Scalable information extraction from point cloud data obtained by mobile laser scanner

Wang, Jinhu

DOI
10.4233/uuid:81d9473e-667e-4301-bd48-f7f0218974af

Publication date
2017

Document Version
Final published version

Citation (APA)
Wang, J. (2017). *Scalable information extraction from point cloud data obtained by mobile laser scanner*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:81d9473e-667e-4301-bd48-f7f0218974af

Scalable information extraction from point cloud data obtained by mobile laser scanner

Scalable information extraction from point cloud data obtained by mobile laser scanner

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op vrijdag 7 Juli 2017 om 12:30 uur

door

Jinhu Wang

Master of Science in Signal and Information Processing,
University of Chinese Academy of Sciences, Beijing, China

geboren te Yan'an, Shaanxi, China

Dit proefschrift is goedgekeurd door de:

promotor: Prof. Dr. M. Menenti
copromotor: Dr. R.C. Lindenbergh

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. M. Menenti,	Technische Universiteit Delft
Dr. R.C. Lindenbergh,	Technische Universiteit Delft
Onafhankelijke leden:	
Prof. dr. ir. R.F. Hanssen	Technische Universiteit Delft
Prof. dr. N. Pfeifer	Technische Universität Wien
Prof. dr. H.G. Maas	Technische Universität Dresden
Prof. dr. Z. Kang	China University of Geosciences (Beijing)
Dr. S.O. Elberink	University of Twente

Keywords: Mobile Laser Scanning, voxels, octrees, geometric information, individual tree separation, object recognition

ISBN 978-94-92683-65-6

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Contents

Summary

The rise of intelligent transportation, autonomous driving and 3D virtual cities demands highly accurate and regularly updated 2D and 3D maps. However, traditional surveying and mapping techniques are inadequate as they are labor intensive and cost inefficient. Mobile Laser Scanning (MLS) systems, which combine Light Detection and Ranging (LiDAR) with navigation techniques, are able to acquire highly accurate 3D measurements of road environments.

MLS is used more and more for a wide range of applications in recent years, from road surface inspection to roadside object recognition and 3D digital city construction. MLS is able to efficiently collect highly accurate 3D point cloud data of road environments, for example one million points per second at centimeter level accuracy. The collected 3D point cloud data consist of geometric and intensity information and is huge in size, i.e. 500 GB of data is easily acquired during two hours of scanning. In addition, the often high complexity and variation of road environments makes the extraction of information from such 3D point cloud data challenging, notably at an operational level.

In this thesis, first a brief introduction to the principle of laser scanning and existing mobile laser scanning systems is provided. Consecutively, an overview of the major data processing challenges of MLS point cloud is given. Next, efficient spatial data structures for organizing 3D point clouds are introduced, notably kd-trees, voxels and octrees. Their application on spatial partitioning and neighbourhood searching for 3D point clouds are provided as well. Geometric information of mountain road and urban road environments, such as terrain volume, water flow directions, locations and shape of individual trees, lamp poles and traffic signs, were studied and analyzed. Next, algorithms to extract geometric information from MLS point cloud data of road environments are designed and presented.

In mountain road environments, in case of road widening and road maintenance after hazards, excessive material needs to be removed. An algorithm to estimate such excavation volume is presented based on MLS scanned point clouds. A 2.5D grid based method involving gradients and normals is presented to estimate water flow to indicate the most likely location of damage caused by rainfall.

In urban scenarios, road side objects such as trees and road furniture are considered. Firstly, the huge point clouds collected by MLS systems are organized by either 3D voxels or octrees. Voxels enable efficient processing compared to using individual discrete points. To identify individual road side trees, a 3D voxel adjacency based method, named VoxTree, is proposed. For road side furniture, a work-flow consisting of re-tiling, filtering, voxelization and clustering is presented. To identify objects of interest, a 3D feature descriptor, named SigVox, is designed to operate at object level. Point clusters of training objects are selected and the SigVox descriptors of those objects are constructed as template. Then, recognition is performed by evaluating the distance between the template object descriptors and descriptors of candidate point clusters.

Testing results are given to evaluate the reliability of the proposed methodology.

Overall, the work in this thesis contributes considerably to the automatic processing of large mobile mapping data sets, by designing, implementing and validating new algorithms that are notably able to extract geometric information at a near-operational level.

Samenvatting

Nauwkeurige en recente digitale 2D- en 3D-kaartinformatie is steeds meer nodig door bijvoorbeeld de opkomst van intelligente transport systemen zoals zelfrijdende auto's en het toenemende gebruik van virtuele 3D informatie door bedrijven en overheden. Echter, traditionele technieken uit het landmeten en cartografie voor het vergaren van zulke informatie voldoen niet langer, omdat ze arbeidsintensief en mede daarom duur zijn. Mobiele Laser Scan (MLS) systemen, die, bijvoorbeeld vanuit een auto, LIDAR afstandsmetingen combineren met navigatie- en positionerings-technieken, maken het sinds een paar jaar mogelijk om in korte tijd veel en nauwkeurige 3D data punten te verkrijgen. Tezamen geven de vele 3D data punten een gedetailleerd beeld van de 3D omgeving van een weg.

MLS data wordt meer en meer gebruikt voor een breed scala van toepassingen, van inspectie van de staat van het wegdek en het identificeren van objecten rond de weg tot een volledige 3D virtuele stadsreconstructie. MLS is zeer efficiënt in het verzamelen van zogenaamde 3D puntenwolken, een miljoen punten kan bijvoorbeeld in een seconde worden ingemeten met een nauwkeurigheid in de order van centimeters. Behalve 3D informatie bevatten de puntenwolken voor elk punt vaak ook een intensiteitswaarde, een maat voor de sterkte van het ontvangen signaal, en vaak ook kleurinformatie, die wordt verkregen door foto's over de puntenwolk heen te leggen. De efficiëntie waarmee al deze data verkregen wordt, resulteert wel in zeer grote data sets, bijvoorbeeld 2 uur scannen kan zo 500 GB aan data opleveren. Daarnaast is de 3D omgeving van wegen vaak geometrisch zeer complex. Deze complexiteit in combinatie met de grote data volumes maakt het inwinnen van bruikbare informatie uit 3D puntenwolken zeer uitdagend, zeker op operationeel niveau.

Dit proefschrift geeft eerst een inleiding waarin het principe van laser scanning wordt uitgelegd, en waarin een overzicht wordt gegeven van de verschillende componenten van een Mobiel Laser Scansysteem. Vervolgend wordt een overzicht gegeven van de belangrijkste uitdagingen bij het verwerken van MLS puntenwolken en van data structuren die gebruikt kunnen worden voor het efficiënt organiseren van 3D puntenwolken op een computer, zoals zogenaamde kd-trees, voxels en octrees. Met name wordt uitgelegd hoe zulke structuren gebruikt kunnen worden om puntenwolken ruimtelijk op te delen en om punten dicht bij een gegeven punt snel te kunnen identificeren.

In bergachtige gebieden moet vaak overtollig materiaal worden afgevoerd als een weg wordt verbreedt of getroffen door een aardverschuiving. Gegeven een puntenwolk van zo'n weg en de directe omgeving presenteer ik een algoritme dat de hoeveelheid te verwijderen materiaal schat. Water dat in grote hoeveelheden op zo'n weg terecht komt, kan de weg beschadigen. Daarom presenteer ik ook een methode die schat waar water precies de weg op stroomt door de helling en hellingsrichting systematische te evalueren aan de hand van een 2D raster van de hoogte van de weg en zijn directe omgeving.

In meer stedelijke omgevingen beschouw ik verschillende soorten objecten die direct naast wegen voor komen, zoals bomen, lantaarnpalen en verkeersborden. Eerst wordt de grote MLS puntenwolk georganiseerd door middel van voxels of een octree. Het is in principe veel efficiënter en vaak voldoende om een klein aantal voxels te verwerken dan de vele individuele punten in een puntenwolk. Om elke boom langs de weg te identificeren, introduceer ik een methode, VoxTree genoemd, die voxels op een systematische manier afloopt en toekent aan een bepaalde boom, door na te gaan aan welke bomen eerder bezochte voxels in de directe omgeving zijn toegekend.

Voor het identificeren en analyseren van straatmeubilair presenteer ik een stappenplan, bestaande uit achtereenvolgens opdelen in stukken weg, scheiden van punten op en boven het terrein, verdelen van punten boven het terrein over voxels, en groeperen van voxels in samenhangende clusters. Om vervolgens objecten ook daadwerkelijk te kunnen identificeren, heb ik een zogenaamde 3D object beschrijver ontworpen, die ik SigVox heb genoemd, en die toegepast kan worden om volledige objecten zoals lantaarnpalen en verkeersborden te beschrijven. Eerst wordt de SigVox beschrijver van trainingsobjecten bepaald en vervolgens wordt nagegaan of clusters een soortgelijke beschrijver hebben. Toepassing van deze methode op een MLS puntenwolk van de TU Delft campus laat zien dat deze methode meer dan 90

Over het geheel genomen draagt het werk in dit proefschrift aanzienlijk bij aan de mogelijkheden voor het automatisch verwerken van grote MLS puntenwolken, door het ontwerpen, implementeren en valideren van verschillende nieuwe algoritmes die met name geometrische informatie kunnen extraheren op een bijna operationeel niveau.

1
Introduction

1

1.1. Laser scanning

Complete and up to date city and urban spatial inventories, containing road geometry, buildings, roadside trees, street lamp poles and traffic signs, are essential to society in economic and practical aspects. Conventionally, the 3D coordinates of features and objects are surveyed by theodolites, total stations and leveling (Anonymous, 1957). However, those traditional surveying techniques either need to have direct access to the objects of interest or have limited measuring distance. Since the 1970s, the availability of Global Positioning Systems (GPS) made it possible to acquire 3D coordinates of objects more efficiently (McNeff, 2002). However, GPS has several limitations, such as low accuracy under circumstances and unstable signal. Especially the 3D point acquisition is approximately one point per second, thus its is less suitable for dense surveying in city and urban scenarios. At the same time, in recent years, high precision urban maps are extensively required for various applications, such as smart cities (Nebiker et al., 2010; Batty et al., 2012), autonomous driving (Li et al., 2004; Schreiber et al., 2013) and intelligent transportation systems (Bishop, 2000; Agamennoni et al., 2011; Ivan et al., 2015). Efficient and frequent updating of urban inventories are crucial to ensure the overall technical and social function of a community. However, the traditional approaches of spatial data acquisition are too labor intensive and too inefficient to meet the increasing demand of high precision and short temporal surveying.

In the last two decades, advances in the fields of solid-state electronics, photonics and computer science have made it possible to construct reliable, high resolution and accurate laser scanning systems (Vosselman and Maas, 2010). Laser scanning applies the Light Detection And Ranging (LiDAR) technique. In 1977, the National Aeronautics and Space Administration (NASA) developed a four-wavelength airborne oceanography LiDAR system. By modeling the laser energy that was transmitted from the laser emitter and reflected back to the detector, the quantification of chlorophyll concentration and other biological and chemical substances in algae was estimated (Browell and Center., 1977). Then by combining the range with Global Navigation Satellite Systems (GNSS) and orientation from Inertial Navigation System (INS), the 3D coordinates of the objects' surface illuminated by the laser light are derived (Vosselman and Maas, 2010). The first modern instruments were built in the early 1990s. In 1993, the first prototype of a commercial airborne laser scanning dedicated to topographic mapping was available (Bufton, 1989; Flood and Guteliue, 1997). Laser scanning technique has benefited from the progress of GNSS and INS for precise position and orientation measurements (King, 1998).

A typical laser scanning system consists of (i) a laser ranging unit, (ii) an opto-mechanical scanner, (iii) a position and orientation measuring unit, and (iv) a control, processing and recording unit (Wehr and Lohr, 1999). The laser ranging unit consists typically of a transmitter, a receiver and the optics for both. The basic principle of LiDAR is to use a laser to illuminate an object and then a photodiode to record the back-scattered radiation (Schawlow and Townes, 1958; Wehr and Lohr, 1999). The range is then determined by making use of the fact that speed of light can assumed to be constant. The receiver optics collects the back-scattered laser light and focuses it onto a photodiode detector, which converts the incident power of photons to electrical

pulses. The position of the laser ranging unit is determined by the observations from GNSS. The orientation of the unit is calculated from the pitch, roll and heading angles around the three axis of the navigation frame, and are measured by an on-board Initial Measurement Unit (IMU) (Baltsavias, 1999). The output of the laser scanner is then a geo-referenced three-dimensional point cloud of LiDAR measurement, representing the geometry of targets, complemented with reflected intensity and possibly waveform information of the returned light (Wehr and Lohr, 1999; Hyyppä et al., 2008).

Laser scanning systems sample up to millions of highly accurate points per second. This makes the acquisition of highly accurate 3D coordinates very efficient (Nelson et al., 1984; Scheier et al., 1985). Profiling LiDAR was widely used for bathymetry, forestry and other applications in the 1970s and 1980s, which established the basic principles of using laser for remote sensing purposes (Clarke et al., 1970; Menenti and Ritchie, 1994; Winker et al., 1996; Liebowitz, 2002; Xiaoli et al., 2013). Within a period of only two decades, LS has been applied to mapping topography, modeling infrastructure, and other areas or objects of interests (Kraus and Pfeifer, 1998; Garvin et al., 1998; Maas and Vosselman, 1999; Hyyppä et al., 2001; Vosselman and Maas, 2010).

