves but still be alive.	• If a tree has a trunk split below the 1.3m height, then it is recorded as multiple trees within the field data. This results into an inconsistency of the "one tree" concept. • They contain small trees, which are non detectable from the acquired data. • The accuracy of the geo-spatial positions is unknown. Even though it is claimed to be within centimetres, there are trees clearing appearing on the ground, once visualised on top of the DEM. An example: 

Table 8.2: The Classification challenges of automated detection of dead eucalyptuses

8.4 Methods and Algorithms

This section provides an explanation of the algorithms implemented. An overview of the work flow is given here:

1. Subtraction of the Digital Terrain Model (DTM) from the FW LiDAR data
2. Generation of training feature vectors characterising dead and alive trees, as well as testing samples of unknown population
3. Identification of the most important relevant features using random forest
4. Generation of a probabilistic field using a weighted k-nearest neighbour (KNN) algorithm.
5. Filtering
6. Height histogram and ground pixels removal
7. Thresholding dead pixels from alive, filtering, applying a seed growth algorithm for grouping nearby pixels and assignment of dead trees position.

8.4.1 Subtract DTM from FW LiDAR

A feature was implemented in DASOS for subracting pre-calculated Digital Terrain Model (DTM) saved into .bil files. Generating a DTM is beyond the scope of this research and the DTM files used were provided by Interpine Ltd Group. The provided DTM files were generated using the Quick Terrain Modeller from discrete LiDAR using the parameters shown in Figure 8-7.

The subtraction of the DTM is done during the voxelisation (Section 4). The terrain height is subtracted from the position of the sample before it is inserted into the volume. Please note that each terrain value is not subtracted from the origin of its pulse but from the position of each sample since the terrain value at the origin and the terrain value at the position of a sample may differ.

Figure 8-8 shows an example of a DEM generated before and after the subtraction using DASOS.

8.4.2 Generating feature vectors using DASOS

The feature vectors is a new feature of DASOS (version 2), which was released on the 20th January 2017 [97]. The dead tree detection is its first application. This feature is useful for characterising object inside the 3D space (e.g. trees). For each column

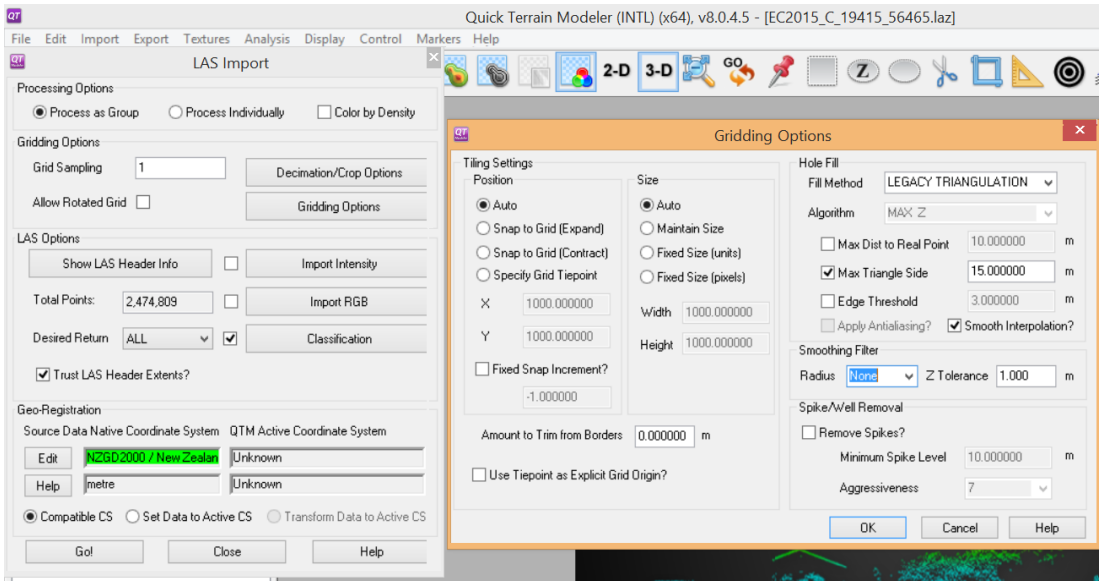
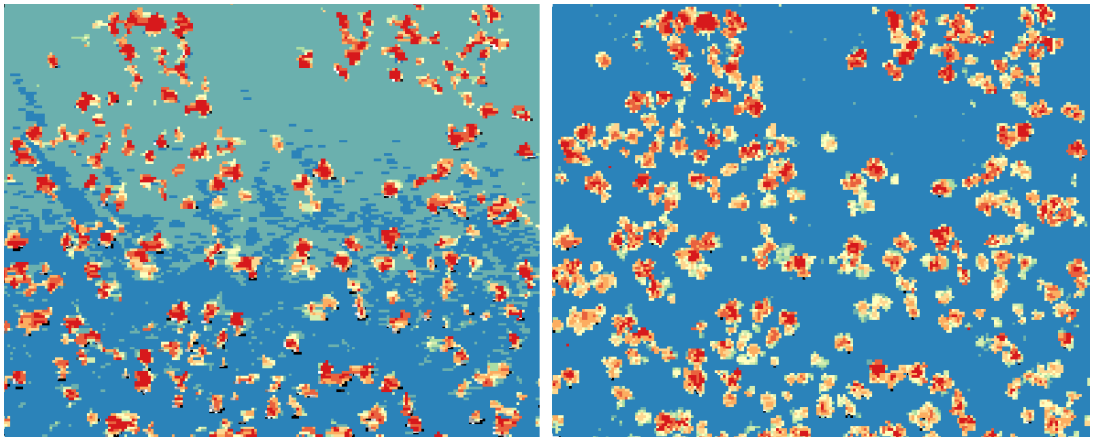


Figure 8-7: Parameters used in Quick Terrain Modeller to obtain the DTM used here.



(a) The DEM before subtracting the DTM (b) The DEM after subtracting the DTM

Figure 8-8: The difference of the DEM before and after subtracting the terrain height. The red indicates big height, while the darker the blue is the lower the DEM is.

of interest within the voxelised FW LiDAR data, information around its local area are exported as feature vector. Multiple feature vectors are listed within .csv files for easy manipulation into software packages specialised in statistical analysis like R and matlab. There are two types of exported information from these local areas: processed and raw. If the processed option is chosen, then information like the distribution of non-empty voxels and the standard deviation of heights are listed. A sample of the exported processed information along with explanations is given in Table 8.3, while the

entire list is provided within the Appendix A. If the exported parameters are raw, then the corresponding intensity values of the local area's voxels are exported. Additionally, there are two available shapes of the local area from where the features are extracted (the cuboid and the cylinder). The size of shape is also user defined. Here, the aforementioned feature of DASOS is used for generating feature vectors used as a likelihood in the classifier.

Explanation of some features of DASOS's 3D priors that proved to be useful for building the classifier		
No	**Label**	**Description**
1	Height_Middle_Column	The height of the middle column of the prior
	Height_Mean	The Mean height of all the columns included in the template
	Height_Median	The Median height of all the columns included in the template
1	Height_Std	The Standard Deviation of the heights of the columns included in the template
2	Top_Patch_Len_Std	The Standard Deviation of all the top patches
3	Dis_Std	The Standard Deviation of the distances between the central voxel and every voxel that contains an intensity above the isolevel
4	Per_Int_Above_Iso	Percentage of voxels that contain an intensity above the isolevel
5	Top_Patch_Len_Mean	The Mean length of all the top patches
	Top_Patch_Len_Median	The Median length of all the top patches
7	Dis_Mean	Mean distance from the central voxel to every voxel that contain san intensity above the isolevel
8	Dis_Median	Median distance from the central voxel to every voxel that contains an intensity above the isolevel
9	Sum_Int_Diff_Z	The Mirror Summed Difference of the intensities using the middle column in the z-axis as the axis of symmetry
10	Sum_Int_Diff_X	The Mirror Summed Difference of the intensities using the middle column in the x-axis as the axis of symmetry

Table 8.3: Explanation of some features of DASOS's 3D priors that proved to be useful for building the classifier. All the features are explained in Appendix A

Within the field data, some plots exist on two flightlines due to the overlapping of the flights. Overlaps happen at the edges of the flightlines and their scan angle

significantly varies. For that reason, each unique set of field plots and corresponding flightlines is considered as a test/training plot. This results into 50 plots. These plots were randomly divided into 5 equal training datasets. Another dataset was also created by merging the first, second and third dataset in order to check whether the increased training data improves the classification accuracy.

The feature vectors generated for each field plot are divided into two categories (processed and raw intensities) and two sub-categories (cylinder and cuboid shape), resulting into four types of feature vectors per plot. For each type, three .csv files are generated. The first one contains the feature vectors derived from areas where the dead trees exist, the second one contains feature vectors from alive trees and the third one contains one feature vector for the area around each column of the voxelised space. The first two are used for training the classifier and the last one for testing. The feature vectors represents the area within either a cuboid or cylinder. The dimensions of those shapes were chosen to be a bit smaller than the estimated average size of the dead trees to reduce the size of the irrelevant information contained within the priors. Figure 8-9 depicts the divisions of the datasets and the information about the feature vectors generated.

8.4.3 Random Forest

Random Forest is able to identify the importance of predicting variables. At first, it generates multiple regression trees by randomly sampling the data at its nodes and choosing the best predicting variables for each sampled data. The variable importance is then defined according to influence it has to the classification once this variable is modified and the rest remain unchanged [98]. In this project, the R package is used for finding the most relevant feature of the 3D priors (Section 8.4.2 in identifying dead trees.

At this point, it worth highlighting that Random Forest failed to find relation between the 3D priors with the "Raw Intensities" due to the irregular shapes of Eucalyptus trees and the variant scan angle of each field plot. Nevertheless, "Raw Intensities" may be useful for other classification, e.g. pine trees in commercial forest, where their shape variance is smaller.

Regarding the "Processed Intensities", Figure 8-10 shows a list with the variable importance according to Random Forest and Table 8.3 gives the explanation of each important variable identified. The most important one is the standard deviation of height. This is reasonable since the canopy of dead trees has bigger height variance in comparison to alive trees whose canopy is leafy. Please note that in Figure 8-10 the union of all datasets is used and that the significant features slightly vary depending

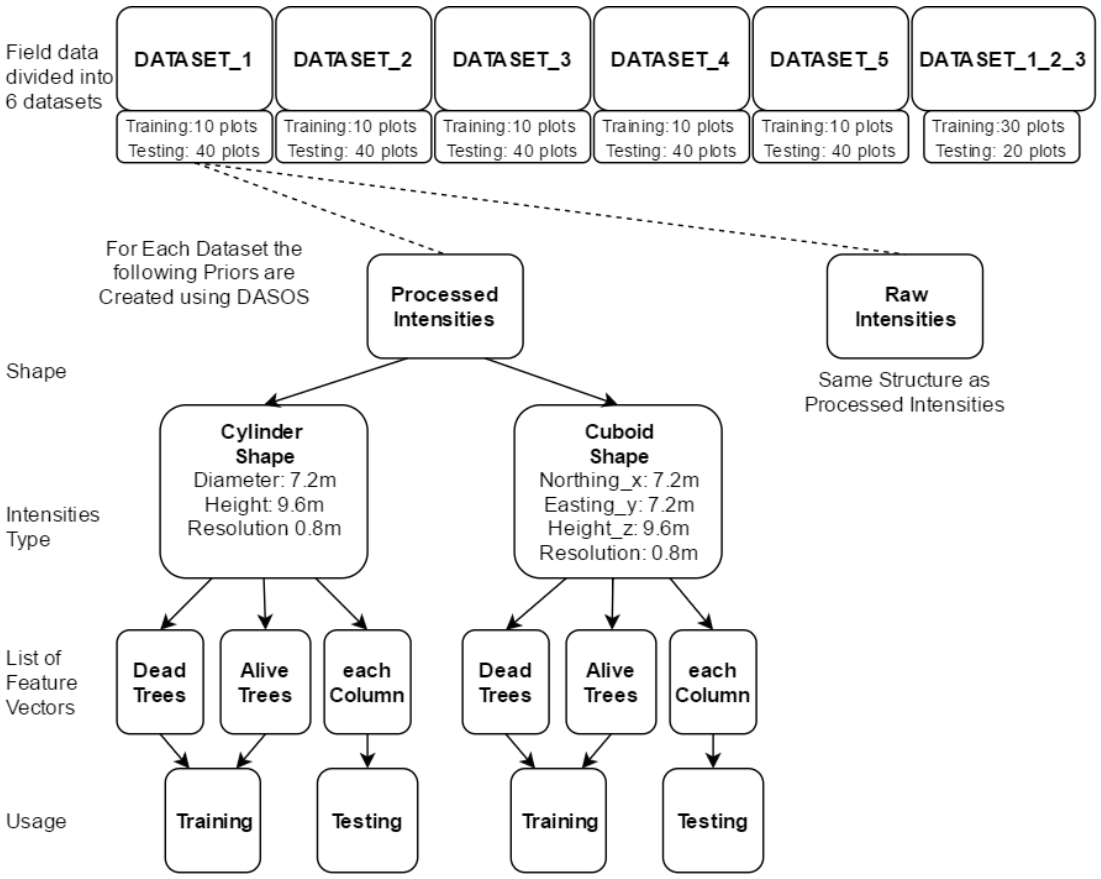


Figure 8-9: This figure shows what feature vectors were created for testing and how they are divided for cross validation.

on each sub dataset used.