1.2. Laser scanning systems

By considering the platform on which the LiDAR system is mounted, laser scanning systems are categorized into four main classes, which are Satellite-borne Laser Scanning (SLS), Airborne laser scanning (ALS), Mobile Laser scanning (MLS) and Terrestrial Laser Scanning (TLS). Since these systems have different measurement principles and different scanning orientations towards objects, the data sets are also differs dramatically. Figure 1.1 shows the aforementioned four types of LiDAR systems and examples of corresponding acquired data sets.

Figure 1.1a shows the ICESat system and a sample point cloud of the globe. As can be seen, SLS has sparse point density, which is 170 meters along track direction but in the order of tens of kilometers across track (Zwally et al., 2002). SLS point clouds have been applied in large scale investigations but are too sparse for urban environment monitoring and mapping.

Point clouds obtained from ALS, as shown in Figure 1.1b, have higher density and provide highly accurate 3D coordinates. Operative ALS data typically provide point densities of 0.5-10 pts/m^2 and beam footprint sizes of a few decimeters and larger (typically 0.3-10 $mrad$). Using this technique, ground topography and building roofs can be captured. Numerous studies have been carried out on different applications and a variety of achievements are obtained. Examples are generating Digital Elevation Models (DEM) and Digital Terrain Models (DTM) (Vosselman, 2000; Pfeifer, 2001; Briese et al., 2002), 3D building reconstruction (Maas and Vosselman, 1999; Vosselman and Dijkman, 2001; Rutzinger et al., 2009; Xiong et al., 2014), forest inventory and parameter estimation (Hyyppä et al., 2001; Persson et al., 2002; Gorte and Pfeifer, 2004; Morsdorf et al., 2006), land cover classification (Rutzinger et al., 2008; Guo et al., 2011; Mallet et al., 2011) and so on. However, the illuminating direction of airborne laser pulses makes it less likely to obtain detailed facades of buildings, road surface of narrow streets and trunks of trees. This will introduce errors when generating full 3D urban maps. Furthermore, only a few points can be obtained for street infrastructure like

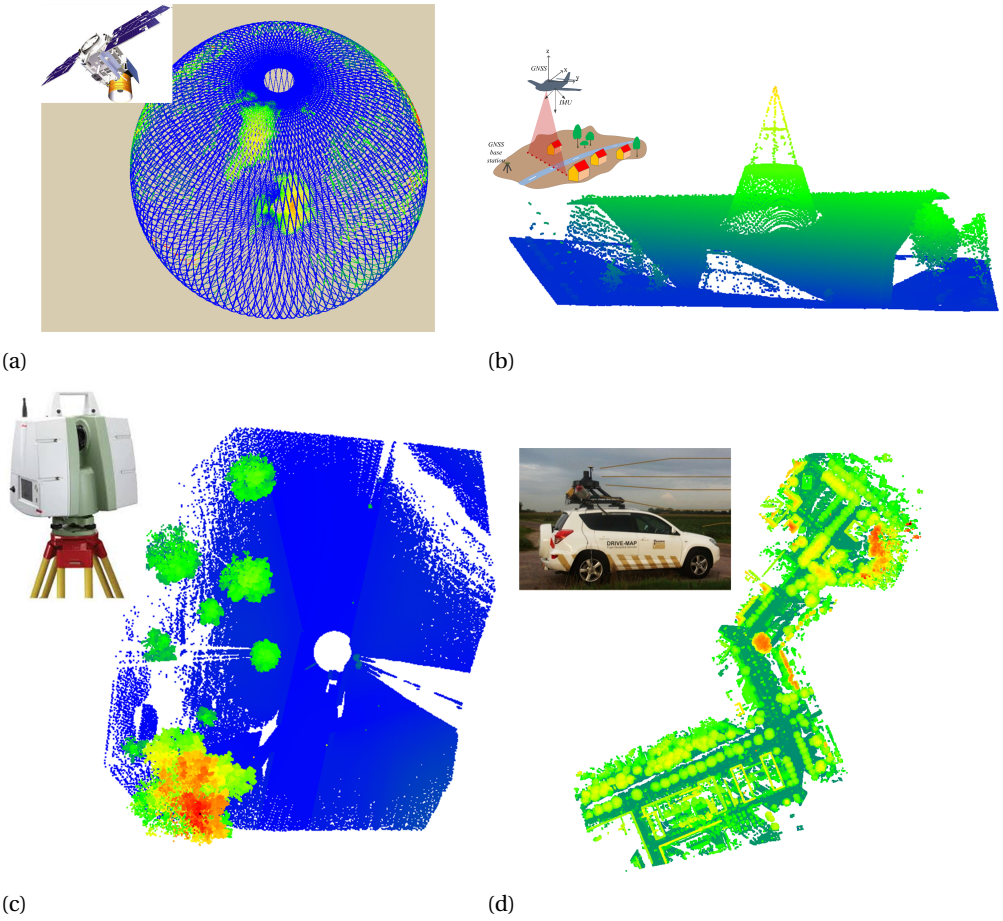


Figure 1.1: Different LiDAR systems and related sample point cloud data sets. (a) GLAS SLS and a global point cloud. (b) An ALS system and sample ALS point cloud data of TU Delft library. (c) Leica C10 TLS scanner and a sample TLS point cloud. (d) Fugro Drive-Map MLS system and sample urban MLS point cloud data.

street lamp poles, traffic signs and traffic lights. This makes it difficult to identify and monitor those elements using ALS point clouds.

TLS systems, however, obtain detailed 3D point clouds of object of interest from a stationary tripod, as illustrated in Figure 1.1c. Stationary TLS sensors have a narrow beam width, wide field of view (FOV) and more precise ranging. The most prominent advantage of TLS is its high point density, which can be as high as tens of thousands of points per square meter. Since TLS sensors are able to scan at a close distance from objects, high density point clouds of the scanned objects can be acquired. This enables to reveal the details of objects (Idrees and Pradhan, 2016). TLS point clouds have been extensively used for documentation of cultural heritage (Fröhlich and Mettenleiter, 2004; Haddad, 2011; Fregonese et al., 2013), change detection (Monserrat and Crosetto, 2008; Zogg and Ingensand, 2008; Abellán et al., 2010; Hohenthal et al., 2011), tunnel inspec-

tion (Van Gosliga et al., 2006), 3D reconstruction (Pu and Vosselman, 2006, 2009; Li et al., 2010b) and modeling (Aschoff et al., 2004; Henning and Radtke, 2006; Brenner, 2005; Liang and Hyyppä, 2013). Despite of those various achievements, TLS systems scan objects from a fixed tripod and have a limited scanning range. Thus TLS systems are less agile and less efficient for large area data acquisition.

The emergency of MLS systems overcomes the gap between the low point density of SLS and ALS systems and the low agility of TLS systems. MLS produce point clouds that have similar point density as TLS but of much larger areas. As illustrated in Figure 1.1d, a LiDAR system is mounted on a vehicle and is continuously profiling the road environment across the driving direction. With on-board highly accurate Positioning and Orientation Systems (POS), a MLS system provides a huge improvement in effective spatial coverage compared to a TLS system and obtains high precision point cloud data Zhao and Shibasaki (2003). Furthermore, objects invisible to ALS, such as building facades and tree trunks, can be scanned by a MLS system. Those properties make MLS especially attractive for road environment scenarios where high resolution surface data is required (Barber et al., 2008; Puente et al., 2013a). In recent years, various MLS systems have been developed. A detailed review of these systems is found in (Puente et al., 2013a). Taking the advantages of MLS point clouds, versatile geometric information of road environment can be extracted, i.e. road surface geometry, roadside trees, lamp poles, traffic signs and traffic lights. So far, MLS point clouds have been applied for extracting urban road surface geometry (Jaakkola et al., 2008; Brenner, 2009; Guan et al., 2015), inventory of road infrastructures (Lehtomäki et al., 2010b; Pu et al., 2011; Yang et al., 2013), road markings extraction (Kumar et al., 2014; Guan et al., 2014), road furniture extraction (Lehtomäki et al., 2010a; Cabo et al., 2014; Yang et al., 2015) and roadside tree extraction and monitoring (Rahman et al., 2009; Rutzinger et al., 2010; Li et al., 2012; Vega et al., 2014).

1.3. Problem statement and processing challenges

Despite of the aforementioned achievements, a problem among many existing methods is that they either extract only a specific type of objects, or perform only on case study level. Nevertheless, there is a large variety of city and urban road furniture that are of crucial interest for inventory, monitoring and management. Meanwhile, the volume of acquired MLS point clouds of urban road environment grows dramatically and most of the existing methods are not scalable which makes the processing unfeasible regarding computational efforts and output quality. Thus, there is a strong need to develop a scalable methodology to extract more detailed geometric information efficiently from huge MLS point clouds.

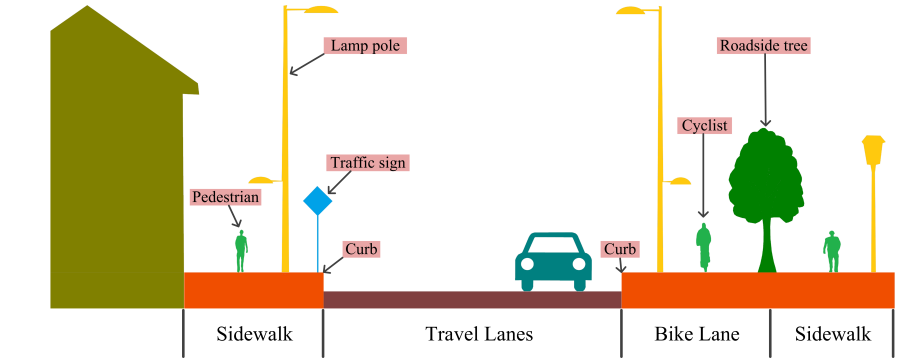
To effectively extract geometric road environment information, approaches based on MLS systems and the acquired point clouds can profit from the resulting large and highly accurate point clouds. Nevertheless, there are several problems in the application of MLS and the acquired point clouds for road environment assessment:

1. The road environment is highly complex.

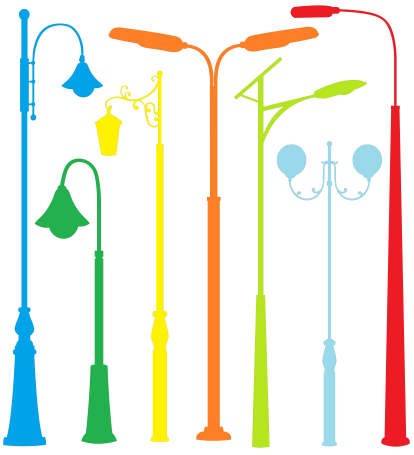
   The urban road itself consists of three major parts, i.e. road structure, road surface and roadside furniture (Turner, 2007). As illustrated in Figure 1.2a and Fig-

ure 1.2b, there are many types of components for each part of urban roads. These factors contribute to the overall complexity of the geometry of the road environment and make the extraction of geometric information on the road environment complicated.

In mountain road environments, excavation is inevitable in engineering works related to road widening and maintenance, compare Figure 1.2c. The terrain complexity makes the estimation of excavation volume difficult. Furthermore, erosion due to water flow is a major cause for road damage, as illustrated in Figure 1.2d. MLS and the consecutive point cloud processing could provide an efficient method for damage assessment and road work preparation in mountain road engineering.



(a)



(b)

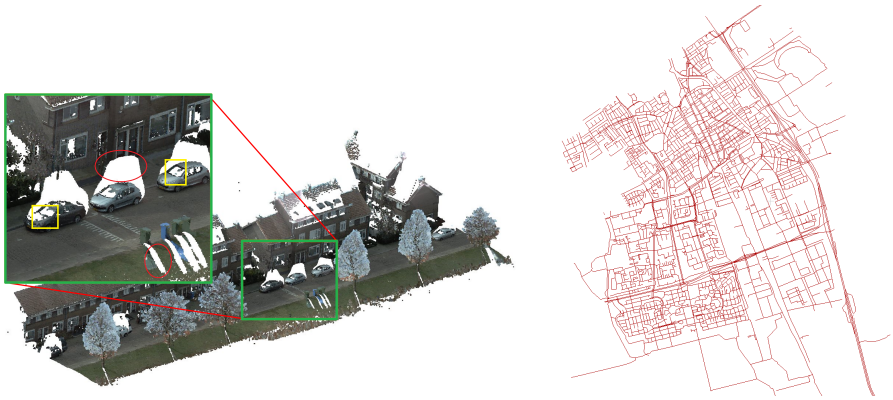
(c)

(d)

Figure 1.2: Road environment is highly complex. (a) Cross section of a urban road. The road environment consists of many road elements, like road surface, pedestrians, vehicles, lamp poles, traffic signs and roadside trees. (b) Different types of roadside lamp poles. (c) Mountain road excavation is inevitable in case of widening or maintenance after landslide. (d) Road damage caused by rain water erosion.

2. The acquired data sets are huge.

   Figure 1.3a shows a point cloud of a stretch of a 100 meter long road. It consists of 2,421,748 points and file size in LAS format is 102.8 MB. When considering roads at city level, the size of the acquired data sets will be huge. Figure 1.3b for example shows the complete road network of Delft city, the Netherlands. The total length of urban road in Delft reachable for a MLS system is 397,544.66 m. The estimated corresponding number of points is almost 10 billion and the size of the point cloud in LAS format is approximately 400GB. In addition, video and panorama images are collected as well. This is too large for a normal computer to store, operate and process in a reasonable time.



(a) (b)

Figure 1.3: An example of MLS point clouds and the urban road network of Delft city, the Netherlands. (a) A MLS point cloud sampling a urban scenario. (b) A top view of urban road network of Delft city.

3. Point clouds have large variations.

   Because of the scanning mechanism of MLS systems and the complexity of the road environment, the obtained MLS point clouds have varying quality in terms of point density, noise level, data gaps and outliers. The gaps in the point cloud data in Figure 1.3a are caused by occlusions and reflections, as indicated by the red circles and yellow rectangles, respectively. Due to the scanning geometry and the principle of the MLS system, the point density varies with respect to the objects' distance. The higher the point density on an object, the more details of the object is captured. Also, the noise level of the points is influenced by the laser incidence angle, scanning range and different reflecting materials. The more skewed the incoming ray is compared to the objects surface normal, the weaker the signal level (Soudarissanane et al., 2011). In addition, the further the objects are from the scanner, the lower of the accuracy of the obtained points (Vosselman and Maas, 2010; Puente et al., 2013a).

4. Lack of scalable methods for object identification.

There are methods available for object recognition in MLS point clouds (Cabo et al., 2014; Yang et al., 2015; Guan et al., 2015; Teo and Chiu, 2015). However, those methods are unable to identify specific objects of interest in the road environment but only roughly classify objects, for example pole-like objects are categorized as one class. Furthermore, those methods are in general not scalable which means that users cannot easily make a balance between computational efforts and the quality of output.

1.4. Research objectives

The main research question of this thesis reads:

How to efficiently extract geometric information on the road environment from huge MLS point clouds?

Based on the main research question, the following sub-questions are derived:

1. What is the current status of MLS systems and the data processing work-flow?
2. How to efficiently organize huge MLS point clouds?
3. How to estimate water flow directions and excavation volume on mountain roads from MLS point clouds?
4. How to individualize roadside trees from MLS point clouds?
5. How to automatically identify objects in the urban road environment in MLS point clouds?

1.5. Scope and limitations

To narrow down the research scope, some aspects concerning MLS systems and the acquired point clouds will not be considered. First, the calibration, including bore-sight alignment and intensity calibration, will not be considered. Second, the procedure of direct geo-referencing is not considered in this study. Third, the accuracy of the acquired point clouds will not be validated with regard to ground truth.

1.6. Organization of the thesis

This chapter briefly reviewed the background of the study, the problems and the research objectives of this study. The main research question is formulated and subsequently divided into five sub-questions. The sub-questions will be tackled in the next five chapters.

Chapter 2 provides an overview of MLS systems and their components, which is related to the first sub-question. Then a typical workflow for MLS point cloud processing is reviewed as well as current MLS applications.

Chapter 3 introduces two data structures that are used in this study for re-sampling high density MLS point clouds, which are voxels and octrees. The contents consists

of neighborhood searching strategies and their applications respectively. This chapter considers the second sub-question.

Chapter 4 to 6 are written based on manuscripts that have been published in or submitted to scientific journals.

Chapter 4 is linked to sub-question 3 and introduced the application of MLS scanned point clouds from mountain road engineering, here applied on excavation volume estimation and water flow direction determination.

Chapter 5 focuses on contiguous roadside tree individualization, which corresponds to sub-question 4.

Chapter 6 is related to sub-question 5 and introduces a 3D feature descriptor that enables robust recognition of roadside objects.

Conclusions and recommendations for future research are given in Chapter 7.

2
Mobile Laser Scanning

This chapter first introduces the main components of a typical MLS system. Then, some MLS systems that operate in different environment are discussed. The last part reviews the state-of-the-art of data processing of point clouds acquired by MLS systems and the subsequent applications.

2

2.1. Mobile laser scanning system

MLS is the most recently developed type of laser scanning system, which uses similar theories and instruments as ALS and TLS. MLS is defined as the technique that deploys LiDAR systems on road, rail or marine vehicles to efficiently acquire 3D geo-referenced point clouds. The basic components of a MLS system are a laser scanner, a Position and Orientation System (POS) and a platform. The laser scanner records coordinates for points on the surfaces of surrounding objects. These coordinates are in an arbitrary system relative to the scanner itself. The POS consists of a GNSS and an INS, which provides position in a global and body system respectively. Data from those three instruments are integrated and result in a globally geo-referenced point clouds.

2.1.1. Laser scanner

The laser scanner, or LiDAR instrument, is the piece of equipment that is actually imaging the surrounding environment. As mentioned in Chapter 1, data points from object surfaces are obtained in the scanner's local coordinate system. These points are measured using a range and angle from the scanner's origin. The scanning instrument uses a laser to measure distance to an object.

There are two typical approaches that a laser ranger uses to determine distance: (i) time-of-flight (ToF) and continuous-wave (CW). A ToF is a pulsed instrument determining the precise time for a short pulse of emitted electromagnetic radiation (EMR) to travel from the scanner to an object and scatter back to the scanner. As shown in Figure 2.1, the laser pulse transmitter emits a laser pulse and the pulse is scattered back to the receiver. The total time of travel is recorded to determine the range between object and scanner.

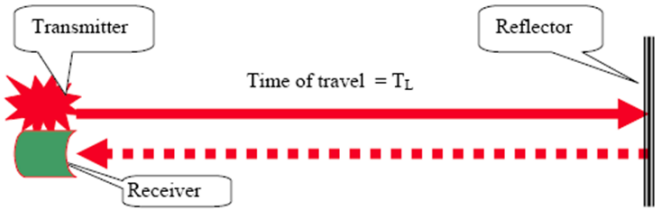


Figure 2.1: Principle of a time-of-flight laser ranger

From the calculated time, a range is determined by Equation 2.1.

$$R = \frac{ct}{2} \tag{2.1}$$

Here $c = 299,792,458\ m/s$ is the known speed of light in vacuum and t is the total travel time of the laser pulse.

A CW, or phase-based, instrument emits a continuous EM wave of precisely known amplitude and determines the precise phase difference between the emitted and returned pulse. These instruments are often more accurate than ToF instruments but have shorter ranging capability. This category of instrument will not be reviewed further, as it was not used in this research.

2.1.2. Positioning and Orientation System

The POS provides the position and orientation information needed to reference the moving platform to a local and global coordinate system. The POS consists of GNSS and IMU, the GNSS determines the location of the MLS system using satellite positioning while the IMU determines its orientation.

An IMU contains inertial sensors called gyroscope and accelerometers. Gyroscopes measure the rate of angular change of a system with respect to inertial space. Accelerometers record the specific force applied to a system (Crassidis, 2006). As the mapping platform moves, it is subject to rotational and specific forces from platform motion. The IMU provides orientation in roll (ω), pitch (φ), and heading (κ) using the gyroscopes and accelerometers. The orientation, or attitude, values are the rotations about the $x-$, $y-$, and $z-$ axes of the mapping platform. Figure 2.2 is a MLS system from Fugro and Figure 2.3 is a close up view of the sensors. There are two laser profilers mounted on the two sides of the system respectively. The points obtained can also be colored by images collected by panoramic cameras.

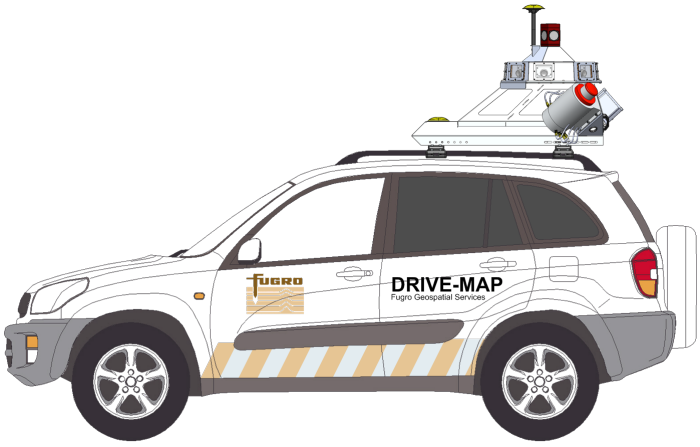


Figure 2.2: A mobile laser scanning system (Image source: Fugro GeoServices B.V.).

2.1.3. Other MLS systems

Besides of mounting on a vehicle, the system can also be mounted on a boat, as illustrated in Figure 2.4. The same system as in Figure 2.2 is mounted on a boat in Fig-

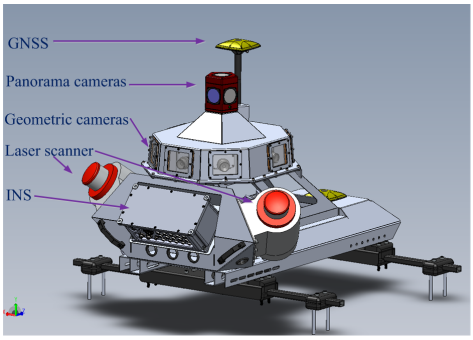
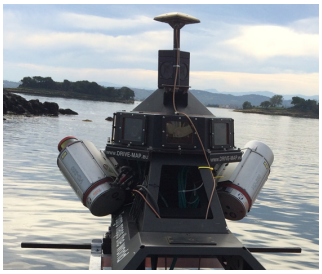
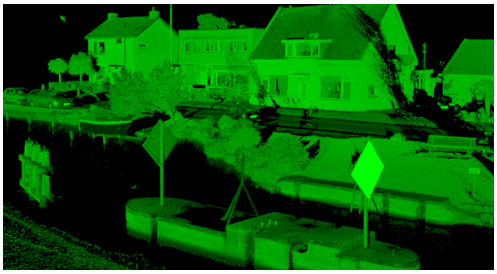


Figure 2.3: Components of a mobile laser scanner.

ure 2.4a and Figure 2.4b is a sample point cloud data collected by the system.



(a)



(b)

Figure 2.4: Boat-Map of Fugro and sampled point cloud. (a) The Boat-Map MLS system mounted on a boat. (b) Point cloud data collected by Boat-Map.

Also, the system can be mounted on a train to scan the environment of a railway. Figure 2.5 shows a MLS system mounted on a train and Figure 2.5b is the collected point cloud data.

In 2012, Kukko et al. developed a Backpack MLS system to enable the access to places where vehicles cannot, such as wetland and dense forest (Kukko et al., 2012). Figure 2.6a is the Backpack MLS system in operation. Figure 2.6b is a sampled point cloud and the scanning trajectory respectively.

2.2. Data processing work flow

This section describes a typical work-flow of processing a point cloud obtained by laser scanning, which often consists of filtering, segmentation, and some particular application.

2.2.1. Filtering

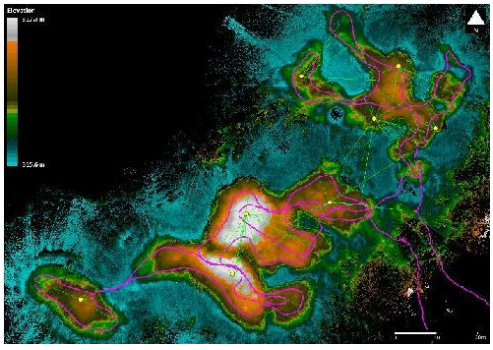
Filtering is the procedure of extracting ground points from the original LiDAR point cloud (Sithole and Vosselman, 2004). Figure 2.7 shows a point cloud sampling an urban

(a)

(b)

Figure 2.5: Rail-Map and the obtained point cloud data. (a) A MLS system mounted on a train. (b) Point cloud data collected by Rail-Map. (Image source: Fugro GeoServices B.V.)

(a)



(b)

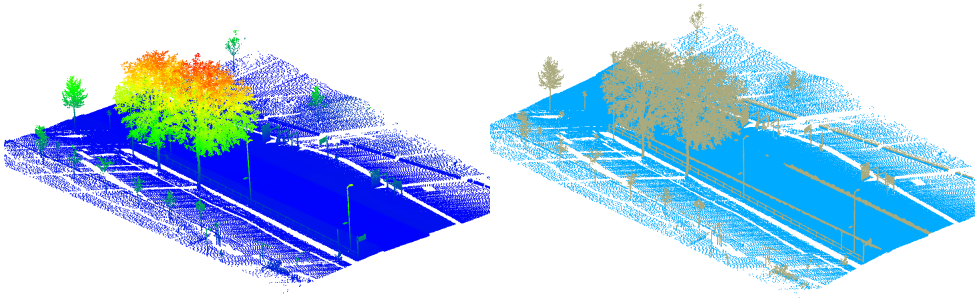
Figure 2.6: Backpack MLS system, and a sample point cloud and scanning trajectory (Kukko et al., 2012). (a) Operating of the Backpack MLS system. (b) Obtained point clouds of the Backpack MLS system and the scanning trajectory (purple line). (Image source: (Kukko et al., 2012))

scenario. Figure 2.7a is the original point cloud and after filtering, the ground points and non-ground points are separated in Figure 2.7b.