8.4.4 **New : Probabilistic Field derived from Weighted K-Nearest Neighbours Algorithm

Once the ten most significant variables are identified using the Random Forest, the $k-$nearest neighbour algorithm is applied to generate a probabilistic field. As mentioned in Section 8.4.2, from DASOS we export training feature vectors of dead and alive trees. There are positive training feature vectors from dead trees and negative feature vectors from alive trees. To reduce bias, the number of dead and alive trees used are the same for each test case.

Let's assume that T is a training dataset with n feature vectors:

$$T : \big(x_n, f(x_n)\big), n = 1 \ldots N. \tag{8.1}$$

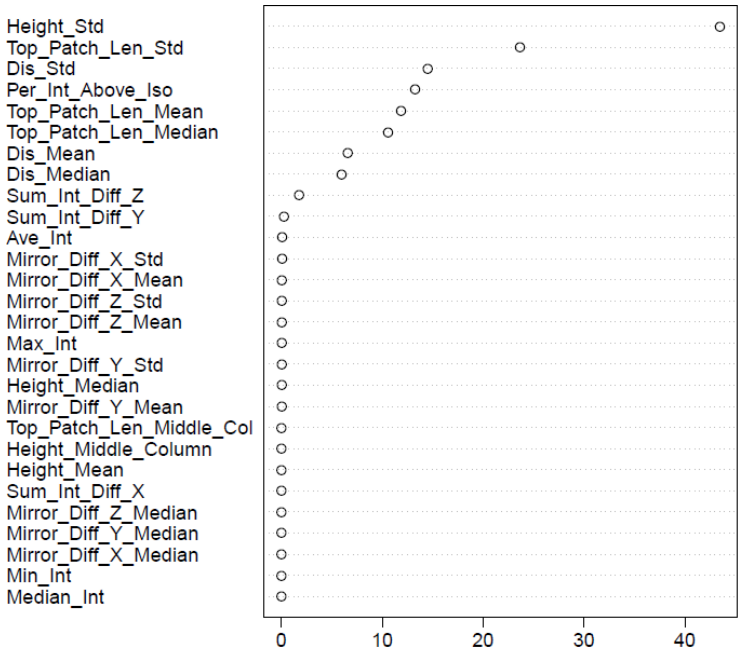


Figure 8-10: Importance of variables, identified using Random Forest.

The outputs of function $f(x_n) \in \{0, 1\}$. The value 0 indicates that the feature vector x_n was derived from an alive tree and the value 1 from a dead tree. For example the dataset T has this form:

$$T : (\boldsymbol{t_1}, 1), (\boldsymbol{t_2}, 0), (\boldsymbol{t_3}, 0), (\boldsymbol{t_4}, 1).....(\boldsymbol{t_n}, 1) \tag{8.2}$$

Every feature vector $\boldsymbol{t_q} \in T$ contains the 10 most important features exported from DASOS, as they were identified from the Random Forest algorithm ($\boldsymbol{t} = \{t_1, t_2, \ldots, t_{10}\}$). Additionally, every feature is associated with a weight value according to its importance ($\boldsymbol{w} = \{w_1, w_2, \ldots, w_{10}\}$). Additionally:

$$\boldsymbol{t_q} \begin{cases} t_1 \\ t_2 \\ \ldots \\ t_{10} \end{cases} \in R^d \qquad\qquad \boldsymbol{w} \begin{cases} w_1 \\ w_2 \\ \ldots \\ w_{10} \end{cases} \in R^d \tag{8.3}$$

Let's define a data vector $\boldsymbol{x} = (x_1, \ldots, x_{10})$ of an unknown population. How do we calculate the probability of vector $\boldsymbol{x}$ to belong to the dead trees population? At first,

the weighted Euclidean distance from $\boldsymbol{x}$ to every $\boldsymbol{t_q} \in T$ is calculated as follow:

$$d(\boldsymbol{t_q}, \boldsymbol{x}) = \sqrt{\sum_{i=1}^{10} \Big(w_i \times (t_{qi} - x_i)^2 \Big)} \tag{8.4}$$

Then the $k-$nearest training samples are selected. In this project $k = 7$ was considered reasonable in respect to the size of training samples. In the future, testing different values of k could evaluate how well the algorithm performs in relation to k. The nearest 7 indices of the training samples are selected as follow:

$$q = \underset{\boldsymbol{t} \in T}{\operatorname{argmin}}\, d(\boldsymbol{t}, \boldsymbol{x}) \tag{8.5}$$

The dataset $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_7}\}$ is a subset of the training samples T and contains the k-nearest indices to $\boldsymbol{x}$. The dataset V may contains samples derived from either dead trees, alive trees or both.

For each $\boldsymbol{v_i} \in V$ a weight $\boldsymbol{u_i}$ is calculated:

$$u_i = \frac{1}{d(\boldsymbol{t_i}, \boldsymbol{i})} \tag{8.6}$$

By the end, the probability of a dead tree is given by the following equation:

$$P(\text{dead}) = \frac{\sum_{i=1}^{k} \Big(u_i \times \delta\big(1, f(\boldsymbol{v_i})\big) \Big)}{\sum_{i=1}^{k} \Big(u_i \times \delta\big(1, f(\boldsymbol{v_i})\big) \Big) + \sum_{i=1}^{k} \Big(u_i \times \delta\big(0, f(\boldsymbol{v_i})\big) \Big)} \tag{8.7}$$

where the function $\delta(a, b)$ returns 1 if a is equal to b and 0 otherwise.

For each column of the voxelised FW LiDAR data, a testing data vector $\boldsymbol{x}$ is created and its probability of been dead is calculated. Figure 8-11 shows the probability field of each column to belong to the dead trees population. The big circle is the location of the field plot and the small circles are the locations of the dead trees. Please note that the white spots contain no data. Those spots appear either when no LiDAR pulse passes through a column or when the pre-defined height of the shape used to calculate the corresponding feature vector is bigger than the elevation of this point.

8.4.5 Filtering

As shown in Figure Figure 8-11, there is Salt and Pepper noise. This noise is removed using a median filter which assigns to every empty pixel the median value of its non-empty neighbouring pixels (Figure 8-12a). A smoothing filter is further applying for further noise reduction (Figure 8-12b).

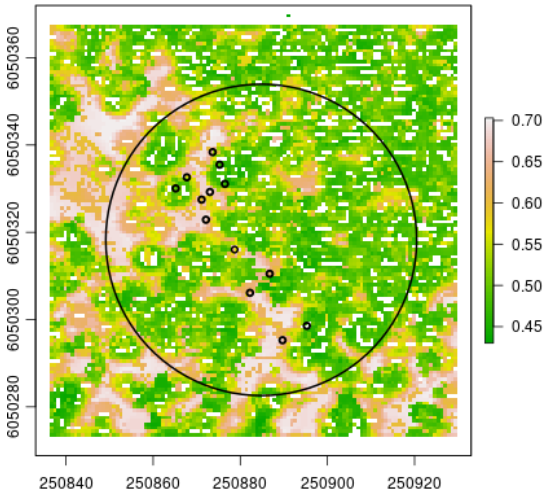
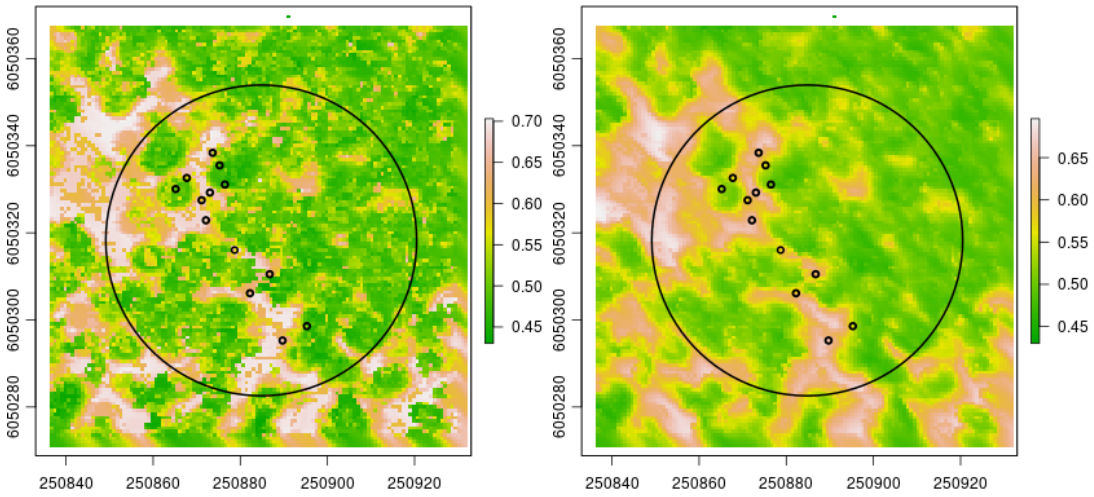


Figure 8-11: The results of the K-NN algorithm

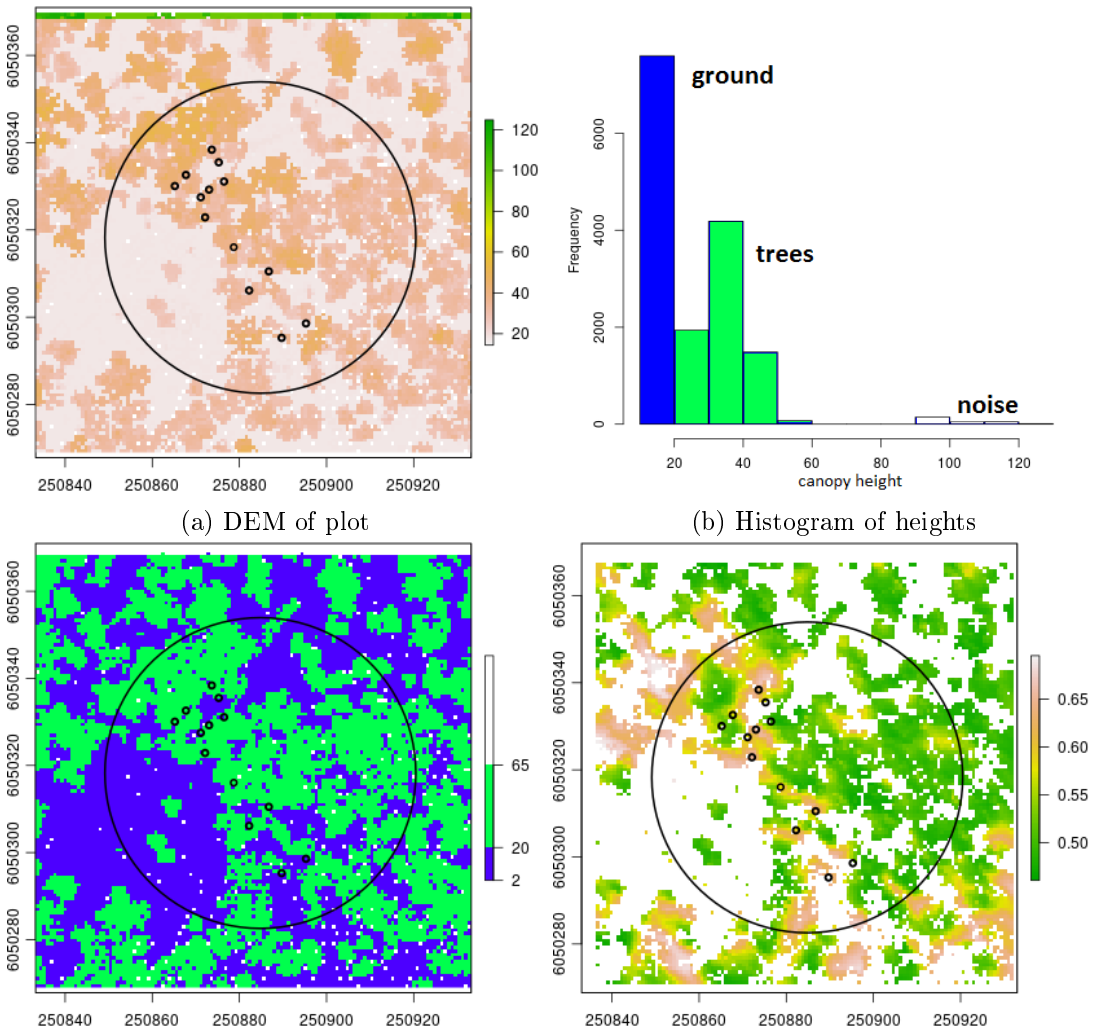


(a) 'Salt and Pepper' noise reduction (b) Filtering using a smoothing kernel

Figure 8-12: Filtering the results of te K-NN algorithm

8.4.6 Removing Ground Pixels

Removing the ground pixels is a trivial task because the DTM has already subtracted from the data and therefore the height of the ground is approximately constant. A histogram of the height values was generated. As shown in Figure 8-13b, there are three well-defined classes (ground, trees and noise). The ground and noise are removed using two thresholds. This processed is illustrated in Figure 8-13.



(a) DEM of plot

(b) Histogram of heights

(c) Thresholds

(d) Results

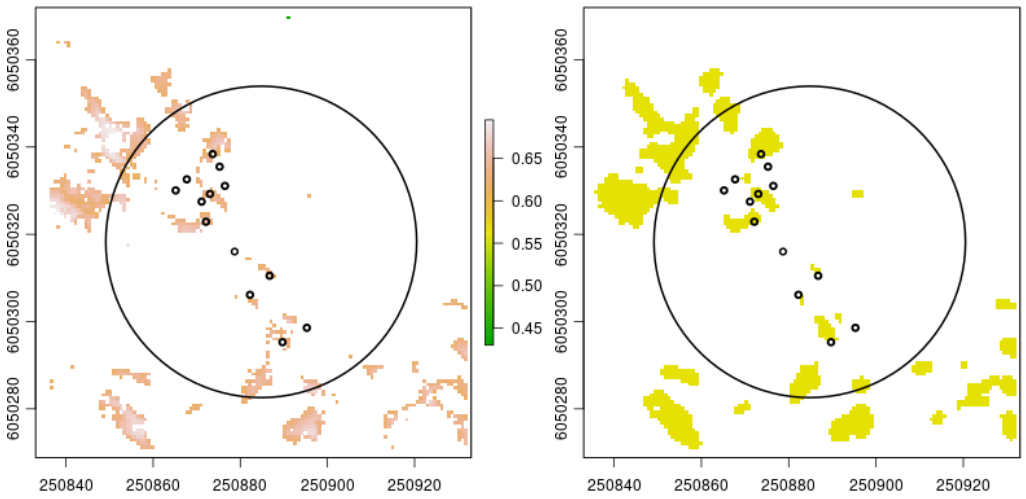
Figure 8-13: Removing the ground pixels

8.4.7 Dead and Alive Threshold, Filtering, Segmentation and Position Assignment

In order to obtain the estimated positions of the dead trees, there are four steps left:

1. Thresholding
2. Filtering
3. Segmentation
4. Position assignment

Up to this stage, we have an image of the probabilistic field and the ground has been removed (Figure 8-13d). After that a threshold for separating dead and alive pixels is chosen using the training data and the alive pixels are removed (Figure 8-14a). The output image contains out-liners; pixels which are classified as dead but have no neigbouring pixels classified as dead. To reduce over-detection of dead trees, these pixels are filtered out (Figure 8-14b). Afterwards, the pixels are grouped into trees relatively to their neighbouring pixels using a seed growth segmentation algorithm (Algorithm 5 and Figure 8-15a). By the end, it is assumed that each segment S is a dead tree and its position is calculated by taking the average geo-spatial location of the pixels that belong in segment S (Figure 8-15b).

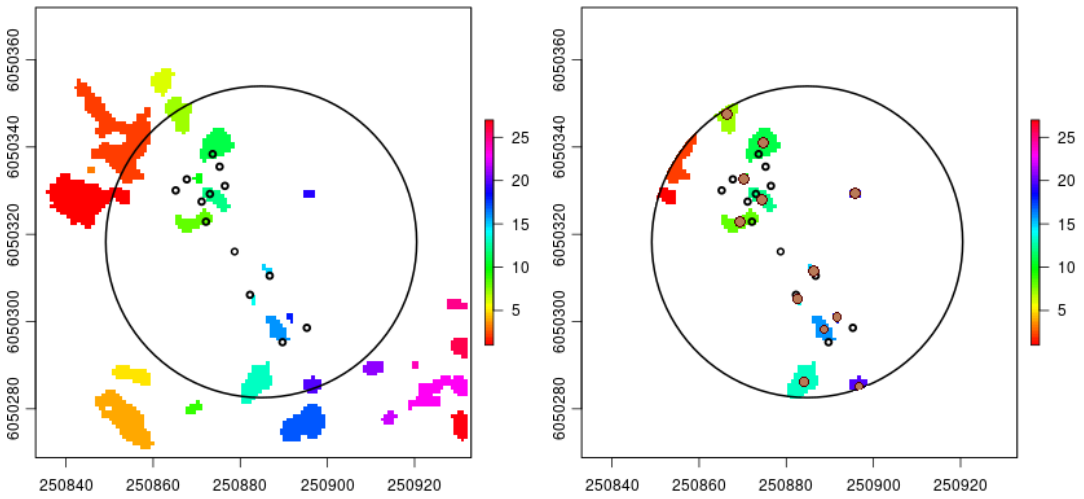


(a) Thresholding dead from alive trees (b) Noise Reduction

Figure 8-14: Thresholding and filtering

Algorithm 5 Seed growth algorithm for segmenting pixels classified as dead

1: $P \leftarrow$ all pixels classified as dead
2: $s \leftarrow 0$
3: **while** not reached the end of set P **do**
4: get next pixel $\boldsymbol{p} \in P$ that is not assigned to a segment
5: Assign pixel $\boldsymbol{p}$ to segment s
6: find $K(\boldsymbol{p_1}, \ldots, \boldsymbol{p_n})$ such that $K \subseteq P$ and every $\boldsymbol{p_i} \in K$ is a neighbour of $\boldsymbol{p}$
7: $\forall \boldsymbol{p_i} \in K$, $\boldsymbol{p} \leftarrow \boldsymbol{p_i}$ and repeat from line 5
8: all pixels of segment s has been labelled
9: $s \leftarrow s + 1$



(a) Segmentation using a seed grownth algortihm

(b) The estimated dead tree positions (brown dots)

Figure 8-15: Segmentation and calculating the dead trees' position.

8.5 Evaluation

The results are evaluated according to the predicted locations of the dead trees and their distance from the actual dead trees. There are three different test undertaken during evaluation: whether the increased training samples improves dead tree detection or not, what shape (cylinder or cuboid) performs better and whether the predictions are better than a random prediction or not.

Please note that, the results have been cross-validated using the fieldplots division depicted in Figure 8-9. The 50 field plots are divided into 5 datasets and each dataset uses 10 plots for testing and the rest 40 for evaluation. Additionally, an extra dataset that uses 30 plots for training and 20 for evaluation was created to check whether the increase amount of training samples improves the precision and recall of the results. For each dataset, feature vectors of processed and raw intensities are generated. But Random Forest failed to identify important variables from the feature vectors with the raw intensities due to the irregular shapes of the dead trees. For that reason, results were only obtained from processed voxel intensity values. The feature vectors with raw intensities should be useful in other application where trees/object shapes are similar. Additionally, a random set of results was generated for comparison. This random prediction uses the probability of a squared meter to contain a dead tree or not and the number of random dead trees distributed within each plot is approximately the same as the number of dead trees that actually exist within each field plot.

Table 8.4 shows the precision and recall percentage achieved using a cylindrical

shape to extract features for the likelihood. The D1, D2, ... , D5 corresponds the five divided datasets in the cross-validation. The D_1_2_3 uses all the training samples from D1, D2 and D3 to train the classifier. As shown in the corresponding charts of precision (Figure 8-16 and recall 8-17), the increased amount of training samples do not improve the prediction. On the one hand, the bigger the training sample set is, the shorter the distances to the k nearest neighbours are. On the other hand, as mention in the challenges (Section 8.3, the field data contain noise and therefore the increased noise compensates the value of the increased samples. A more selective and clean training dataset, that would include trees of similar height and less noise would have definitely improved the results.

Precision (%)										
Distance (m)	1	2	3	4	5	6	7	8	9	10
D1	7.29	12.15	16.1	24.31	32.21	38.9	47.11	49.84	56.23	58.35
D2	2	3.67	8.36	18.39	25.08	33.11	35.78	40.13	46.48	50.5
D3	1.48	5.46	14.2	23.32	29.08	36.6	40.23	46.15	51.38	56.28
D4	0.96	7.24	20.04	28.26	33.09	40.09	44.68	52.17	56.28	62.07
D5	0.75	5.26	8.27	12.03	14.28	21.8	28.57	39.84	47.36	55.63
D_1_2_3	0	8.69	13.04	19.13	24.34	29.56	34.78	36.52	41.73	55.65
Recall (%)										
Distance (m)	1	2	3	4	5	6	7	8	9	10
D1	2.55	9.26	18.84	32.26	45.04	53.35	56.23	58.78	63.25	68.05
D2	8.22	13.48	23.02	31.25	38.48	51.31	62.17	65.78	66.11	69.07
D3	6.69	14.49	26.55	34.77	40.97	48.6	56.61	59.18	60.71	63.41
D4	5.16	15.5	30.09	38.29	43.46	45.89	51.06	52.58	55.31	57.75
D5	0.89	4.45	12.75	24.03	29.37	35.9	41.83	49.85	59.34	61.12
D_1_2_3	0	7.22	14.45	20.07	45.19	50.6	51.8	62.65	63.85	73.49

Table 8.4: The percentage of precision and recall achieved using the cylindrical shape to extract features.

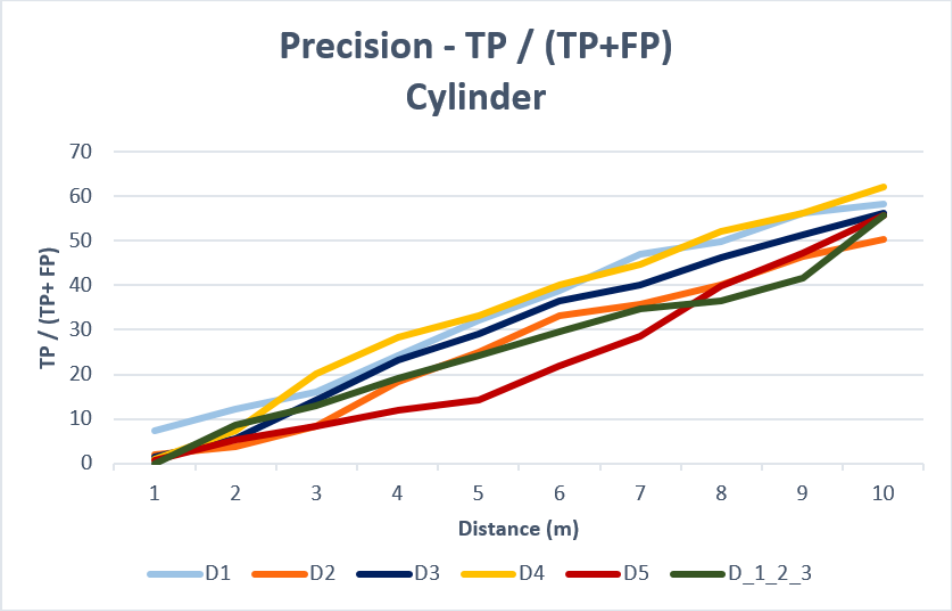


Figure 8-16: Precision results obtained using a cylindrical shape within the voxelised FW LiDAR to extract features. Dataset D_1_2_3 is the dataset that contains more training samples than the rest.