Several types of filtering methods have been proposed. Based on the filter strategy applied, these algorithms can be grouped into two major categories (Hu et al., 2014; Li, 2013; Meng et al., 2010): (i) interpolation based methods (Kraus and Pfeifer, 1998; Axelsson, 2000; Evans and Hudak, 2007; Mongus and Zalik, 2012; Chen et al., 2007). (ii) morphological based methods (Zhang et al., 2003; Chen et al., 2007).

Interpolation based filters

For the interpolation-based filters, initial ground points are selected and used to iteratively create a provisional surface that gradually approaches the final approximated ground surface (Hu et al., 2014). Kraus and Pfeifer proposed a method to approximate ground iteratively by using weighted linear least squares interpolation (Kraus and



(a) (b)

Figure 2.7: Filtering of LiDAR point cloud data. (a) Original point cloud data (before filtering). (b) Ground and non-ground points are segmented after filtering.

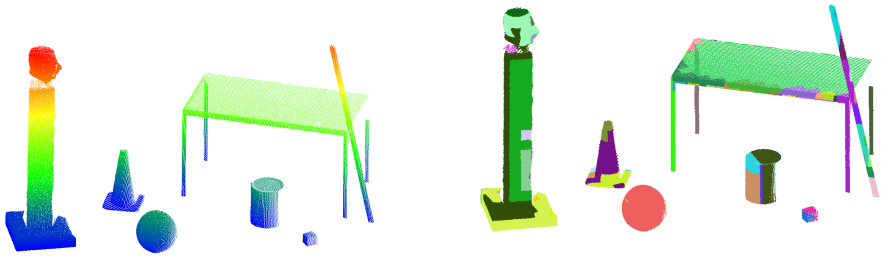
Pfeifer, 1998). In this method, ground points usually have negative residuals while non-ground points have positive residuals. This is used to assign high weight to the points with negative residuals. Pfeifer further extended the algorithm by hierarchical interpolation to improve both on the filter results and on the computational efficiency (Pfeifer, 2001). Continuing the previous work, Lee and Younan used a normalized least squares method to replace the least squares method. This implementation did further improve the filtering results (Lee and Younan, 2003). However, a disadvantage of the method is that a user has to set three thresholds. Axelsson proposed a ground-breaking filtering method for ALS point clouds based on refining the weight assignment using a Triangulated Irregular Network (TIN) Axelsson (2000). Also, an interpolation method based on a Thin Plate Spline (TPS) was presented to filter ALS point clouds (Evans and Hudak, 2007). The method uses a regularly gridded DEM rather than a TIN. Furthermore, a TPS interpolation-based parameter-free filter was developed (Mongus and Zalik, 2012). This method first builds several point clouds of different resolution levels from the considered point clouds and then a surface is interpolated iteratively from the coarse level to the finest level.

Morphological based filters

The concept of morphological filtering is to approximate ground points using morphological operations, notably by opening (Zhang et al., 2003; Chen et al., 2007). When there are adequate laser pulses that reach the ground, by using morphological opening with a small window, ground points can be identified. However, when there are not many pulses illuminating the ground, the window size for morphological opening has to be tuned according to the size of the objects (Li et al., 2013; Mongus et al., 2014). The morphological filtering methods are easy to implement, however, big window sizes will produce a surface with more protruded ground features (Sithole, 2001; Chen et al., 2007).

2.2.2. Segmentation

Point cloud segmentation refers to the procedure of partitioning a 3D point cloud into subsets that satisfy certain pre-defined criteria (Woo et al., 2002; Biosca and Lerma, 2008; Vosselman and Maas, 2010). Figure 2.8 illustrates the segmentation of a TLS point cloud based on planarity. Figure 2.8a and Figure 2.8b show a point cloud before and after segmentation respectively.



(a) (b)

Figure 2.8: Segmentation of LiDAR point cloud data. (a) Original point cloud data (before segmentation). (b) Point cloud after segmentation (based on different planes).

Segmentation methods can be roughly categorized into model fitting based methods (Vosselman and Dijkman, 2001; Schnabel et al., 2007), and region growing based methods (Tovari and Pfeifer, 2005; Vieira and Shimada, 2005; Vo et al., 2015).

Model fitting based methods

Model fitting based segmentation approaches are mainly conducted by fitting geometric models to the point cloud (Vosselman and Maas, 2010). Two widely known methods for estimating model parameters are Hough Transform (HT) (Ballard, 1981) and random sample consensus (RANSAC) (Fischler and Bolles, 1981). The HT is used to detect geometric primitives, such as lines, planes and spheres in both 2D and 3D (Vosselman et al., 2004; Tarsha-Kurdi et al., 2007; Rabbani and Heuvel, 2005). The RANSAC paradigm is applied to extract primitive shapes by randomly selecting minimal point sets to evaluate many candidate shape primitives. The method has been applied to segment 3D point clouds, e.g. for building facade segmentation Boulaassal et al. (2007), planes (Schnabel et al., 2007).

Region growing based methods

Region growing based point cloud segmentation mainly consists of two steps. First, a seed point or initial point segment needs to be selected. Next, the refinement of the segment grows iteratively (Vosselman and Maas, 2010). Triangle segments from a TIN are selected as seed surface and the angle and distance between the neighboring triangles for the region growing strategy (Gorte, 2002). Based on local planar fit as a criterion for selecting the seed region, an octree was employed to search neighboring points of

those points that are within a threshold are merged to the corresponding point segments. (Wang and Tseng, 2004; Douillard et al., 2011; Vo et al., 2015) presented an algorithm that first organizes a point cloud in an octree data structure. Then the voxels that belong to the same local plane are selected as seed voxels, before neighboring voxels are merged recursively to the seed clusters.

2.2.3. Applications

Since the emergence of the MLS systems, the technique has been applied in a variety of scenarios. This section reviews different applications of point clouds acquired by MLS.

Road elements inspection and inventory

In the road environment, there are lots of elements that need to be monitored and documented automatically for security and management purposes, such as roadside trees, lamp poles, traffic signs, road markings and curbstones. Such elements are traditionally measured and mapped manually. However, the point clouds obtained by MLS systems are sufficiently dense and accurate to extract the above urban road elements automatically and efficiently. In the remainder of this section, applications of MLS point clouds on the extraction of road markings and curbstones are given. A detailed review of roadside trees is given in Chapter 5, lamp poles and traffic signs will be discussed in Chapter 6.

- **Road and curbstone extraction**
  Roads play a crucial role in transportation and delivering services. Efficient road inspection and monitoring are necessary for safety purposes. In the past decade, methods have been developed for road segmentation and curbstone extraction from MLS scanned point clouds. In (Jaakkola et al., 2008), a method for automatic classifying road surface and curbstone points was presented and the testing results showed that 92.3% and 79.7% of the road points and curbstones were correctly identified. In (Boyko and Funkhouser, 2011), a method for segmenting road points from large scale unorganized 3D point clouds was presented. The method first fits a 2D active contour to an attractor function to predict the locations of curbs. Points lying within the active contour are labeled as road. Test results show that the method provides 86% correctness and 94% completeness. Luo et al. presented an algorithm to segment road road points using a patch-based framework. The method first build a 3D patch0based match graph structure from 3D point patches. Then, the errors occurred during transferring point patch labels to road surface are rectified using contextual information from Markov random fields (Luo et al., 2016). In (Miraliakbari et al., 2015), road surface was automatically extracted from geo-referenced mobile laser scanning data with region growing. The basic hypothesis of the method is that road surface is a continuous smooth and bounded by curbstones. Comparison to ground truth showed that the completeness and correctness of the method were 92% and 95% respectively.

- **Road markings extraction**
  The markings painted on road surfaces, for guidance of drivers, are highly im-

portant for the road safety. In recent years, the development of MLS and data processing techniques enabled the automatic extraction and recognition of road markings. In (Zhou and Vosselman, 2012), a three-step method for mapping curbstones was proposed, consisting of coarse curbstone detection, midpoint determination and connecting collinear line segments. The results illustrated that the completeness of road marking detection varies ranging from 54% and 83%. In (Kumar et al., 2014), an automated algorithm for road marking extraction from MLS data was proposed. The method employed a range dependent threshold function and was applied to the intensity of points. Then, binary morphological operations were applied to complete the shape of the road markings. The results showed that 80 out of 88 road markings were successfully extracted. In (Yang et al., 2012c; Guan et al., 2015), a method for automatic road marking extraction from MLS point clouds and images was presented. The three-step method first generates a geo-referenced feature image from the point cloud. Then, point density dependent thresholds were used to locate potential areas of road markings in the image. Finally, morphological operations were applied to extract road markings. The feasibility of the method was also verified. Yan et al. presented a scan line based method to extract road markings from point clouds (Yan et al., 2016). The method consists of three steps, i.e. pre-processing, road points extraction and road markings extraction and refinement. The evaluation results of the proposed method showed that the completeness and correctness were 0.96 and 0.93 respectively. Zhang et al. presented a robust algorithm for road pavement markings inspection from MLS point clouds (Zhang et al., 2016). The algorithm consists of three steps, i.e. pre-processing, extraction and classification. The testing results demonstrated that the proposed method can achieve 0.93 and 0.95 in completeness and correctness. These obtained road markings and lane information can then be documented for other purposes, such as digital city and autonomous driving (Levinson et al., 2011).

Railway monitoring

Railway maintenance requires regular inspection of railway environments, including railway tracks, overhead wires, signal poles and switches (Salvini et al., 2013; Rhayma et al., 2013). MLS systems sample the railway corridor and acquire high density point cloud data. Thus the elements of the railway environment can be automatically inspected and monitored (Hung et al., 2015). In (Elberink and Khoshelham, 2015), a method for extracting railway centerlines was presented. The method first generates center points in a data-driven manner by projecting rail track points to the parallel track and the midpoint initial center points. Consecutively, a piecewise linear function is fitted through the center points to obtain center points at a regular interval. Then, piecewise 3D track models are fitted to the rail track points. The results were compared with reference data and the accuracy of the position of the centerlines is 2-3 cm. In (Yang and Fang, 2014), an automatic method for railway track extraction from mobile point clouds was introduced. The method uses both geometric and intensity information of laser scanned point clouds to extract track points. Then, 3D track models are fitted to the obtained track points. Comparison with ground truth data shows that the method is able to identify railway tracks with an overall accuracy of 95%.

Highway and tunnel monitoring

The application of laser scanning for highway monitoring mainly focuses on the extraction of geometric information (Hatger and Brenner, 2003). During highway construction, the MLS scanned high density point clouds can be used to survey and investigate the construction progress (Gräfe, 2008). In the highway maintenance phase, the geometric information of highway roads is determined from MLS point clouds to monitor the roughness and damage of road surface and inspect the safety of the highway (Kim et al., 2008).

Stimulated by the highway construction, a large number of tunnels are drilled. Point clouds are applied in surveying and inspecting of those tunnels. Yoon et al. (Yoon et al., 2009) developed a laser-based scanning system for automatic tunnel inspection. Meanwhile, an algorithm for extraction damaged parts of the tunnel using geometric and radiometric features of the scanning data. In (Fekete et al., 2010), In (Boavida et al., 2012), a 25 km tunnel was scanned by a MLS system to validate the possibility of applying MLS point clouds for deformation inspection. The results showed that MLS shortens 15 times of on-site surveying time compared to TLS corresponding to a cost reduction of approximately 80%.

Environmental applications

Using the types of MLS systems mentioned in Section 2.1.3, other environmental applications are conducted as well. Bitenc et al. assessed the state of Dutch sandy coast using a land-based MLS system (Bitenc et al., 2011). The proposed method acquired a high quality DTM and concluded that the relative precision was 3 mm. Vaaja et al. mapped topography changes on a 58 km-long river using a MLS system mounted on a boat (Vaaja et al., 2011; Alho et al., 2011). The results showed the standard deviation of change mapping was 3.4 cm and mean square error (RMSE) of the generated DEM was 7.6 cm. Liang et al. demonstrated the Backpack MLS system's feasibility in estimating diameter at breast height (DBH) of forest trees with a RMS 14.63% (Liang et al., 2014). In (Ryding et al., 2015), point clouds obtained by a MLS system are applied to investigate forest trees, and DBH and stem positions were estimated. Change detection was conducted based on point clouds acquired by MLS point clouds both on urban street and forest area (Qin and Gruen, 2014; Xiao et al., 2015; Yu et al., 2004). Also, automatic urban accessibility diagnosis is conducted based on the point clouds obtained by MLS systems. For example in (Serna and Marcotegui, 2013), an automatic approach for urban accessibility analysis was presented. The method consists of two phases, i.e. urban object segmentation and curb detection. The test results showed that individual completeness were 82%.

2.3. Conclusions

In this chapter, the following research question was investigated and answered:

What is the current status of MLS systems and the data processing work-flow?