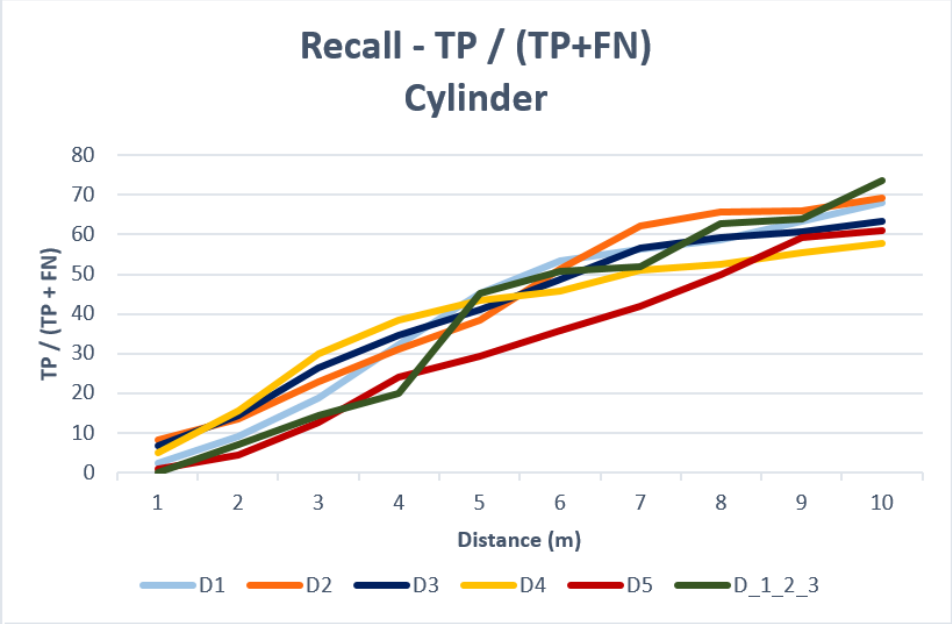


Figure 8-17: Recall results obtained using a cylindrical shape within the voxelised FW LiDAR to extract features. Dataset D_1_2_3 is the dataset that contains more training samples than the rest.

The second comparison is whether a cuboid or a cylindrical shape performs better in extracting useful features of dead tress. The previous Table 8.4 and Figures 8-16 and 8-17 show the results of the prediction using a cylindrical shape to extract features, while Table 8.5 and Figures 8-18 and 8-19 show the results obtained using the cuboid shape. The average results are very close but the cuboid shape has a wider range of good and bad results. It is reasonable for the cylindrical shape to perform better because trees do not have corners and therefore the information retrieved with the cuboid shape are less meaningful. Nevertheless, the cuboid shape is slightly bigger and this may be the justification of the wider range of good/bad results. Since it is a bit bigger it may collect better information from big trees but more noise in respect to small trees. Therefore, the size of the trees plays a significant role for the quality of the classifier.

Precision (%)										
Distance (m)	1	2	3	4	5	6	7	8	9	10
D1	7.78	10.47	14.07	24.85	32.63	40.11	47.9	53.59	59.28	61.97
D2	3.66	12.5	16.37	22.84	26.72	34.26	42.02	46.76	51.72	56.68
D3	1.24	3.42	20.56	25.23	29.28	36.76	41.74	43.92	50.15	53.27
D4	8.36	19.86	33.79	36.58	39.02	44.25	50.52	55.05	63.76	66.89
D5	1.96	1.96	5.88	15.68	19.6	25.49	35.29	41.17	45.09	62.74
Recall (%)										
Distance (m)	1	2	3	4	5	6	7	8	9	10
D1	9.58	19.16	35.14	44.4	50.47	56.86	62.93	65.49	69.96	74.76
D2	10.52	20.39	33.22	37.82	47.36	62.17	67.1	70.39	74.01	75.65
D3	5.48	20	31.93	38.7	46.12	54.83	60.64	67.09	72.9	77.09
D4	7.9	17.93	24.92	26.74	33.13	38.29	45.28	47.41	50.75	52.27
D5	4.74	4.74	5.34	11.86	16.02	16.32	21.95	23.73	27.29	31.75

Table 8.5: This table gives the percentage of precision and recall achieved using the Cuboid shape to extract features.

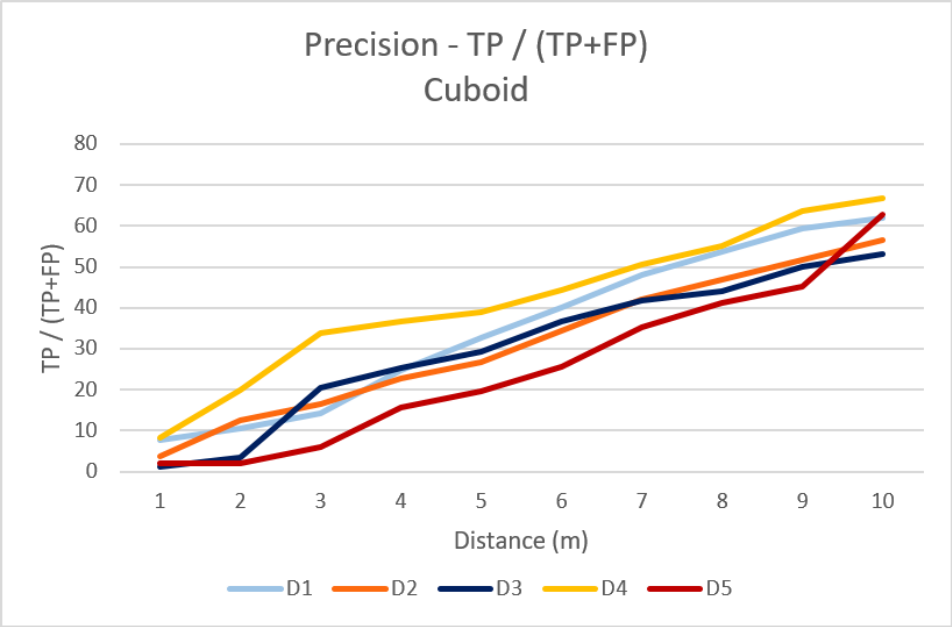


Figure 8-18: Precision results obtained using a cuboid shape within the voxelised FW LiDAR to extract features.

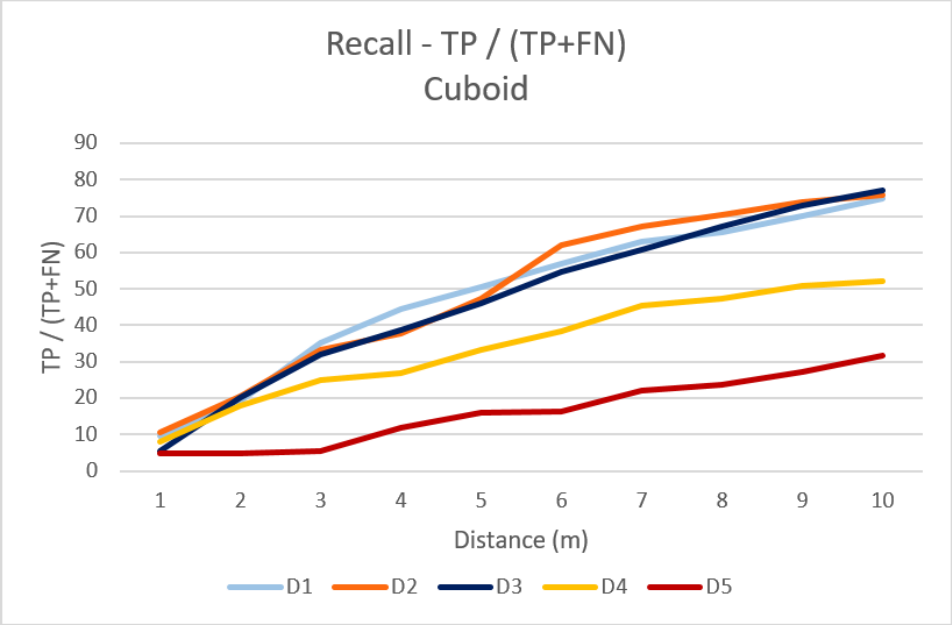


Figure 8-19: Recall results obtained using a cuboid shape within the voxelised FW LiDAR to extract features.

The last comparison is in relation to the random. For each field plot, a random dataset is generated with randomly distributed locations of potential dead trees. The sum of these locations is approximately as the same number of dead trees that exist within the 50 field plots. These random location were evaluated the same way as the as the predicted result using the methodology explained in Section 8.4. The average results obtained using a cylindrical shape, a cuboid shape and the random dataset are shown in Table 8.6 and Figures 8-20 and 8-21. Please note that the dimensions of the shapes used are $7.2m$ diameter or $7.2m$ width for cylinder and cuboid respectively. Therefore, prediction above this distance are above te desire resolution of prediction. From those figures, it is clearly showed that the methodology proposed performs better than the random. This is an important outcome; it is an indication that forest health assessment, including dead trees detection, is possible to be done without tree delineation. Of course, the is a new research direction and many improvements could be done; some of them are mentioned after at the end of the following section.

Precision (%)										
Distance (m)	1	2	3	4	5	6	7	8	9	10
Cylinder	2.50	6.76	13.40	21.27	26.75	34.10	39.28	45.63	51.55	56.57
Cuboid	4.60	9.645	18.14	25.04	29.45	36.18	43.50	48.10	54.00	60.31
Random	0	0	2.56	8.06	11.36	23.81	52.75	57.14	72.16	79.49
Recall (%)										
Distance (m)	1	2	3	4	5	6	7	8	9	10
Cylinder	4.71	11.44	22.26	32.13	39.47	47.02	53.58	57.24	60.95	63.88
Cuboid	7.65	16.45	26.11	31.91	38.63	45.70	51.59	54.83	59.00	62.31
Random	0	0	6.18	7.34	8.88	10.04	12.74	20.85	21.62	28.59

Table 8.6: Distance based evaluation. This table gives the percentage of precision and recall of the average results of each shape (Cylinder and Cuboid), the Random prediction generated for comparison and the the dataset with that its training dataset is three times larger.

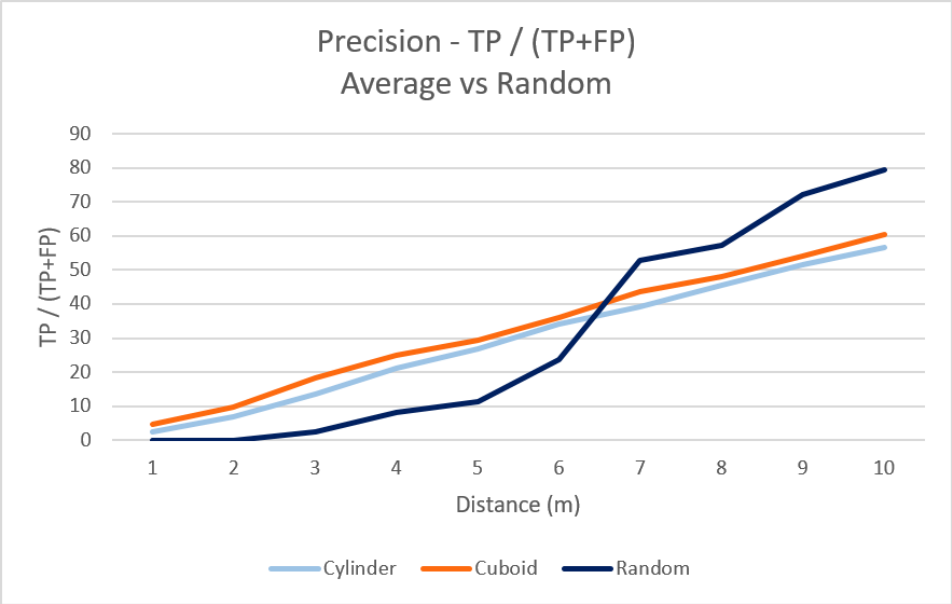


Figure 8-20

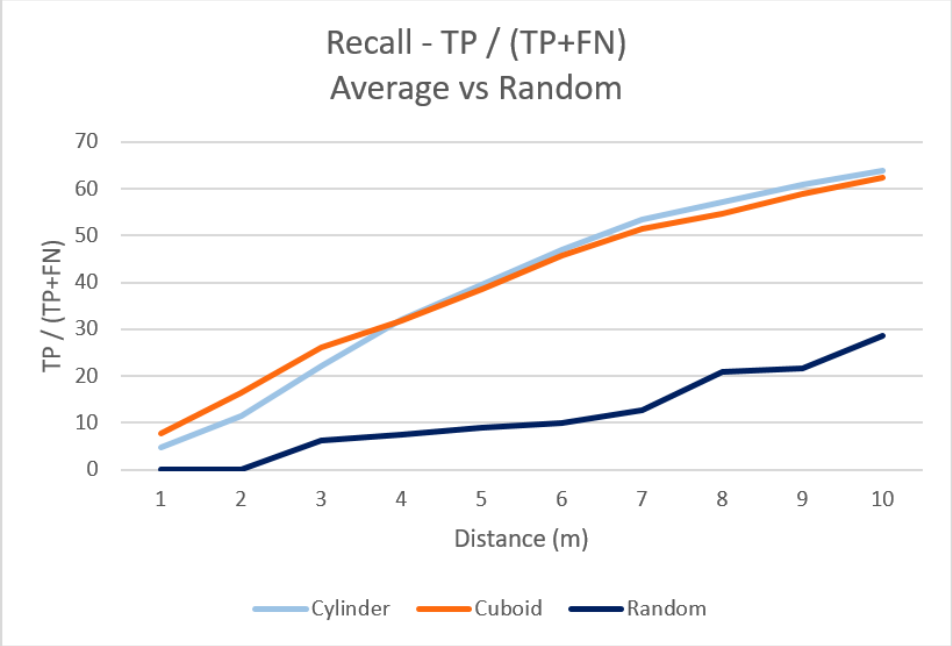


Figure 8-21

8.6 Conclusions and Future Work

The importance of dead wood our ecosystem in large and it is monitor for managing biodiversity. The study area of these project is a native Australian forest with Red River Gum (Eucalyptus) trees, where shortage of hollows available for colonisation is predicted. Dead trees are more likely to be aged and have hollows and therefore detecting them is essential for protecting their habitants.

In this chapter, a new direction of detecting dead standing eucalyptuses was proposed. Previous work on forest health assessment uses tree delineation but this leads to over-segmentation when applied to eucalyptuses due to their irregular shapes and multiple trunk splits. Additionally the density of the acquired LiDAR data makes bottom to top delineation impossible since information about the trunks are missing. Therefore, this thesis investigates the possibility of detecting dead trees from voxelised FW LiDAR data without tree delineation.

Field data were provided by Forestry Cooperation of NSW, Australia and Interpine Ltd group, New Zealand. The GPS positions of the dead and alive trees within the plots are given but the data contain an unknown amount of noise; from the DEM is shown that some trees are positioned on the ground. Additionally, other trees are not predictable from the acquired data due to their small size.

Regarding the methodology, DASOS is firstly used to extract feature vectors characterising dead and alive trees. Then those feature vectors are run over the volume to generate a 2D image for each testing plot with the probability of a pixel to be a dead tree or not. Salt and pepper noise removal and smoothing filters are applied. The ground is afterwards removed and a threshold is defined to separate pixels containing dead and alive trees. Then extra filtering is applied and a seed growth algorithm is used to label each segment. Each segment corresponds to a dead tree prediction and its estimated location is the average position of the pixels that belong to the segment.

Cross validation is used to validate the predicted position of the results. Overall, there are three outcomes. The increase amount of training samples do not improve the results of the classification, probably because while the training datasets increases, the noise increases as well. The feature vectors derived from cylindrical shape are more reliable because the range of the recall and precision percentages was smaller. By the end, the most important outcome is that the results was clearly better than random prediction, justifying this way that it is possible to identify dead trees without tree delineation.

Nevertheless, this is the first research attempting health forest assessment without tree delineation and this direction is in early research stages. Therefore many improve-

ments could be done. Some improvements are proposed here:

- Manually check and improve position accuracy of dead trees using visualisations of the field and acquired FW LiDAR data.
- Separate trees from field data according to their height. Trees with different heights have different shape properties. In this research, the size of the shapes used to derive the feature vectors was constant.
- Make the size of the object adjustable and derive features that are not height dependant.
- Or categorise the trees according to their size and derive a sets of feature vectors according to the height of the trees.
- After the application of the seed growth algorithm, check the size and shape of the segments and look into the possibility of merging two segments into a single tree or dividing big segments into multiple dead tree.
- The system is usually confused at the edges of the alive trees. Adding negative samples from ground and edges of alive trees could further improve prediction.

Chapter 9

Overall Conclusions

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," *Photogrammetric Engineering & Remote Sensing*, vol. 78, no. 12, pp. 1275–1284, 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," *International Journal of Applied Earth Observation and Geoinformation*, vol. 26, pp. 145–15, 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.," *Forest Ecology and Management*, vol. 214, no. 1.

[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] J. Kerle, "Collation and review of stem density data and thinning prescriptions for the vegetation communities of new south wales," 2005.

[95] N. Wilson and N. C. C. of N.S.W, "The flooded gum trees : land use and management of river red gums in new south wales," *The Council, Sydney*, 1995.

[96] E. van Rees, "Trimble's ax60i and ax80," *GeoInformatics*, vol. 7, no. 5.

[97] M. Miltiadou, N. D. F. Campbell, M. Brown, A. S. G., M. Warren, D. Clewley, and M. Grant, "User guide of the 2nd version o dasos," 2017.

[98] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R news*, vol. 2, no. 3.

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

Appendices

Appendix A

DASOS's user guide, released on the 20th of January 2017

A.1 Introduction

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 takeup 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.

The open source software DASOS was developed to encourage foresters to use the FW LiDAR data. DASOS was named after the Greek word "$\delta\acute{\alpha}\sigma o\varsigma$" (=forest) and it was firstly presented at the 36th International Symposium of Remote Sensing of the Enviroment, 2015. The main way of interpreting FW LiDAR data in DASOS is fundamentally different from the state-of-art available software packages. In a few words, the FW LiDAR data are voxelised by inserting the wave samples into a 3D discrete density volume. It accumulates intensity values from multiple shots and stores

them into a 3D regular grid, resolving this way the problem with the sinusoidal footprints pattern of the Leica system. For more information please refer to the related paper at <https://www.researchgate.net/publication/277347868_Alignment_of_hyperspectral_imagery_and_full-waveform_LIDAR_data_for_visualisation_and_classification_purposes>

This user guide aims to give an in depth understanding of DASOS's functionalities. In a few words there are three functionalities of DASOS:

1. the construction of 3D polygon meshes;
2. the generation of 2D metrics aligned with hyperspectral images and
3. the characterisation of objects using feature vectors.

A.2 License

DASOS is released under the GNU General Public Licence, Version 3. The full description of the usage licence is available here:<https://github.com/Art-n-MathS/DASOS/blob/master/License.txt>

The following paper is the paper that introduced DASOS and it must be cited in any publications, software or other media using DASOS:

Miltiadou M., Warren M. A., Grant M., Brown M., 2015, Alignment of Hyperspectral Imagery and full-waveform LiDAR data for visualisation and classification purposes, ISPRS Archives 36th International Symposium of Remote Sensing of the enviroment. [27]

Full paper available here: at:<https://www.researchgate.net/publication/277347868_Alignment_of_hyperspectral_imagery_and_full-waveform_LIDAR_data_for_visualisation_and_classification_purposes>

The 1^{st} sample dataset provided for testing was collected by the NERC Airborne Research and Survey Facility (ARSF). Copyright is held by the UK Natural Environment Research Council (NERC). The data are free for non-commercial use, but NERC-ARSF must be acknowledged in any publications, software or other media that make use of these data.

The 2^{nd} sample dataset provided by Interpine Group Ltd. Copyright is held by Interpine Group Ltd and the data are free for non-commercial use, but Interpine Group Ltd must be acknowledged for any publications, software or other media that make use of these data.

A.3 Installation Guide

A.3.1 Windows

The windows executables are available at:
<https://github.com/Art-n-MathS/DASOS/tree/master/DASOS_win>

DASOS is a command line program and a terminal is required. For Windows XP, Vista and 7, Command Prompt is the default terminal and it can be found from the search tab on the start menu. If you are using Windows 8, then right click on the start icon and choose 'Command Prompt'. Find the directory where DASOS is saved (the command 'dir' shows the files inside the current directory and the command 'cd' to open folder). Once you are in the correct directory, execute the following command to test that the program works fine.

$: DASOS --help [1]

Information of all the available commands should be printed. If an error is occur, then either a .dll file is missing or DASOS is not supported at your computer.

A.3.2 Linux

The source code is available at: <https://github.com/Art-n-MathS/DASOS>.

For compiling DASOS on Linux, there are three major dependencies:

1. qmake-qt4 (or later release) / qtcreator
2. gmtl library - please update the .pro file to point to the correct directory
3. -std=c++11

Once those are installed compile DASOS as shown below:

```
$: qmake-qt4
$: make
```

To test it, write the following command:

$: DASOS --help

Information of all the available commands should be printed.

[1] the '$:' is not included in the command. It just illustrate the start of a command in the terminal

A.4 Instructions

A.4.1 Overview

DASOS is a command line program and can either be used in Command Prompt on Windows or a Linux shell.

At first change directory (cd) to go to the directory DASOS is saved or compiled in. Then for Windows run:

`$: DASOS <arg1> <arg2> ... <argN>`

or `$: DASOS.exe <arg1> <arg2> ... <argN>`

On Linux run:

`$: ./DASOS <ar1> <ar2> ... <argN>`

For consistency this guide uses the 1^{st} Windows example since all of the inputs, parameters and output arguments are the same for both Windows and Linux.

The tags of DASOS are divided into three groups: Inputs, Parameters and Outputs.

1. Inputs (Section A.4.2)
2. Parameters (Section A.4.3)
3. Outputs (Section A.4.4)

Even though many tags are optional or contain default values, it's essential to follow the order <inputs> <parameters> <outputs> because if the outputs are defined first unexpected results may occur, due to adding outputs to the stack before parameters are initialised. The aforementioned Sections give an explanation of all the possible tags of DASOS.

Before proceeding to the explanation, it worth highlighting and numbering the three main outputs of DASOS. The corresponding sub-sections do not only explain the output but also the parameters that are only specific to the corresponding output. The user guide refers to those outputs using their numbers:

1. The generation of 3D polygonal meshes (Section A.4.4.1)
2. The 2D metrics aligned with Hyperspectral Imagery (Section A.4.4.2
3. The feature vectors for local inspection of data (Sub-section A.4.4.3)

At the end of this guide, there is a list of limitations and the dependencies.

A.4.2 Inputs

The inputs are divided into FW LiDAR, Hyperspectral and fieldplots. The FW LiDAR are compulsory for all the functionalities of DASOS. The Hyperpsectral Inputs are optional for the 1st and 2nd output of DASOS (3D polygonal meshes and 2D metrics), while the fieldplots option is compulsory for the 3rd option, the feature vectors.

Table A.1 explains the tags for loading the FW LiDAR files files. Please note that it is compulsory to load one of those options. If more than one FW LiDAR files are loaded then it is essential to keep consistency between projects; load only one type of full-waveform LiDAR data simultaneously. Table A.2 outlines how the hyperspectral imagery is loaded, how to subtract a pre-calculated DTM and also how the file with fieldplots is loaded if the 3rd output option is chosen (feature vectors).

Tags	**Description**
-las <file1> <file2> ... <fileN>	The name/directory of a number of LAS files (i.e. `"C:\Dir Las\LAS1.las"`). It is further suggested to manually define the boundaries of the area of interest when multiples input files are loaded (use command -userLimits explained at Table A.3). Otherwise the boundaries of the first LAS file loaded are used, which may lose data from subsequent input files. Furthermore DASOS only supports LAS1.3 with waveform packet format 4.
-pw <file1> <file2> ... <fileN>	loads a number of pulsewave files (*.pls). Same rules apply as the -las tag
-volume <file>	loads an exported volume, generated using DASOS, instead of reading a LAS or pulsewave file.
-vols <dir>	loads all the exported volume that are inside the given directory "dir". This option must and only be used for generating feature vectors (Section A.4.4.3).

Table A.1: DASOS fundamental file inputs

Tags	Description
-igm <igmFileName>	The name/directory of the .igm file that defines the geolocation of the hyperspectral pixels. This file is an ENVI bil with latitude and longitude per pixel.
-bil <bilFileName>	The name/directory of the .bil file that contains the hyperspectral cube.
-fodis <fodisFile>	The name/directory of the fodis (upward looking illumination sensor) .bil file for hyperspectral imagery
-dtm <dtmFileName>	loads a pre-calculated DTM and subtracts it from the position of each waveform sample before importing it to the volume. Please note that the DTM file format must be .bil and saved into float pointing numbers. Potential further file format limitations may exist. This is optional.
-csv <field-plots.csv>	The input csv file that lists all the trees from a number of field-plot. This is a compulsory input for generating feature vectors;

Table A.2: Optional or output request dependant.