This chapter first introduced the MLS system and its components, which mainly consists of laser scanners, cameras, and position and orientation systems. The role of each component is discussed. Afterwards, four types of MLS systems are investigated and their data sets are also shown. Next a typical overall data processing work-flow of

MLS point clouds is discussed. The work-flow for different applications can be different, however, filtering and segmentation are common to most applications. Thus, the methods for conducting those two steps are reviewed. Finally, some potential applications of MLS point clouds are discussed. Despite of the available methods and applications for MLS data sets, there is not yet a standard work-flow for processing huge MLS point clouds. There are various types of objects on roadsides, moreover, these objects have complicated shapes and are always imperfectly sampled by MLS systems. Algorithms for effectively identifying those objects are highly demanded. What's more, the high efficiency of the data collection makes the MLS point clouds huge, thus, clever strategies and algorithms for storing and processing those data sets are needed.

3

Spatial data structures

The point clouds acquired by MLS systems consist of a huge number of discrete and apparent unorganized points. These huge number of points contain versatile information, i.e. 3D coordinates, intensity and colors. The size of the collected data set makes it hardly to store, inefficient to query and impossible to manipulate and operate on a normal computer. Therefore, efficient and agile spatial data structures are required. This chapter first reviews the state-of-the-art of widely used spatial data structures in point cloud processing. Then, two extensively used data structures in this thesis, i.e. *voxels* and *octrees* are introduced by discussing their components, spatial partitioning, neighborhood searching and their applications.

3

3.1. Introduction

As mentioned in Section 1.3, MLS systems collect point clouds of streets and surrounding objects at a speed of approximately 1 million points per second. Therefore, an acquisition of e.g. 1 hour, typically results in a point cloud of about 3.5 billion discrete points. It is difficult to load the entire resulting point clouds on a normal computer at one time. What's more, point cloud processing, i.e. visualization, classification, segmentation, feature extraction and data management, rely extensively on point querying, neighborhood searching and spatial sub-setting (Mitra et al., 2004; Vosselman and Maas, 2010; Holzer et al., 2012; Elseberg et al., 2013; Otepka et al., 2013). It is expensive to conduct those operations on huge unorganized discrete points. Thus, efficient spatial data structures and a scalable processing strategy are essential for successful processing huge MLS point clouds. A scalable processing strategy is achieved by either down-sampling the entire data or by subdividing the 3D spatial data in more manageable chunks. In the next sections, an overview of data structures used in point cloud processing is given. Several spatial data structures, i.e. Triangulated Irregular Networks (TIN), KD-trees and quadtrees, are introduced and their applications are reviewed. The complexity of those data structures is discussed with respect to construction, storage and point querying. A general overview of using complexity as a way to evaluate the efficiency of algorithms can be found in (Samet, 2006).

3.2. Spatial data structures

Spatial data sets consist of spatial objects composed of points, lines, surfaces, volumes, all containing geo-referenced coordinates, and sometimes even including temporal information (Frank, 1992; Samet, 1995). Spatial data sets represent roads, buildings, mountains, rivers, trees, etc. Moreover, the properties of the elements are also part of the spatial data sets. Spatial data structures enable efficient storage and manipulation of those spatial data sets (Papadias and Theodoridis, 1997; Anselin et al., 2006). Generally, spatial data sets are stored by explicitly defining a type of data structure that contains all the properties (Samet, 2006).

The point clouds collected by MLS systems can also be stored in the aforementioned common way. However, point clouds not only have huge numbers of points, but also the properties of each point are meaningful for processing. Moreover, the strategy of point organization is crucial for the efficiency of massive point manipulating operations (Lalonde et al., 2007; Wand et al., 2008; Elseberg et al., 2011; Beserra Gomes et al.,

2013; Richter and Döllner, 2014). Nevertheless, data structures also need to be memory efficient for practical purposes (Elseberg et al., 2013). During the last decade, many existing data structures have been used in point cloud processing. Hash tables (Wand et al., 2007; Nahangi et al., 2016; Yoshimura et al., 2016) and hierarchical grids (Börcs et al., 2015) were used to store feature points, B-trees were used for out-of-core point cloud storage and efficient indexing (Wu et al., 2010; Cura et al., 2015; Chrószcz et al., 2016), 3D R-trees were used for efficient management of large point clouds (Wang et al., 2009; Gong et al., 2012; Yang and Huang, 2014), TINs were used for segmentation and mesh generation (Zhang and Lin, 2013; Kang et al., 2014). However, those data structures were feasible for specific applications only and are not applicable for efficient point querying and scalable spatial subdivision. In general, the most widely used spatial data structure for point querying in large point clouds is the KD-tree and its modifications (Mount and Arya, 2010; Muja, 2011; Nuechter et al., 2016; Muja and Lowe, 2014). For scalable spatial partition and subdivision, quadtrees and octrees are used for 2D and 3D partitioning respectively. In the next sections, a more detailed review of the KD-tree, quadtree and octree data structures and their applications is given.

3.2.1. Triangulated irregular network

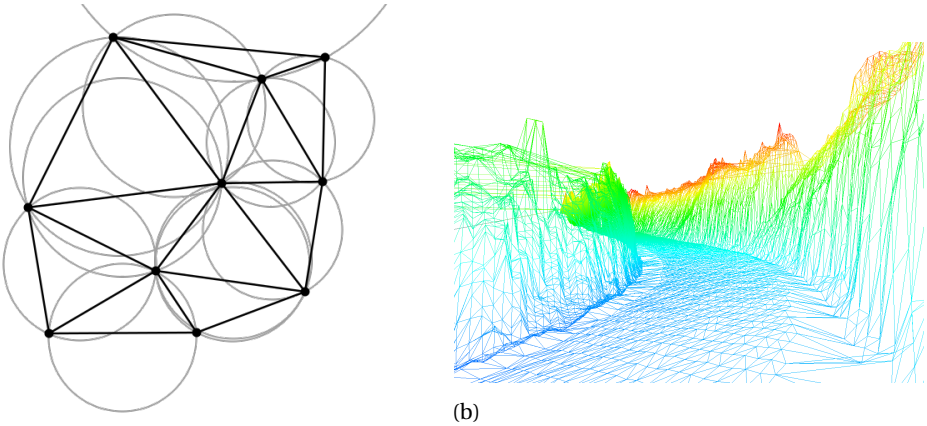
A Triangle Irregular Network (TIN) is a vector-based representation of surfaces, consists of irregularly distributed 3D nodes and lines that are arranged in a network of non-overlapping triangles (Peucker et al., 1978; Floriani and Magillo, 2009; Chen et al., 2011). In 3D point cloud processing, TINs are widely used for meshing geographic surfaces, such as DEM and DTM representation (Kraus and Pfeifer, 2001; Remondino, 2003; Ma, 2005; Xiaoye Liu, 2008) and geometric model reconstruction (Vosselman and Dijkman, 2001; Kwon et al., 2004; Teo et al., 2006; Abo-Akel et al., 2009; Ummenhofer and Brox, 2013). The most common way of TIN generation from 3D point clouds is via a 2D Delaunay triangulation of the 2D locations of the 3D points (Paul Chew, 1989; Tse et al., 2007; Wu et al., 2011; Chao et al., 2015).

In a Delaunay triangulation of a set of 2D points, the triangles are generated by the triples of points that contain no other point in their circumcircle (Okabe et al., 2009). Optimal algorithms for constructing a Delaunay triangulation of n points in the plane requires $O(n \log n)$ running time. A detailed definition of Delaunay triangulation is given in (Delaunay, 1934). Figure 3.1a is an example of Delaunay triangulation. Figure 3.1b is a TIN presentation of a mountain road generated from the point clouds acquired by a MLS system.

Notably, TINs are generated based on the 2D points obtained by the vertically projected 3D points. Thus, points that are far away in 3D may appear very close in the corresponding 2D Delaunay triangulation (Vosselman and Maas, 2010). It is possible to perform point querying using TINs, however, this is not straightforward as close by points are not simply searched with regard to Cartesian directions (Devllers2002 et al., 2002).

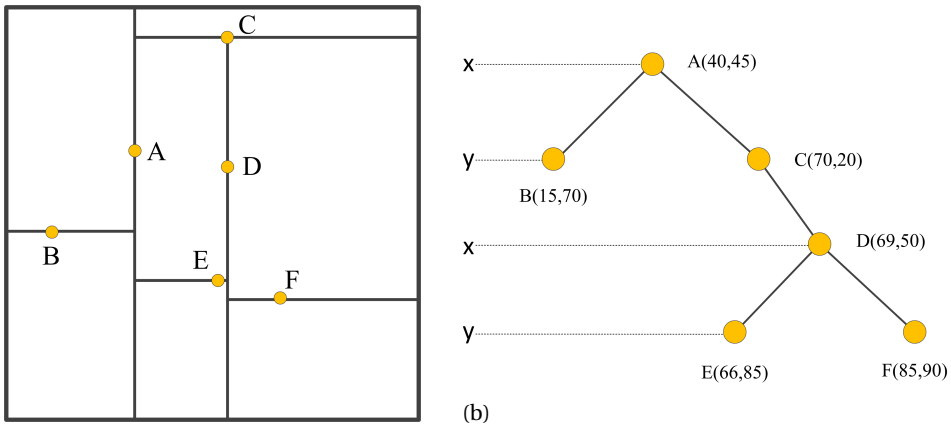
3.2.2. KD-tree and neighborhood searching

Currently, the so-called KD-tree, i.e. K-Dimensional tree, is the most widely used hierarchical data structure for nearest neighbor searching in 3D point cloud data (Bent-



(a) (b)

Figure 3.1: Delaunay triangles and a generated 3D TIN. (a) A Delaunay triangulation in the plane with circumcircles. (b) A 3D TIN generated from MLS point clouds.



(a) (b)

Figure 3.2: KD-tree spatial splitting of six 2D points, i.e. A(40,45), B(15,70), C(70,20), D(69,50), E(66,85), F(85,90), and the resulted tree data structure (Shaffer, 1998). (a) An example of 2D spatial splitting by a KD-tree. (b) The resulting binary tree data structure of the splitting.

ley, 1975; Samet, 2006; Sankaranarayanan et al., 2007; Vosselman and Maas, 2010). A KD-tree is a binary tree data structure in which every node represents a *k-dimensional* point. Every non-leaf node refers to a splitting hyperplane that separates space into two half-spaces. The points in the left *half-space* and in the right *half-space* are represented by the left and right *sub-tree* respectively. In Figure 3.2a, six points in 2D space, i.e. $A(40,45)$, $B(15,70)$, $C(70,20)$, $D(69,50)$, $E(66,85)$, $F(85,90)$, are organized by a KD-tree. In the figure, point A is the starting point and the space is vertically split into two half-spaces. The point whose x coordinate value is bigger than that of point A, is

then assigned to the right sub-tree. As the x value of point B is smaller, it is assigned to the left *sub-tree*. Then, the resulting two half-spaces are horizontally split and subsequently four half-spaces are generated. For example, the right *sub-tree* of point A in Figure 3.2b is split with regard to point C. Points whose y value are smaller than that of point C, i.e. point D, E and F, are assigned to the right *sub-tree*. The same procedure is conducted recursively over x and y until all points are traversed. In 3D, the KD-tree is built by a similar splitting procedure as in 2D, however, the splitting *hyperplane* is conducted recursively on the x, y and z coordinates.

Neighbor searching is one of the oldest problem in computer science (Chen and Lu, 2008). The nearest neighbor (NN) of a query point q is defined by Equation 3.1.

$$NN(q) = \{p | p \in P, \forall o \in P, (o \neq p), |qp| < |qo|\} \tag{3.1}$$

Here, P is a set of n points in a *d-dimensional* space R^d, o, p and q are points in R^d. In geo-referenced point clouds, R^d usually refers to a 3D Euclidean space, and $|qp|$, $|qo|$ are the Euclidean distance between points p and q, and points q and o respectively.

KD-trees have been used for neighborhood querying in huge point cloud data sets because of their efficiency. For a balanced KD-tree, its average query complexity for a set of n points is $O(n\log(n))$ and the worst-case complexity is $O(kn\log(n))$ with k the space dimension (Bentley, 1990; Wald and Havran, 2007; Brown, 2014). A few open libraries that implemented nearest neighbor searching are given in Table 3.1.

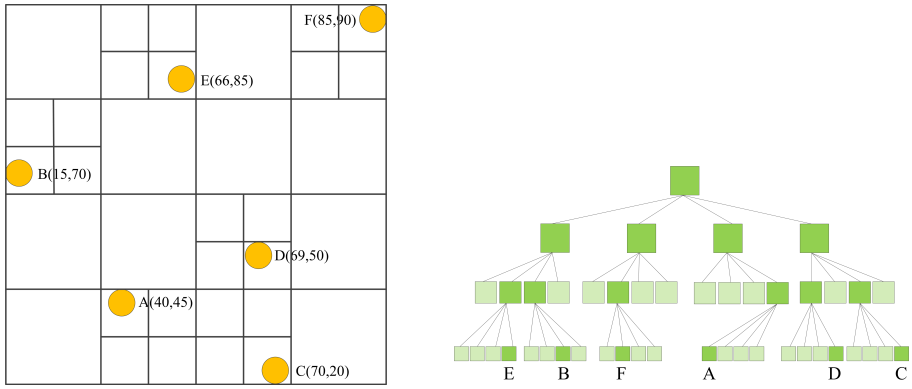
Library	Version	Core structure	kNN search	Radius search
3DTK (Nuechter et al., 2016)	3.0	KD-tree/Octree	✓	×
ANN (Mount and Arya, 2010)	1.1.2	KD-tree	✓	✓
CGAL (Alliez et al., 1997)	4.9	KD-tree	×	✓
FLANN (Muja, 2011)	1.8.0	KD-tree	✓	✓
nanoFLANN Muja and Lowe (2014)	1.2.2	KD-tre	✓	✓

Table 3.1: Open libraries for NNS and their properties.