A.4.3 General Parameters

All the general parameters has been pre-defined and they are therefore optional. Nevertheless, parameters are advised to be adjusted for each project. Table A.3 contains information about all the parameters and how to modify the the voxelised FW LiDAR data during construction. Figures A-1 and A-2 show how the results are affected when these parameters are modified.

Tags	Description
-userLimits <maxNorthY> <minNorthY> <maxEastX> <minEastX>	User define boundaries of the area of interest. If not defined then the boundaries of the first file loaded are used (as defined in the header).
-vl <voxelLength>	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. Default value is 2.5m
-nl <noiseLevel>	The noise level is the threshold of the low level filtering applied during voxelisation. Default value is 25. Please note that the intensity of each wave sample is not transformed to volts. Additionally, it is recommended to use the -exportPulses tag to export the amplitude of a few pulses and use those as sample data to define an appropriate noise threshold.
-iso <isolevel>	The iso-level is the intensity boundary that defines whether a voxel is empty or not. This is mostly used during polygonisation. The default value is zero. Please note that noise level and isosurface level are closely related but only the isolevel can be modified from an exported volume.

Table A.3: Description of the all the available tags that customises voxelisation of the FW LiDAR data.

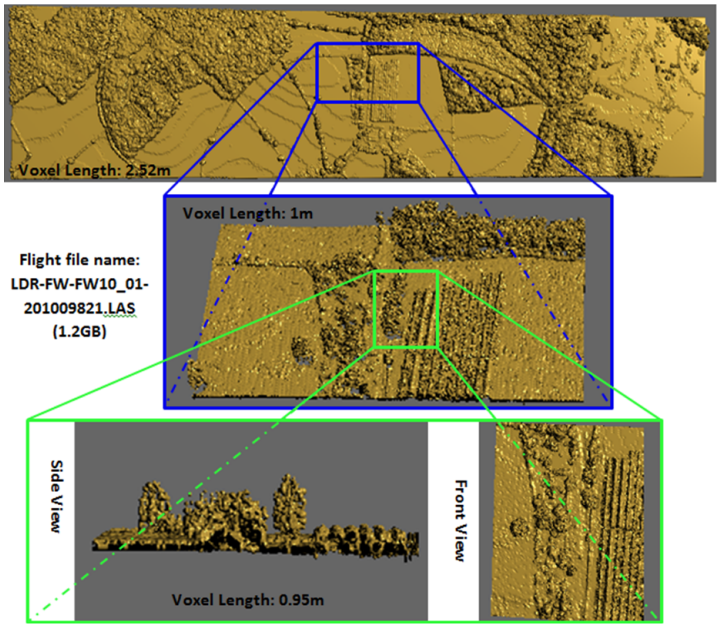


Figure A-1: Selecting Region of Interest

Voxel Length	Visualisation with different voxel lengths	Isolevel	Visualisations with various isolevels	Noise Level	Visualisations with various noise levels
10.0 m		45		5	
5.7 m		15		15	
4.44 m		-45		17	
1.43 m		-60		30	
1.0m		-85		75	
0.67 m		-100		135	

Figure A-2: Effect of modifying the user defined parameters; voxel length, isolevel and noise level.

A.4.4 Outputs

DASOS has three main outputs and three supplementary. At least one of them must be requested for the program to run.

The main outputs are the following:

1. **Polygonal Meshes**: exported into an .obj which is a standard graphics format that stores the vertices, edges and faces of the polygon. An image is also exported if hyperspectral data are loaded. (Section A.4.4.1)

2. **2D Metrics**: information about the scanned area in .asc format. If hyperspectral Images are loaded then aligned metrics from both datasets are available. (Section A.4.4.2)

3. **List of Feature Vectors**: exported into .csv files. Each row of the spreadsheet contains information about a local 3D cylinderal or cubic area. (Section A.4.4.3)

The three suplementary outputs are explained in Table A.4 while the main outputs are explained in Sections A.4.4.1, A.4.4.2 and A.4.4.3 respectively.

Tags	**Description**
--help	It prints a list with all the available commands along with their description.
-exportPulses <noOfPulses> <fileName.csv>	Method that exports a number of pulses into a .csv file. The input <noOfPulses> the number of sample pulses to be exported into the <fileName.csv> file. It is used for deciding the noise level threshold for each project.
-exportVolume c <volumeFileName>	Exports the volume into an ASCII file to speed up future interpolation of the data. 'c' refers to compressed and it's an implicit functionality. If 'c' is not included then a non compressed file is exported, which sometimes is too big to be read back into DASOS. Therefore 'c' should always be included.

Table A.4: The supplementary ouput options of DASOS.

A.4.4.1 Polygonal Meshes

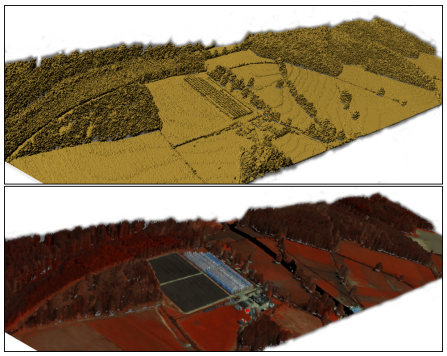
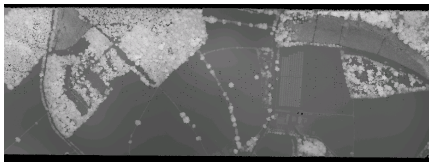
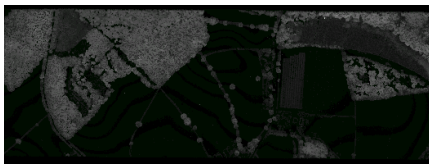
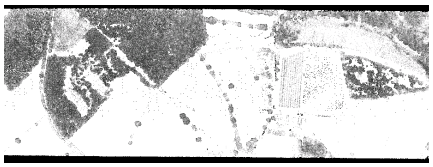
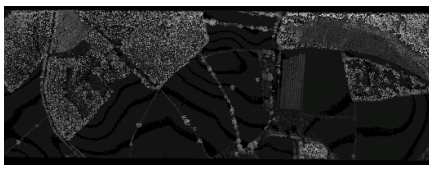
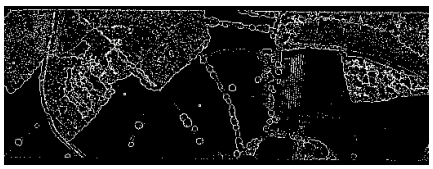
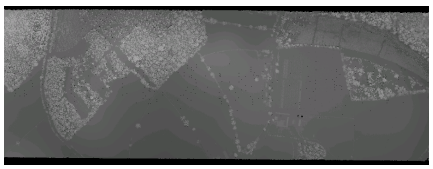
1st Main Output: 3D Polygon Mesh Generation		
Tags	**Description**	**Output Example**
-obj <objFileName>	The input <objFileName> is the name of the .obj file where the polygon representation of the LiDAR file will be exported to. A texture is also exported when hyperspectral images are loaded.	

Table A.5: Description of generating polygonal meshes and example outputs

A.4.4.2 2D Metrics Aligned with Hyperspectral Imagery

2nd Main Output: Generation of 2D metrics aligned with hyperspectral imagery	
Tags	**Description**
-map <type> <outputName>	The available types are the following. Full description of each option is given in Table A.7 along with output examples. • HEIGHT • THICKNESS • DENSITY • FIRST_PATCH • LAST_PATCH • AVERAGE_HEIGHT_DIFFERENCE • LOWEST_RETURN • INTENSITY_MAX • INTESNSITY_AVG • HYPERSPECTRAL_MEAN • NDVI • ALL_FW All the maps are exported into .asc format and can be loaded into QGIS and other software packages. The ALL_FW option generates one metric for each available full-waveform LiDAR related metric and their names are: outputName+metricsType+.asc Table A.7 explains what each metric is and gives output examples.
-map HYPERSPECTRAL <band> <outputName>	The hyperspectral map needs an extra parameter defining which band will be output.

Table A.6: DASOS ouput options

Metric Description	Example
HEIGHT (DEM): The distance between the top non-empty voxel and the lower boundaries of the volume.	
THICKNESS: The distance between the first and last non empty voxels in every column of the 3D volume.	
DENSITY: Number of non-empty voxel over all voxels within the range from the first to last non-empty voxels.	
FIRST_PATCH: The number of non-empty adjacent voxels, starting from the first/top non-empty voxel in that column.	
LAST_PATCH: The number of non-empty adjacent voxels, starting from the last/lower non-empty voxel in that column.	
AVERAGE_HEIGHT_DIFFERENCE: An edge detection algorithm.	
LOWEST_RETURN The height of the lowest non empty voxel (the actual heights are very low and close to each but the example image has been scaled and the different seems bigger)	

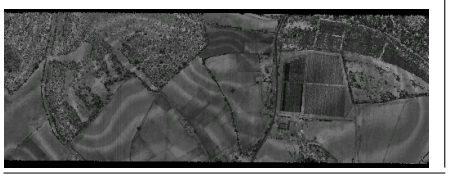
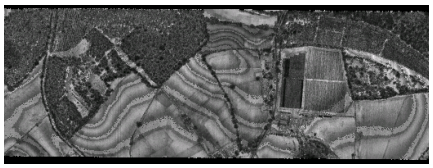
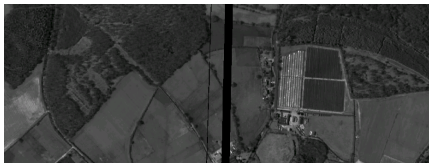
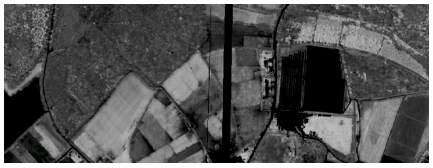
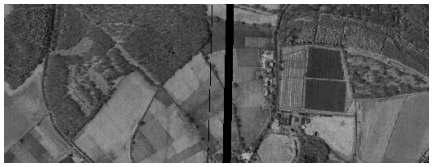
INTENSITY_MAX The maximum intensity of each column	
INTENSITY_AVG The average intensity per column	
HYPERSPECTRAL_MEAN The mean of the hyperspectral spectrum.	
NDVI The Normalised Difference Vegetation Index indicates whether green vegetation exists or not and it is derived from the electromagnetic spectrum as follow: $$NDVI = \frac{NIR - VIS}{NIR + VIS} \quad \text{(A.1)}$$ where the NIR is the near-infrared region of the spectrum (700-2500nm) and VIS is the Visible/Visual spectrum (430-770) [36].	
HYPERSPECTRAL A single user defined hyperspectral band.	

Table A.7: Description of generating polygonal meshes and example outputs

A.4.4.3 List of Feature Vectors

This is useful for characterising object inside the 3D space (e.g. trees). For each column of the voxelised FW LiDAR, information around its local area are exported.

Similar to the previous functionalities of DASOS, the program requires <inputs> <parameters> and <outputs>. Those requirements are described in Tables A.8, A.9 and A.10 respectively. Please note that these inputs are also described with the rest of the inputs in Section A.4.2.

3rd Main Output: List of Feature Vectors - Inputs	
Tags	**Description**
-vols <volDir>	the directory of the volume of interest generated beforehand.
-icsv <fieldplots.csv>	the input csv file that contains all information about the field-plots.

Table A.8: Explanation of how to define the two compulsory inputs to get the 3rd main output of DASOS

Figure A-3 shows an example of a file with fielplots. A file may contain multiple fieldplots, but it has to have at least 6 columns: the 3 columns define the fieldplot (northing, easting and radius) and the rest give information about the trees (northing, easting and class). The order of the columns has no significance. Figure A-3 shows an example. The labels of the those columns could vary and can be defined as explained in Table A.9.

IsDead	Northing	Easting	X	Y	RADIUS
Live	60	70	55	75	40
Live	60	70	75	85	40
Dead	60	70	65	55	40
Live	60	60	20	60	40

Figure A-3: Example of fieldplot input

Additionally, the size and shape of the investigated area from where the features are extracted is user defined and Table A.9 lists all the related, modifiable parameters.

3rd Main Output: List of Feature Vectors - Parameters	
Tags	**Description**
-column <label>	the label of the column that defines the class of each entry (e.g. <label> = isDead)
-class <className or ALL>	the name of the class (e.g. dead or alive) of interest or ALL. If a class is chosen, then only the columns that contain a tree of that class are taken into consideration; a feature list is exported for each tree that belongs to this class only. The ALL option is the area of interest and generates a template for each column that lies inside the voxelised space.
-ttype square <x> <y> <z>	generates a feature vector derived from a cuboid area of size x, y, z voxels. The systems finds the first non empty voxel starting from the top of the column. By default it moves one voxel upwards and sets that to be the top of the cuboid/cylinder. It is highly recommended to use odd numbers, otherwise the centre of the cuboid/cylinder will be wrongly set and unpredicted output values may occur.
-ttype cylinder <h> <r>	generates a cylindrical template with height h and diameter $(2 \times r + 1)$ voxels and height h. The systems finds the first non empty voxel starting from the top of the column. By default it moves one voxel upwards and sets that to be the top of the cuboid/cylinder.
-mheight <n>	moves the template into the y-axis n voxels upwards instead of one which is the default. The value n must be a positive number.
-eparameters <raw or processed>	the 'raw' option saves all the intensity values of the template and the 'processed' option saves parameters derived from the raw intensities. Table A.11 explains how each processed parameters is derived.

Table A.9: Explanation on how to Modify the Parameters of the 3rd Main Output of DASOS

3rd Main Output: List of Feature Vectors - Outputs	
Tags	**Description**
-ocsv <nameStart>	For each .vol file found in the given directory (using -vols), a csv file is exported. The name of each file exported is: <nameStart> + <volFileName> + ".csv" and it contains the list of the feature vectors generated from the corresponding volume

Table A.10: Explanation of the tag that exports the list of feature vectors

Figure A-4 shows examples of two exported list of feature vectors: one with processed parameters and one with raw intensities. In each .csv file exported, each line is a feature vector. The first column is its ID as it defined during run time. The second and third columns define the centroids of each investigated local area (cuboid/cylinder). The other columns contain either processed or raw parameters. If they are processed, then information like mean height and standard deviation of heights are listed. Table A.11 is a full list of all the proccessed parameters. If the parameters are raw, then the corresponding voxel intensity values are exported. The label of each voxel is "v_x_y_z", where "v_0_0_0" is the lower voxel of the cuboid/cylinder and it has the minimum easting and northing it as well.

Index	centroid_x	centroid_y	Height_Middle_Column	Height_Mean	Height_Median	Height_Std	Sum_Int_Diff_X	...
0	251836.109	6048994.5	36	35.5	36	0.943	95.125	...
1	251843.906	6048980.5	19.8	20.1	20.4	0.671	0	...
2	251846.312	6048979	16.8	16	15.6	1.02	169.167	...
3	251849.312	6049022.5	36	35.7	36.6	0.964	169.278	...
4	251851.703	6048988	17.4	16.2	16.2	0.346	408.065	...
5	251852.906	6048975	27	26.4	26.4	0.917	68.537	...
6	251857.109	6048974	17.4	17.4	18	0.849	162.25	...
7	251858.312	6049010.5	40.8	40	39.6	1.02	251.36	...
8	251860.703	6048984	17.4	16.6	16.2	0.663	67.883	...
9	251861.312	6049000	19.8	20.1	20.4	0.671	0	...

Index	centroid_x	centroid_y	V0_0_0	V0_0_1	V0_0_2	V0_0_3	V0_0_4	V0_1_0	V0_1_1	V0_1_2	V0_1_3	...
0	251836.109	6048994.5	7	14	10	26	0	0	9	10.25	11.875	...
1	251843.906	6048980.5	0	0	0	0	0	0	0	0	0	...
2	251846.312	6048979	9	60.75	70.75	13	8	0	0	0	7.667	...
3	251849.312	6049022.5	48.556	93.222	20.5	0	7	0	0	0	0	...
4	251851.703	6048988	100.2	53.222	10.5	7.143	0	0	0	0	47.25	...
5	251852.906	6048975	0	0	0	0	0	26.875	0	10.444	13.182	...
6	251857.109	6048974	0	0	0	0	0	45.667	93	16.333	7.25	...
7	251858.312	6049010.5	0	0	0	0	0	0	0	8	6	...
8	251860.703	6048984	0	45.75	8	7.333	0	0	0	0	6.8	...
9	251861.312	6049000	0	0	0	0	0	0	0	0	0	...

Figure A-4: Example of .csv files with a list of feature vectors exported.

Explanation of the List of Feature Vectors Output with the Processed Intensities	
Label	**Description**
Height_Middle_Column	The height of the middle column of the cuboid/cylinder
Height_Mean	The Mean height of all the columns included in the template
Height_Median	The Median height of all the columns included in the template
Height_Std	The Standard Deviation of the heights of the columns included in the template
Sum_Int_Diff_X	The Mirror Summed Difference of the intensities using the middle column in the x-axis as the axis of symmetry
Sum_Int_Diff_Y	The Mirror Summed Difference of the intensities using the middle column in the y-axis as the axis of symmetry
Sum_Int_Diff_Z	The Mirror Summed Difference of the intensities using the middle column in the z-axis as the axis of symmetry
Max_Int	The maximum intensity found inside the cuboid/cylinder
Min_Int	The minimum intensity found inside the cuboid/cylinder
Ave_Int	The average intensity of the voxels that contain an intensity above the isolevel
Median_Int	The median intensity of the voxels
Per_Int_Above_Iso	Percentage of voxels that contain an intensity above the isolevel
Dis_Mean	Mean distance from the central voxel to every voxel that contain san intensity above the isolevel

Dis_Median	Median distance from the central voxel to every voxel that contains an intensity above the isolevel
Dis_Std	The Standard Deviation of the distances between the central voxel and every voxel that contains an intensity above the isolevel
Top_Patch_Len_Middle_Col	The length of the top patch of the middle column of the cuboid/cylinder
Top_Patch_Len_Mean	The Mean length of all the top patches
Top_Patch_Len_Median	The Median length of all the top patches
Top_Patch_Len_Std	The Standard Deviation of all the top patches
Mirror_Diff_X_Mean	The Mean Mirror Difference of the voxel intensities with the middle column of the x-axis as the symmetric axis
Mirror_Diff_X_Median	The Median Mirror Difference of the voxel intensities with the middle column of the x-axis as the symmetric axis
Mirror_Diff_X_Std	The Standard Deviation Mirror Difference of the voxel intensities with the middle column of the x-axis as the symmetric axis
Mirror_Diff_Y_Mean	The Mean Mirror Difference of the voxel intensities with the middle column of the y-axis as the symmetric axis
Mirror_Diff_Y_Median	The Median Mirror Difference of the voxel intensities with the middle column of the y-axis as the symmetric axis
Mirror_Diff_Y_Std	The Standard Deviation Mirror Difference of the voxel intensities with the middle column of the y-axis as the symmetric axis
Mirror_Diff_Z_Mean	The Mean Mirror Difference of the voxel intensities with the middle column of the z-axis as the symmetric axis

Mirror_Diff_Z_Median	The Median Mirror Difference of the voxel intensities with the middle column of the z-axis as the symmetric axis
Mirror_Diff_Z_Std	The Standard Deviation of the Mirror Difference of the voxel intensities with the middle column of the z-axis as the symmetric axis

Table A.11: Explanation of the processed parameter exported within a feature vector

A.5 Exercises

A.5.1 Sample Data

These exercises will give you an in depth understanding of DASOS, while working with real examples. At first, copy the folder `"DASOS_userGuide"` into your `C:\` drive. This folder is available to download from `<https://github.com/Art-n-MathS/DASOS/tree/master/DASOS_win>`: To ease typing, all the example commands are given in the ExerciseCommands.bat file, which can be opened in a text editor.

There are three datasets provided for the following exercises and they are available at: `<https://www.dropbox.com/sh/hzpl16gue5xvjmb/AADQsJOsqKkxOlCX4mJjvBPVa?dl=0>` and `<https://plymouthmarinelaboratory.webex.com/plymouthmarinelaboratory/j.php?MTID=m305f59dda16e653b2946c6a3b00e93f4>`. Please copy the data inside the directory <DASOS/DASOS_win/SampleDATA> check that the following files are included:

1. 1^{st} sample dataset inside < `C:\DASOS_userGuide\SampleDATA\DATASET_1`>:
   (a) `LDR-FW-FW10_01-201009821.LAS`
   (b) `e098211b_FODIS.bil`
   (c) `e098211b_FODIS.bil.hdr`
   (d) `e098211b_masked.bil`
   (e) `e098211b_masked.bil.hdr`
   (f) `e098211b_osgn.igm`
   (g) `e098211b_osgn.igm.hdr`
   (h) `Readme.txt`
2. 2^{nd} sample dataset inside < `C:\DASOS_userGuide\SampleDATA\DATASET_2`>

(a) `Australia_1.pls`

(b) `Australia_1.wvs`

(c) `Australia_1_dtm.bil`

(d) `Australia_1_dtm.hdr`

(e) `Australia_2.las`

(f) `Australia_2.wdp`

(g) `Australia_2_dtm.bil`

(h) `Australia_2_dtm.hdr`

(i) `Australia_3.las`

(j) `Australia_3.wdp`

3. 3^{rd} sample dataset inside < `C:\DASOS\_userGuide\SampleDATA\DATASET_3`>

   (a) `myTestVol_.vol`

   (b) `myTestVol_flat.vol`

   (c) `testFieldplot.csv`

Information about data usage and related license are given in Section A.2

Once all the files are copied across, open the command Prompt and type:

```
$: cd C:\DASOS_userGuide\DASOS
```

This will bring you to our working directory. In case you are using a different directory then go to your work directory inside the folder DASOS and the rest of the commands should work OK.

A full guide of all the available tags is given with the following command.

```
$: DASOS --help
```

The same information can be found inside the Readme.txt file and this User Guide (Section A.4).

A.5.2 Exercises

A.5.2.1 Deciding Noise Threshold

The following examples export the amplitudes of 12 pulses into a .csv file to help us decide what noise threshold to use.

```
$:DASOS -las ..\SampleDATA\DATASET_1\LDR-FW-FW10_01-201009821.LAS
```

```
 -exportPulses 12   ..\LAS21pulsesSamples.csv

$: DASOS -las ..\SampleDATA\DATASET_2\Australia_2.las -exportPulses 12
   ..\Australia_2_pulsesSamples.csv
```

A.5.2.2 Exporting metrics from DASOS

The following commands export a height map into .asc files. These files can be used in QGIS. This will give us the location of the flightlines and the relation between them.

```
$: DASOS -las ..\SampleDATA\DATASET_2\Australia_2.las -nl 6 -vl 2 -map
 height   ..\Australia_2_vl2_height

$: DASOS -las ..\SampleDATA\DATASET_2\Australia_3.las -nl 6 -vl 2 -map
 height   ..\Australia_3_vl2_height
```

Generating a single map at the beginning is useful for deciding which flightlines lie inside the area of interest.

A.5.2.3 Loading Multiple Flightlines

As mentioned before, for loading multiple flightlines it is suggested to manually define the boundaries of the area of interest. The following command loads two flightlines, generates a volume from the area of interest and exports it into the Australia2-3.vol file.

```
$: DASOS -las ..\SampleDATA\DATASET_2\Australia_3.las
   ..\SampleDATA\DATASET_2\Australia_2.las -nl 6 -vl 2 -iso 4 -userLimits
 6199990  6199639  762405 761951 -exportVolume c ..\Australia2-3.vol
```

A.5.2.4 Exporting Metrics

The following command loads the pre-computed volume and creates a height map and all the FW related metrics. Please note that height is also a FW related metric, therefore it will be created twice.

```
$: DASOS -volume ..\Australia2-3.vol -map height ..\Australia2-3 -map
all_fw   ..\Australia2-3
```

A.5.2.5 Subtracting Pre-computed Digital Terrain Model

The next command loads two LAS files, a pre-computed DTM file is subtracted from the wave samples' positions while the volume is created, the volume is exported into the `Australia2-3_dtm.vol` file and finally it exports a height metric.
Please note that when a DTM is introduced, a new volume must be created. Since the volumetric files are raster data and contain no information about pulses.

```
$: DASOS -las ..\SampleDATA\DATASET_2\Australia_3.las
  ..\SampleDATA\DATASET_2\Australia_2.las   -dtm
 ..\SampleDATA\DATASET_2\Australia_2_dtm.bil -nl 6 -vl 2 -iso 4
 -userLimits   6199990  6199639 762405 761951 -exportVolume
c ..\Australia2-3_dtm.vol -map  height   ..\Australia2-3_vl2_dtm_height
```

You may then use the same volume to export more metrics:

```
$: DASOS -volume ..\Australia2-3_dtm.vol -map AVERAGE_HEIGHT_DIFFERENCE
   ..\Australia2-3_dtm_AVG_height_diff
```

A.5.2.6 Pulsewave Data

As mentioned before, it is suggested to first export the amplitudes of a few pulses to decide on an appropriate noise threshold.

```
$: DASOS -pw ..\SampleDATA\DATASET_2\Australia_1.pls -exportPulses 15
   ..\PLS_amplitudeSamples.csv
```

And then you can generate the desired metrics:

```
$: DASOS -pw ..\SampleDATA\DATASET_2\Australia_1.pls -nl 5 -dtm
   ..\SampleDATA\DATASET_2\Australia_1_DTM_1m.bil -vl 3 -map thickness
 PLS_vl3_thickness   -exportVolume ..\Australia_1_vl3_dtm.vol
```

A.5.2.7 Polygon Representation

DASOS create 3D polygon representation using the '-obj' tag. The 3D polygon representations are exported into .obj format, which can be visualised using animation software packages. For this workshop we are using Meshlab because it is a free tool and it can handle millions of triangles.
Meshlab is available to download from here: `<http://meshlab.sourceforge.net/>` and it is also included into our working directory `"DASOS_userGuide"`.