3.2.3. Quadtree and 2D spatial partitioning

Spatial partitioning is the procedure to divide a space into two or more non-overlapping regions (Samet, 2006). Discrete points obtained by a MLS system are apparently distributed randomly in 3D Euclidean space. Spatial partitioning is usually conducted to efficiently organize such data sets. This section reviews the utility of a quadtree and its advantage in point cloud processing and management.

A quadtree is a tree data structure of which each internal node has exactly four branches. Based on what data the quadtree represents and if the structure of the tree is independent on the sequence in which the data is accessed, quadtrees are categorized as point quadtrees or trie-based quadtrees (Samet, 2006). A trie-based quadtree is able to decompose a space in 2D by dividing a region into four equal sub-quadrants, with each leaf node containing points corresponding to a sub-quadrant. Figure 3.3 is an example of spatial partitioning using a trie-based quadtree and its hierarchical tree data structure representation. Each branch of the quadtree either has four children or is a leaf. A uniform trie-based quadtree of depth n consists of $2^n \times 2^n$ sub-quadrants.



(a) (b)

Figure 3.3: An example of spatial partitioning using a trie-based quadtree and the corresponding tree data structure. (a) Spatial subdivision of a trie-based quadtree. (b) The hierarchical data structure of a quadtree. Internal and leaf nodes are in green and empty nodes are in light green.

Trie-based quadtrees, i.e. region quadtrees, are widely employed to organize point clouds acquired by airborne LiDAR systems (Bi et al., 2014; Zhu and Hyyppa, 2014; Richter et al., 2015; Palha et al., 2017). Notably, as a quadtree is a 2D data structure, the organization of 3D point clouds using quadtrees typically starts by projecting 3D points to a 2D plane. IF the quadtree principle is extended to 3D, we obtains octree. The octree data structure is discussed in detail in Section 3.2.4.

3.2.4. Voxels and Octrees

This section introduces the basic concepts of *voxels* and *octrees* data structures respectively by giving their definitions and explaining how they can be used in point cloud re-sampling and neighborhood searching. Next, a detailed implementation of the octree data structure used in this thesis in the *C++* programming language is discussed. Finally, existing applications of octrees in point cloud organization and processing are presented.

Voxel

As a *pixel* is a 2D representation, a *voxel* is a cube with predefined edge lengths in 3D Euclidean space. Normally, the edge lengths in the three directions are the same. In this thesis, the lengths in the three coordinate directions can be different to make algorithms more flexible. Figure 3.4 is an illustration of a voxel in 3D Euclidean space. Note that the coordinate system used here is right-handed Cartesian.

In this thesis, the data structure of a voxel cell is designed in *C++* as below, i.e. **struct** `VoxelCell`:

Algorithm 3.1: Definition of a VoxelCell data structure.

```
struct VoxelCell
{
```

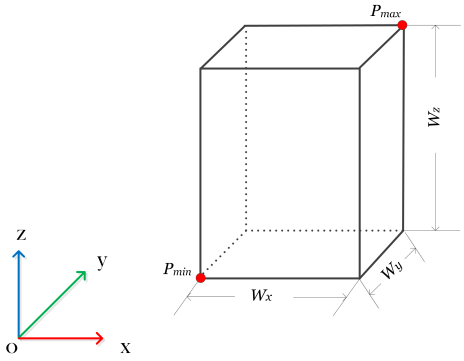


Figure 3.4: Voxel cell in a 3D Euclidean space. P_{max} is the upper right back point and P_{min} is the lower left front point of the voxel cell. W_x, W_y and W_z are the lengths of the edge in the three axis directions respectively.

```
        long x;
        long y;
        long z;
        list<int> PointId;
        Point3D Centroid;
}
```

Here, x, y, and z are the 3D indices of each voxel, and $PointId$ is a container which stores the indices of the points inside each voxel cell. $Centroid$ stores the centroid of all the points inside each voxel cell.

Voxels are extensively used to re-sample point clouds acquired by MLS systems. Figure 3.5 illustrates the re-sampling, i.e. voxelization, of point clouds collected by MLS systems. Firstly the lower left front point and upper right back point are obtained from the axes parallel bounding box of the input point cloud data, i.e. Point P^{min} and Point P^{max} in Figure 3.5. With predefined voxel cell sizes in the three axis directions, the bounding box of the point cloud is divided into cubic cells and the number of bins in the three directions are then calculated respectively. Consecutively the 3D voxel indices of each point are computed by comparing its coordinates with the lower front left point P^{min} in the three directions using Equation 3.2.

$$n_i = \frac{P_i - P_i^{min}}{size_i} \tag{3.2}$$

Here i denotes the x, y, z direction respectively, and n_i is the cell index to which point P_i belongs to. The voxel cells which have no points inside are defined as $Zero$ cells, while the cells which have points are defined as $Positive$ cells. In Figure 3.5, the $Positive$ cells are colored light green and $Zero$ cells are blank. Here, the minimal

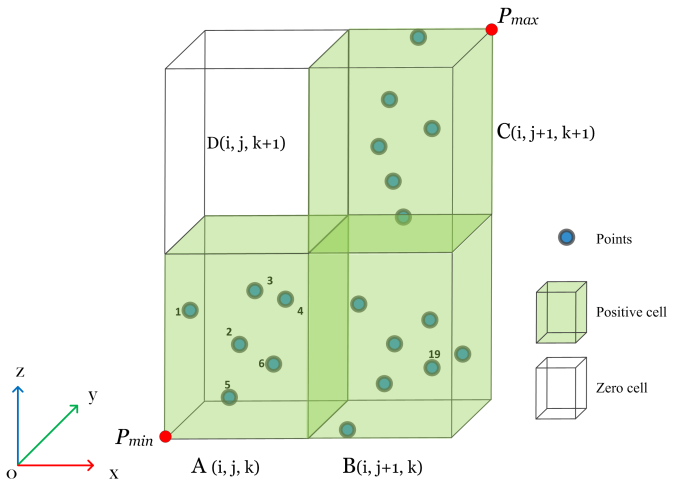


Figure 3.5: Point cloud re-sampling using voxel cells in 3D Euclidean space. Blue dots are points inside each voxel cells and the size of the edges can be different.

voxel size needs to consider the average point density. Normally the minimal voxel size needs to be bigger than the average distance between points. However, the optimal voxel size depends on requirements on the amount of detail and computation time.

Neighbors of a 3D voxel cell are categorized in three classes by considering the type of adjacency to the query cell. Figure 3.6, the 26 neighbor cells in 3D of a voxel cell are categorized as 6 face-adjacent neighbors, 12 edge-adjacent neighbors and 8 vertex-adjacent neighbors. The three categories of neighbors are displayed in green, blue and orange respectively.

After the voxelization of a point cloud using voxels, each voxel cell has an address ID, i.e. x, y and z in Algorithm 3.1. Voxelization using voxels is a procedure of re-sampling a point cloud using 3D voxel cells of uniform dimensions. Thus, the voxel space is interpreted as a 3D array and neighborhood searching in voxel cells is straightforward.

Although neighborhood searching in voxel space is easy and efficient, there exists a lot of memory redundancy when re-sampling using voxels. Thus, a more memory efficient data structure such as *octrees* is preferred when dealing with huge point clouds. The next section will describe the *octree* data structure.

Octree

An octree is a tree data structure of which each node is either a leaf or has exactly eight children (Samet, 2006). It is a 3D extension of the quadtree in 2D (Payeur, 2006). The octree data structure is mostly used in spatial partition of a 3D Euclidean space by recursively subdividing a root node into eight identical octants. In general, building an octree requires $O(n \log n)$ and searching in an octree typically requires $O(\log n)$ running time, here n is the number of points (Samet, 2006).

Figure 3.7 illustrates an octree based spatial partition and its hierarchical tree data

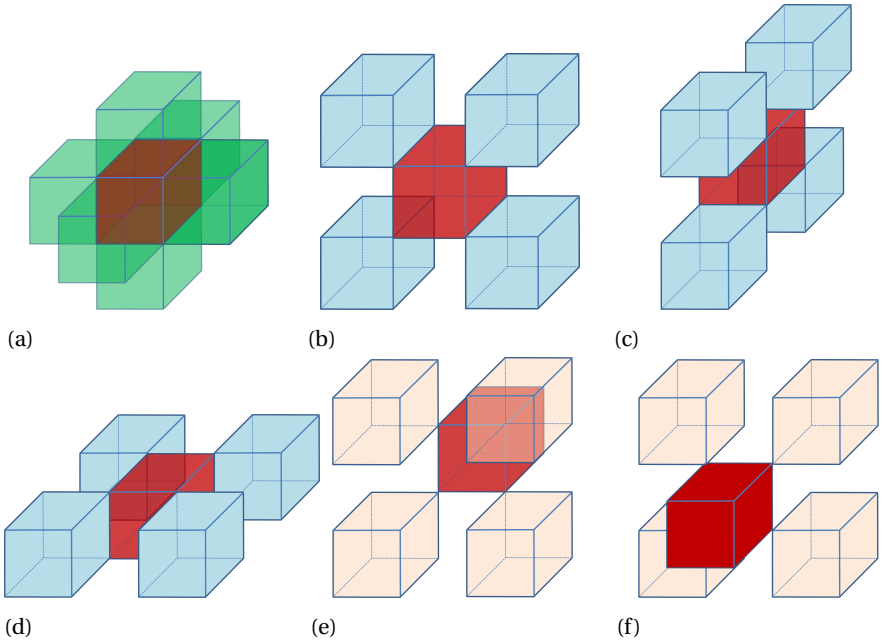


Figure 3.6: Voxel cell and its 26 neighbors in 3D Euclidean space and the red cell is the query cell. (a) Face-adjacent neighbors. (b), (c) and (d) are edge-adjacent neighbors. (e) and (f) Vertex-adjacent neighbors.

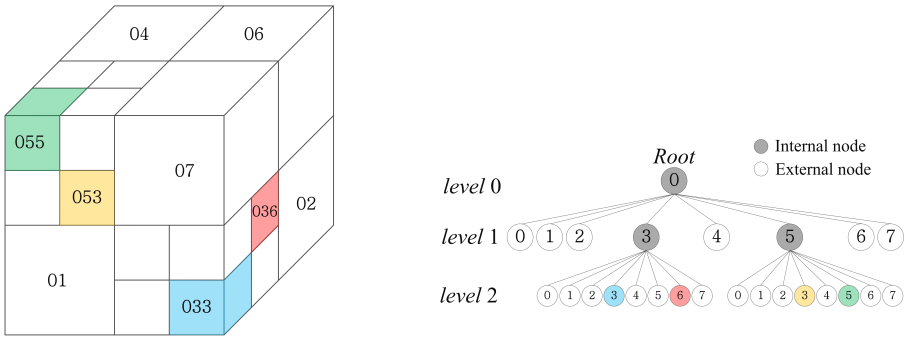
structure. In this thesis, the implemented addressing scheme for each octant of an octree is employed from (Major et al., 1989). As indicated in Figure 3.7a, the partition starts with a root node, which is labeled by digit 0. The procedure goes on by subdividing the root node into eight identical octants. If the octant is not empty, the subdivision continues until the pre-defined stop criteria are reached. Octants of the same subdivision level have the same number of digits. For example, the octants in light green and in light orange, which are labeled by 055 and 053, belong to level 3 and thus the number of digits is 3. Figure 3.7b illustrates the hierarchical tree data structure of an octree, in which the internal nodes are in gray. Each node corresponds to an octant in the subdivision in Figure 3.7a as their addressing digits indicate.

The data structure of an octree node in this thesis is defined as *OctreeNode* below:

Algorithm 3.2: Definition of octree node data structure.

```
struct OctreeNode
{
        bool Leaf;
        long PointNumber;
        OctreeBound BoundingBox;
        Point3D* Points;
        OctreeNode* Child[8];
        vector<int> Path;
}
```

Here, *Leaf* indicates whether the current octant is a leaf node, which has no sub-branches.



(a) (b)

Figure 3.7: Octree based space partition and its hierarchical data structure. (a) Octree based Cartesian space subdivision and voxel indexing. (b) Octree hierarchical data structure. The octant numbered 055 in the top left of (a) indicates where in the hierarchy in (b) this voxel can be found.