An example of generating polygons is given below:

```
$: DASOS -las ..\SampleDATA\DATASET_1\LDR-FW-FW10_01-201009821.LAS -nl 20
 -vl 1.7  -obj   ..\LAS21.obj -exportVolume c ..\LAS21_vl1.7.vol
```

The generated volume is also saved because we need it for the following exercises.

A.5.2.8 Hyperspectral Imagery

One of the key functionalities of DASOS is the alignment with the hyperspectral imagery. DASOS can export 3D coloured polygon representations and aligned metrics between FW LiDAR and hyperspectral data.

For the 3D coloured polygon representations you must not use any directory for the exported .obj file Analysisname because the link between the texture and the .obj file will not work. Here is an example:

```
$: DASOS -volume ..\LAS21_vl1.7.vol -bil
 ..\SampleDATA\DATASET_1\e098211b_masked.bil -igm
   ..\SampleDATA\DATASET_1\e098211b_osgn.igm -fodis
   ..\SampleDATA\DATASET_1\e098211b_FODIS.bil -rgb 240 78 23 -obj
   LAS21_coloured.obj
```

The LAS21.obj file will be saved into the current directory, which in our case is: `C:\DASOS_userGuide\DASOS`.

Please note that the following command should give the same results, but as mentioned before importing an exported volume is faster than generating from scratch.

```
$: DASOS -las ..\SampleDATA\DATASET_1\LDR-FW-FW10_01-201009821.LAS -nl 20
-vl 1.7  -bil    ..\SampleDATA\DATASET_1\e098211b_masked.bil -igm
   ..\SampleDATA\DATASET_1\e098211b_osgn.igm -fodis
   ..\SampleDATA\DATASET_1\e098211b_FODIS.bil -rgb 240 78 23 -obj
 LAS21_coloured.obj
```

An examplre of generating aligned metrics is given below. The NDVI map is quite slow, so we may need to wait a bit for that.

```
$: DASOS -volume ..\LAS21_vl1.7.vol -bil
 ..\SampleDATA\DATASET_1\e098211b_masked.bil -igm
   ..\SampleDATA\DATASET_1\e098211b_osgn.igm -fodis
   ..\SampleDATA\DATASET_1\e098211b_FODIS.bil -map hyperspectral 140
..\LAS21_band140 -map    height ..\LAS21_height -map NDVI ..\LAS21_ndvi
```

A.5.2.9 All Commands Together

Of course, we are able to use multiple outputs into a single command, even though that's not recommended due to the long processing time. An example of merging previous commands into one is given below:

```
$: DASOS -las ..\SampleDATA\DATASET_1\LDR-FW-FW10_01-201009821.LAS -nl 20
-vl 1.7 -bil    ..\SampleDATA\DATASET_1\e098211b_masked.bil -igm
   ..\SampleDATA\DATASET_1\e098211b_osgn.igm -fodis
   ..\SampleDATA\DATASET_1\e098211b_FODIS.bil -rgb 240 78 23 -obj
 LAS21_coloured.obj -map
   hyperspectral 140 ..\LAS21_band140 -map  height ..\LAS21_height -map
 NDVI ..\LAS21_ndvi   -exportVolume ..\LAS21_vl1.7.vol
```

A.5.3 Exporting feature vectors from exported voxelised FW LiDAR

This examples takes as input two test .vol files and a fieldplot file. The file named "myTestVol_flat.vol" contains a flat surface, while inside the "myTestVol_.vol" the middle column of the first dead tree that is defined inside the "testFieldplot.csv" is one voxel higher. The covered area of the two .vol files is identical and for that reason the fieldplot circle lies inside both files. The following command produces a list of vectors with features derived after processing the voxel intensities of the cuboids that contain dead trees according to the input field data:

```
   $: DASOS -vols ..\SampleDATA\DATASET\_3  -icsv
..\SampleDATA\DATASET_3\testFieldplot.csv -eparameters processed -column
isDead -class dead -ttype square 3 3 5 -ocsv templatesProcessedCuboid
```

The following command produces a list of vectors with the voxel intensities of cylinders that contain dead trees according to the input field data:

```
$: DASOS -vols ..\SampleDATA\DATASET_3  -icsv
..\SampleDATA\DATASET_3\testFieldplot.csv -eparameters raw -column
 isDead -class ALL -ttype cylinder 5 3 -ocsv templatesALLRawCylinder
```

A.6 Limitations

Limitation and bugs have been reported throughout the report, but here is a short summary of them.

- Exporting polygon representation could end up generating a bunch of cones instead of a nice smooth surface.
- Subtracting DTM depends on the input file format and, by subtracting the height, the input data may end up outside the boundary of the volume.
- DASOS may be not be perfectly portable to all systems as development and testing was done on two computers only.
- The raw waveform amplitude is used as intensity and it hasn't been converted to an absolute digitizer voltage, for the LIDAR systems where these raw values are scaled.
- Intensities also have not been calibrated.
- Sometimes memory allocation exceptions occur.

For full bug reports and under development improvements please check the following link:
`<https://docs.google.com/spreadsheets/d/10yE5p463cLA_GtKkyiaWEzScW7N9cVxbPs5yOmuXuZY/edit?usp=sharing>`

A.7 Related Forums and Social Media

Online social media are used for sharing DASOS updates and discussing issues or potential improvements. Information about DASOS can be found in the following:

- Google Groups: DASOS - the native full-waveform (FW) LiDAR software
  `<https://groups.google.com/forum/#!forum/dasos---the-native-full-waveform-fw-lidar-`
  This group is used for bringing potential issues and possible improvements up in discussion.
- Blogger: ART & M@thS
  `<http://miltomiltiadou.blogspot.co.nz/2015/03/las13vis.html>`
  This blog is more general. The blog contains updates and explanation of DASOS but usually the code used in DASOS is broken down into small projects and explained how they can be used in other applications.
- Twitter: `@MiltoMiltiadou`
  Milto Miltiadou's twitter, where all the updates and news of DASOS are published.

Appendix B

Case Study: Field Work in New Forest

B.1 Introduction

This section is a case study containing field work from a programmers 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 on the ground 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 [99]. 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 [99]. 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 [99], 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

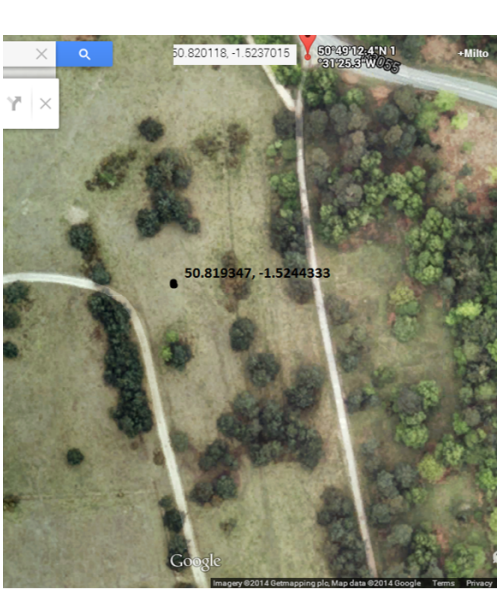
identified and estimates of the approximate heights of the trees were also noted down.

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 [99],
- the coloured tree species map and
- the approximated height map.

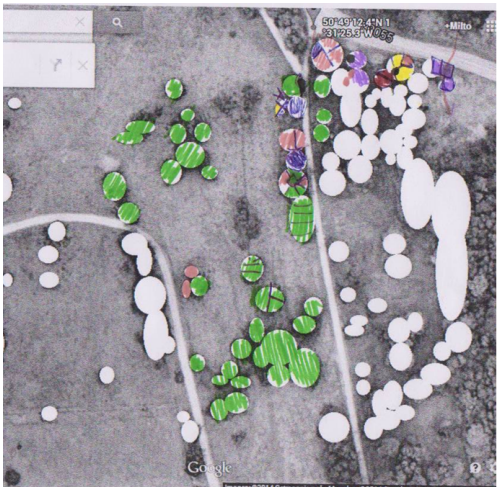
Comparing the validation dataset created with the classifications done at Bournemouth University (which were not validated), it is clear there are misclassifications. This is shown in Figures B-1 and B-2 and it is likely that is occurs due to the over-segmentation of trees. Those wrong classifications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



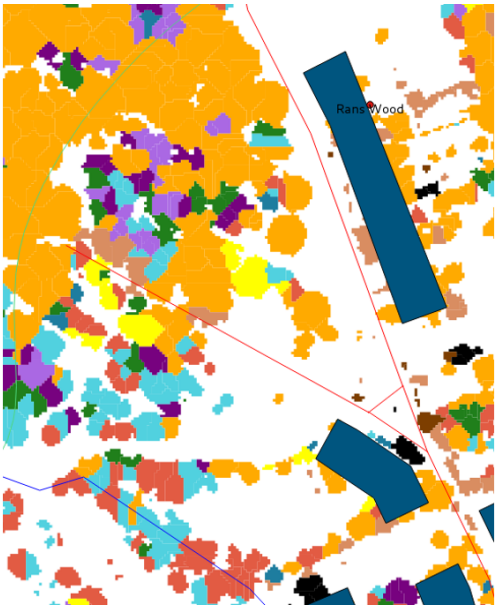
(c) Tree species map, from field work

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

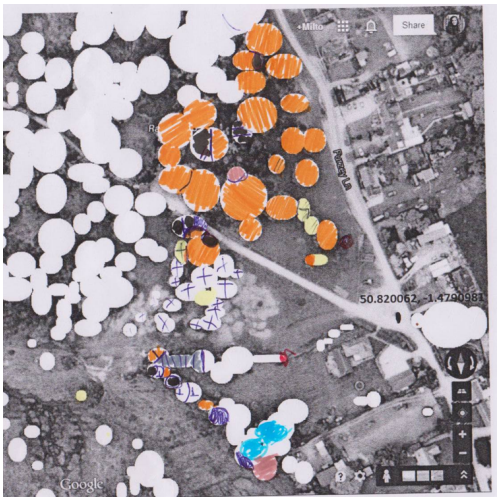
Figure B-1: The first area of interest and the related maps.

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.

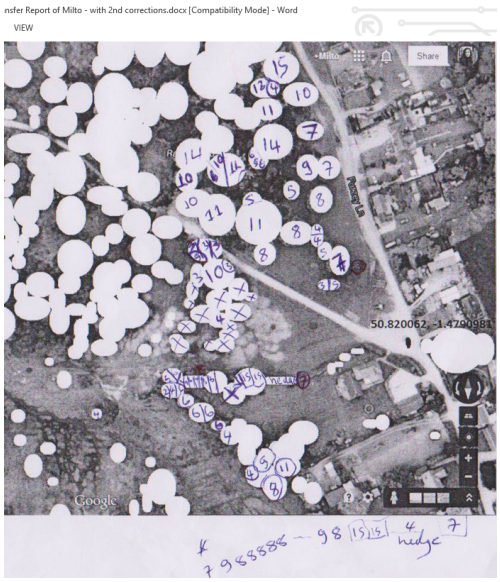
(a) Google map screenshot



(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.

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 time consuming 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

Figure B-8: Trees that have been cut down

in the tree assessment held by Sumnall [99] 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 background clutter that need to be invariant to.

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.