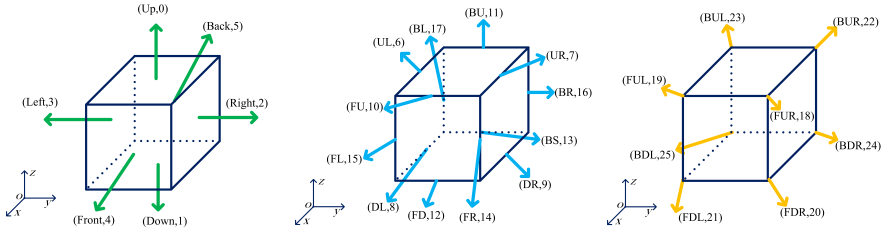
BoundingBox defines the 3D bounding box of each octant and *Points* stores the points inside of the octant. *Child[8]* is an array of 8 same *OctreeNode*, which are the eight branches of an internal node. *Child[8]* is an array of pointers to the 8 children *OctreeNode* which are either leaves or octree node of one level deeper. Unlike in voxel space, octrees organize the octants in a hierarchical tree structure and thus neighborhood searching using octrees is not straightforward. However, as in voxel space, the number of possible adjacency directions in 3D is 26 for a uniform resolution octree. As shown in Figure 3.8, corresponding to neighbors of the three categories given in Figure 3.6, the possible directions are given and labeled by an index number as indicated. There are a few octree implementations as illustrated in Table 3.2. However, there is no kNN searching, neigher in octant space nor in point space, implemented in those open source libraries. Thus, neighborhood searching in octant space based on the octree data structure is implemented from scratch in this thesis.

Name	version	Open source	kNN search
PCL (Nuechter et al., 2013)	1.7.1	✓	×
PoTree (Sch"utz, 2016)	1.3	✓	×
CloudCompare (Girardeau-Montaut, 2003)	2.9	✓	×
OctoMap (Hornung et al., 2013)	1.8.1	✓	×

Table 3.2: Open source libraries of octree implementation.

To efficiently search neighbor octants in an octree hierarchical structure, lookup tables for the possible directions are given in Table 3.3, Table 3.4 and Table 3.5 for face-adjacent, edge-adjacent and vertex-adjacent neighbor cells respectively.

For example, the octree cell labeled 033 in Figure 3.7a is a Front-Down (FD) edge-adjacent neighbor octant of the cell labeled 036. Assuming the query octree cell is 036, its edge neighbor octants are searched. Firstly, the right-most digit of its *Path* is 6 and the *FD* direction is encoded as direction 12. Next, in Table 3.4 the digit in column



(a) (b) (c)

Figure 3.8: Neighbors of an octree octant and the possible adjacency directions. (a) Possible directions for face-adjacent neighbors. (b) Possible directions for edge-adjacent neighbors. FU indicates Front-Up direction. (c) Possible directions for vertex-adjacent directions. FUL indicates Front-Up-Left direction.

Query direction		Up 0	Down 1	Right 2	Left 3	Front 4	Back 5
Orignal Path Digit	0	4	4+Down	2	2+Left	1	1+Back
	1	5	5+Down	3	3+Left	0+Front	0
	2	6	6+Down	0+Right	0	3	3+Back
	3	7	7+Down	1+Right	1	2+Front	2
	4	0 + Up	0	6	6+Left	5	5+Back
	5	1 + Up	1	7	7+Left	4+Front	4
	6	2 + Up	2	4+Right	4	7	7+Back
	7	3 + Up	3	5+Right	5	6+Front	6

Table 3.3: Lookup table for face-adjacent neighbor octant searching in an octree structure.

Query Direction		UL 6	UR 7	DL 8	DR 9	FU 10	BU 11	FD 12	BS 13	FR 14	FL 15	BR 16	BL 17
Original Path Digit	0	6+L	6	6+DL	6+D	5	5+B	5+D	5+BD	3	3+L	3+B	3+BL
	1	7+L	7	7+DL	7+D	4+F	4	4+FD	4+D	2+F	2+FL	2	2+L
	2	4	4+R	4+D	4+DR	7	7+B	7+D	7+BD	1+R	1	1+BR	1+B
	3	5	5+R	5+D	5+DR	6+F	6	6+FD	6+D	0+FR	0+F	0+R	0
	4	2+UL	2+U	2+R	2	1+U	1+BU	1	1+B	7	7+L	7+B	7+BL
	5	3+UL	3+U	3+R	3	0+FU	0+U	0+F	0	6+F	6+FL	6	6+L
	6	0+U	0+UR	0	0+R	3+U	3+BU	3	3+B	5+R	5	5+BR	5+B
	7	1+U	1+UR	1	1+R	2+FU	2+U	2+F	2	4+FR	4+F	4+R	4

Table 3.4: Lookup table for edge-adjacent neighbor octant searching in an octree structure.

Query direction		FUR 18	FUL 19	FDR 20	FDL 21	BUR 22	BUL 23	BDR 24	BDL 25
Original Path Digit	0	7	7+L	7+D	7+DL	7+B	7+BL	7+BD	7+BDL
	1	6+F	6+FL	6+FD	6+FDL	6	6+L	6+D	6+DL
	2	5+R	5	5+DR	5+D	5+BR	5+B	5+BDL	5+BD
	3	4+FR	4+F	4+FDR	4+FD	4+R	4	4+DL	4+D
	4	3+U	3+UL	3	3+L	3+BU	3+BUL	3+B	3+BL
	5	2+FU	2+FUL	2+F	2+FL	2+U	2+UL	2	2+L
	6	1+UR	1+U	1+R	1	1+BUR	1+BU	1+BL	1+B
	7	0+FUR	0+FU	0+FR	0+F	0+UR	0+U	0+L	0

Table 3.5: Lookup table for vertex-adjacent neighbor octant searching in an octree structure.

(6,FD), i.e. 3, is found. Then, the right-most digit 6 is replaced by 3. Thus, the Front-Down edge-adjacent neighbor octant of 036 is the octant labeled by 033. Similarly, neighbors of all the 26 directions in 3D can be queried using the three lookup tables.

The octree data structure has been widely used in computer science. Octrees are used to generate models of different levels of details Luebke et al. (2003); Döllner and Buchholz (2005), ray tracing (Meagher, 1982; Lastra and Revelles, 2000; Barboza and Clua, 2011; Mcguire and Mara, 2014), geometric modeling (Losasso et al., 2004; Lee H., 2009; Zia et al., 2013). In robotics and game design, octrees are used for collision detection (Jung and Gupta, 1997; Chengming and Zhiyong, 2009; Tang et al., 2010; Hornung et al., 2013; Xu and Barbič, 2014), mesh generation (Sampath and Biros, 2010), and real-time mapping (Thrun et al., 2000).

In recent years, octrees are used in point cloud processing, such as point cloud compression (Schnabel and Klein, 2006), out-of-core massive simplification (Lindstrom, 2000; Scheiblauer and Wimmer, 2011; Elseberg et al., 2013), skeletonization (Bucksch and Lindenbergh, 2008), point cloud segmentation (Woo et al., 2002; Vo et al., 2015; Su et al., 2016).

3.3. Conclusions

In this chapter, the following research question was investigated and answered:

How to efficiently organize huge MLS point clouds?

For manipulation, processing and storage of the huge point clouds acquired by MLS systems, it is fundamental to have proper organization for efficient point access, querying and storage. Vector data structures, i.e. TINs and R-Trees, are do not combine efficient space organization with querying. KD-trees and quadtrees are the two data structures that are efficient in querying and spatial partitioning in 2D. Next, voxels and octrees are spatial data structures that generalized and efficiently perform the above mentioned operations in 3D. Their capability of organizing huge point clouds are emphasized by analyzing the procedure of voxelization and neighborhood searching strategies.

Throughout this thesis voxels and octrees will be used to organize point clouds. Moreover, their data structures will be exploited in the design of different algorithms. In Chapter 4, voxels are used to generate 2D grids for estimation of water-flow direction in mountain road environment. Chapter 5 uses voxels to re-sample tree points, then a voxel adjacency based algorithm for tree individualization is developed. In Chapter 6, MLS scanned point clouds are organized by octrees and an algorithm for object recognition is presented.

4

Estimation of Excavation Volume and Water Flow Around A Mountain Road

Mountain roads are prone to natural hazards, such as land slides and rain water draining. When mountain roads are widened, the engineering work is expensive and needs to be well planned. This chapter concentrates on the application of MLS point cloud data in a mountain road environment. First, an algorithm for automatic estimation of excavation volume of mountain road from MLS data is presented. Consecutively, an algorithm for water flow direction forcasting is proposed.

4.1. Introduction

For mountainous rural areas, roads are lifelines to the people and the safety of the road and its environment is an important concern. The safety and condition of the roads need regular inspection and monitoring for security reasons. Traditionally, the road and roadside geometry are measured by surveyors on a point-by-point basis using Total Station or GNSS. Unlike these methods, MLS does not measure the 3D positions of well-defined points in the terrain. Rather, it acquires many random 3D points are acquired by sampling the whole road surface and the surface of the roadside, visible by the MLS system. The quality of the individual 3D MLS points is worse than the quality provided by Total Station or GNSS, but an MLS is able to provide full coverage instead of acquiring only single points. In addition, the acquisition is fast and automatic, and there is no need for surveyors to leave the car. These latter advantages also distinguish MLS from static Terrestrial Laser Scanning, in which a point cloud is obtained from a panoramic scanner mounted on a tripod (Vosselman and Maas, 2010; Gikas, 2012). Such panoramic scans need to be combined with GNSS positioning data in post-processing to acquire a geo-referenced point cloud. To conclude, the MLS is currently the fastest ground based method for acquiring 3D surface information in large areas and to get a large point cloud of a long object like a road. It has already been applied for high way surveying (Bitenc et al., 2011; Zhou and Vosselman, 2012), sandy coast morphology and riverine erosion measurements (Kukko et al., 2009; Vaaja et al., 2011), railway monitoring and rail geometry extraction (Kukko et al., 2009; Gikas and Stratakos, 2012), road environment management (Vaaja et al., 2011; Tao, 2000; Pu et al., 2011; Jeong et al., 2007), and highway documentation (Foy et al., 2007; Mancini et al., 2012; Sérgio et al., 2005; Nuechter et al., 2013; Jaakkola et al., 2008). Road management starts with the planning phase and ends with the rehabilitation or maintenance phase (Kukko et al., 2009). When the road has been constructed, road information is needed for an increasing number of other applications, such as noise modeling, road safety, road maintenance, location-based services and navigation (Foy et al., 2007). Road documentation and management mainly consist of recording the road geometry and monitoring the road environment (Mancini et al., 2012; Gikas and Daskalakis, 2008). Road geometry refers to parameters used for the design of a road, such as design speed, stop sign distance, line of sight, number of lines, line width, longitudinal and transverse slope, road pavement materials and so on. Road environment refers to the surroundings of the road on both sides, including buildings, trees, vegetation, traffic signs, traffic light poles and other objects.

The information discussed above plays an important role in the ongoing maintenance of the road, especially for mountain roads where there is a risk for rock and stone fall. Additionally, the geometric features observed on a mountainous road could be

helpful for monitoring the flow of rainwater in the case of heavy rain and may be used to assist natural disaster prevention (Tarolli et al., 2013). Moreover, steep and unstable road sides may cause landslides, resulting in further road damage (Razak et al., 2011).

The MLS point cloud data contains information that can serve as input for flood hazard and landslide prediction. In Poppenga et al. (2010), a Digital Elevation Model (DEM) is constructed from point cloud data to model surface flow, and was applied to flood inundation and erosion estimation. Also in Kazuhiro et al. (2005); White et al. (2010); Ziegler and Giambelluca (1997) high-resolution DEM data is generated to predict surface erosion and to estimate the amount of sediment drained by streams. Especially for mountainous roads, rocks on roadside hills could fall down and cause risks. Also, water flow may cause erosion at the side of the road, eventually resulting in road damage. Moreover, steep and unstable roadsides may cause landslides resulting in further road damage.

This chapter first presents an approach for automatically estimating the roadside material volume to be excavated for road widening. Firstly, MLS point cloud data in a mountainous area are down sampled and the outliers and noisy points are removed. Based on the data, normal vectors, and 2D slopes are estimated at every point. Then, an automatic iterative floating window approach, taking advantage of point height, normal vector and slope, is used to filter and segment the road points. After that, a local neighborhood feature is defined based on the vectors between a query point and its neighboring points to obtain the outline and skeleton of the road. These steps finally allow us to compute the volume of the roadside material that would have to be moved to widen the road, in our study, by 4 m. In addition, an analysis of the quality of the results is presented notably by comparing results from different data sets both sampling the region of interest. Finally, conclusions and future work are discussed.

For water flow direction forecast, this chapter computes the roadside environment catchments and estimates where and how water would flow over the surface if it rains. To some extent, stone fall is expected to follow the water flow direction as well. After that, the D8 algorithm is used to estimate the water flow direction on the road surface and roadside terrain. Based on these directions, the road environment is divided in runoff sections.

4.2. Methodology

This section firstly explains the procedure for estimating the amount of roadside material that has to be removed for road widening. Secondly, the methodology for estimation of runoff from MLS point cloud data is presented.

1. Methodology for excavation volume estimation, as shown in Figure 4.1.
    (a) Point cloud data pre-processing; the original point cloud data have a very high point density and need to be down-sampled before processing. Additionally, outliers are removed, and a Digital Surface Model (DSM) is generated;
    (b) Surface normal estimation;
    (c) Slope and aspect estimation;

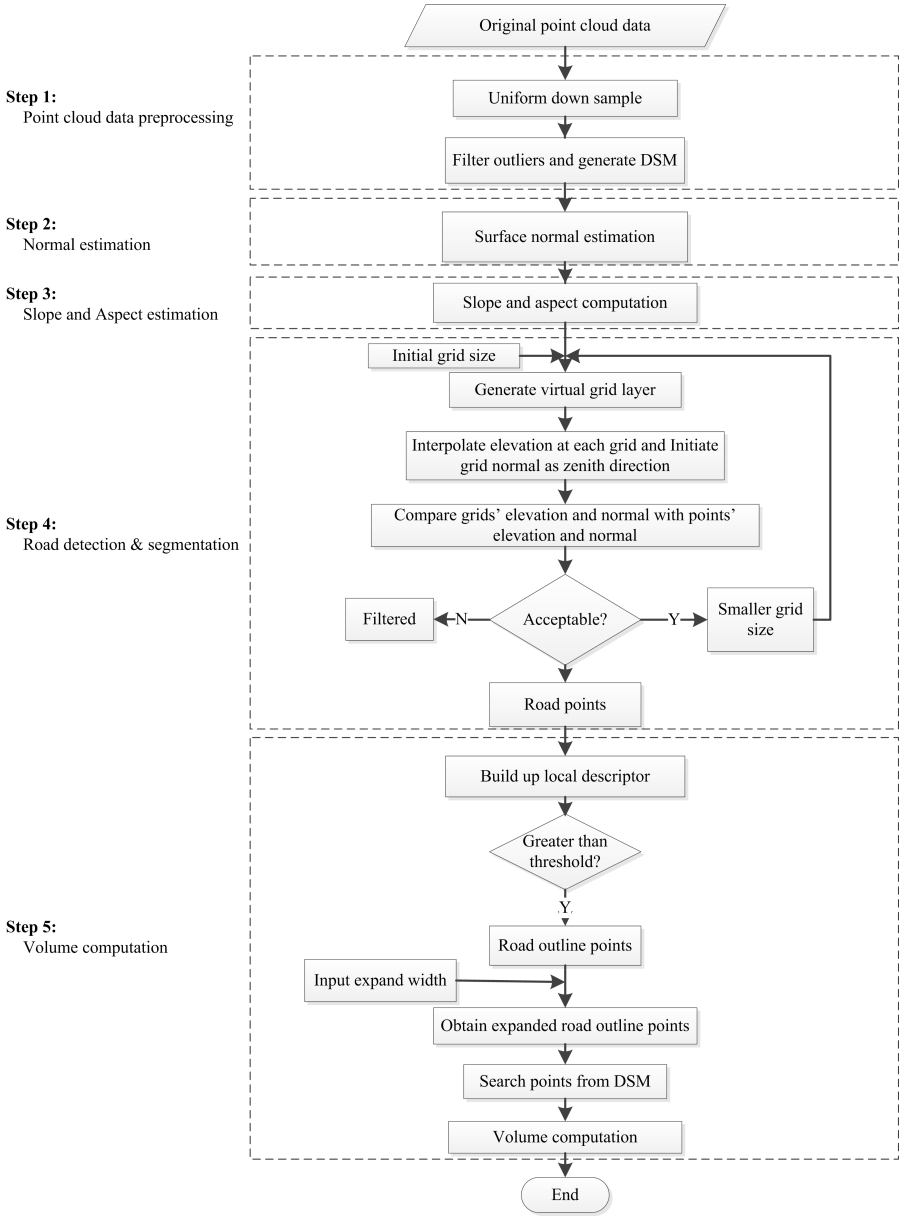


Figure 4.1: Overall methodology of mountain road excavation volume estimation.

(d) Road detection and segmentation; taking the normal and slopes as estimated in steps (b) and (c) as input, an automatic iterative filtering approach is used to segment the road points from the down-sampled point cloud

data;

(e) Volume computation; the material volume that needs to be moved to widen the road is computed based on step (d).

2. Methodology for estimating water runoff, as shown in Figure 4.2

(a) Point cloud pre-processing. The original point cloud data have very high density and is down-sampled before processing. Outliers are also removed as well;

(b) Local surface normal and 2D slope estimation;

(c) Road delineation, which allows to decompose the point cloud into road and roadside points;

(d) Estimation of flow direction (in case of rain).

4.2.1. Pre-processing

Raw point cloud data have a very high point density, so for quick and efficient processing, down-sampling is a necessity. The approach followed here is to represent the point cloud data by voxels (Slattery et al., 2012; Gorte and Pfeifer, 2004; de la Puente et al., 2008). In this chapter, a uniform-sized voxel filtering approach was used, as implemented in the Point Cloud Library (PCL) (Gorte and Pfeifer, 2004). The main concept of the approach is to create a 3D voxel using a given voxel size. All points in a voxel are represented by their centroid (Foy et al., 2007).

The outliers of the down-sampled point cloud data have to be removed before estimating geometric features from the point cloud data. In this procedure, the query point neighborhood concept is introduced, as shown in Figure 4.3. Here, P represents the set of points within radius R_{query} of query point p_{query}, so P is defined in Equation 4.1.

$$P = \{p_i \,|\, \|p_i - p_{query}\| \leq R_{query}\} \tag{4.1}$$

Here, p_i represents the individual points from the point cloud.

Outliers are the measurements located at edges or discontinuous boundaries where there should be no points (Petitjean, 2002). There is abundant literature on the approaches used to remove outliers (May et al., 2008; Rusu et al., 2008; Rusu, 2010). The approach used in this chapter is based on a statistical analysis of each point's neighborhood (Li et al., 2010a; Castillo, 2013). For each point $p_{query} \in P$, the mean distance $\overline{d}$ to its closest k neighbors is calculated. After that, for each point in the point cloud, the mean distance and standard deviation of the distances to their k nearest neighbors are determined. The main objective is to retain only those points whose mean distance to the closest neighbors is similar to the mean distance. As this is a measure of the underlying point cloud density surrounding a point, the criterion for keeping a point is simply formulated as Equation 4.2.

$$P^* = \left\{p_q \in P | (\mu_k - \alpha\sigma_k \leq \overline{d} \leq (\mu_k + \alpha\sigma_k)\right\} \tag{4.2}$$

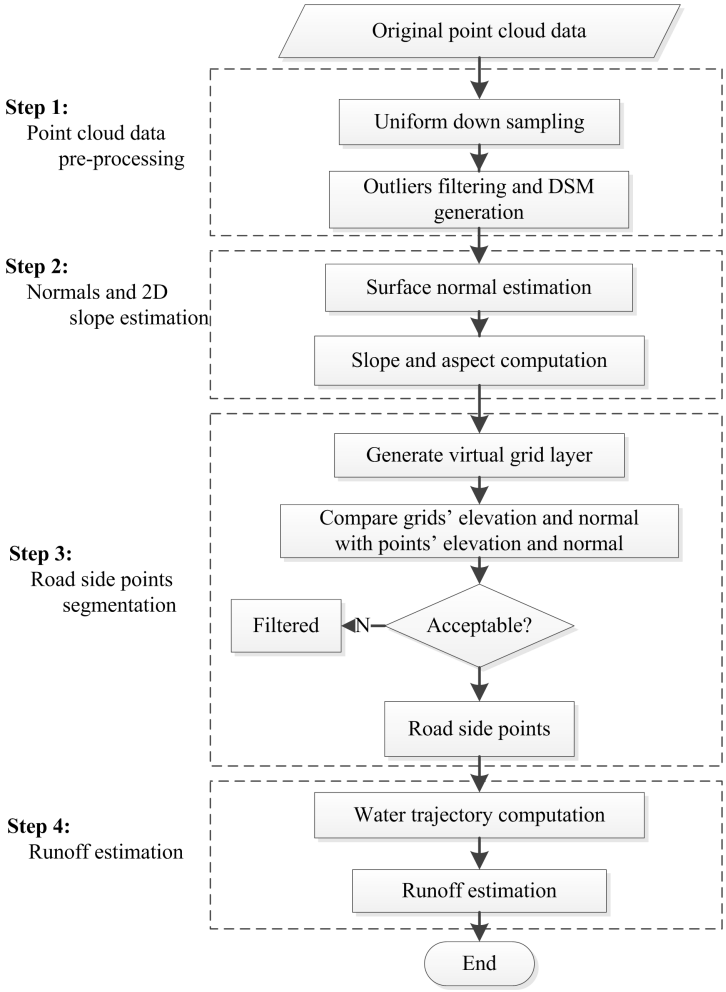


Figure 4.2: Method for estimation of flow direction (in case of rain) from MLS point cloud data

Here, α is a desired density restrictiveness factor, while μ_k and σ_k are the mean and standard deviation of the distance from a query point to its neighbors respectively, and P^* is the set of remaining points.

4.2.2. Local surface normal estimation

The surface normal at a discrete point is a vector perpendicular to the tangential plane of the local surface at that point. Various methods exist to estimate a normal at a certain point in 3D point cloud data (Castillo, 2013; Dey et al., 2005; Thürmer and Wüthrich, 1997; Klasing et al., 2009; Puente et al., 2013a). The simplest is based on 3D plane fitting. With this method, the problem of determining the normal of a point becomes

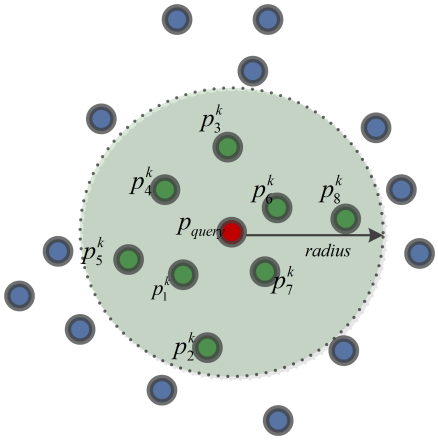


Figure 4.3: Neighborhood of a query point within a certain radius.

a least-square 3D plane estimation problem for a suitable spatial neighborhood. Figure 4.4 depicts the concept of normal estimation in discrete 3D point cloud data.

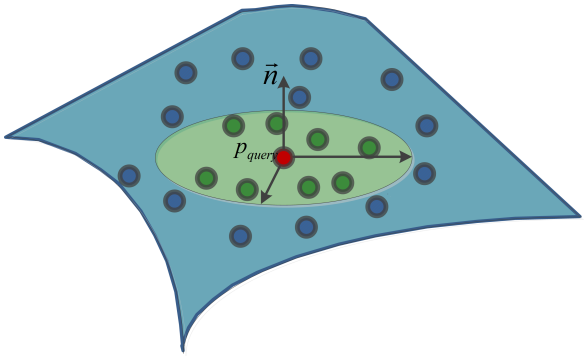


Figure 4.4: Normal estimation at a query point using a local plane fitting approach.

When estimating a local normal using least-squares, the goal is to approximate the tangent plane at a point of interest and take that plane's normal. The best fitting plane is obtained by determining those planar parameters that minimize the squared distances between suitable neighboring points and the plane to be estimated. Suppose we have a point of interest with Euclidean coordinates $(x, y, z)^T$ and a set of k neighboring points. The least-squares method yields a normal vector $\vec{n} = (n_x, n_y, n_z)^T$, for which the error defined in Equation 4.3 is minimized.

$$error = \sum_{i=1}^{k} (p_i^T \vec{n} - d)^2 \tag{4.3}$$

Additionally, $|\vec{n}| = 1$where $p_i = [x_i, y_i, z_i]^T$ is a neighborhood point and k is the predefined number of considered neighborhood points.

The solution of Equation 4.3 for $\vec{n}$ is in general given by the smallest eigenvector of the matrix M in Equation 4.4.

$$error = \frac{1}{k}\sum_{i=1}^{k}(p_i - \overline{p})(p_i - \overline{p})^T \tag{4.4}$$

Here, $\overline{p} = \frac{1}{k}\sum_{i=1}^{k} p_i$.

4.2.3. Local slope computation

The 2D slope, also known as the 2D gradient, is the vector field of a surface. The vector direction points in the direction of the greatest change in height, and the vector's magnitude is equal to the rate of change. Based on the previously generated gridded DSM, the height at every grid point is interpolated from neighboring points by inverse distance interpolation with power 2. If the grid size is d_{grid}, then the slope S_i in the direction of each of the eight neighboring grid cells is given by Equation 4.5.

$$S_i = \frac{H_i - h_{query}}{d_i} \tag{4.5}$$

Here, H_i is the elevation of the *i-th* neighbor of the query point, and h_{query} is the elevation of the query point itself. The variable d_i is the distance between the *i-th* neighbor and the query point. Note that d_i is the square root of the grid size in diagonal direction.

4.2.4. Road detection

Based on the normal vectors and slopes that were computed at each point in the previous steps, an automatic iterative point cloud data filtering approach is used to detect road surface points from the point cloud data. The main steps are:

1. Input an initial grid and window size;
2. Generate a virtual reference 3D gridded layer. The elevation of each grid point is interpolated from its neighboring points, and also the grid point's normal as unit vector and its direction to zenith. Based on this layer, a window of predefined size is created to move over the grid and point cloud;
3. In the current moving window, compare for each grid point the 3D layer grid elevation and the normal vector direction with that of the point cloud; Calculate the height and angle differences between the 3D layer and the point cloud to verify if the differences are beyond the threshold;
4. If the difference is less than the distance and direction threshold, then the point is accepted as a road point, else the point is regarded as off-road point;
5. Go to step 2 and generate a new virtual layer using a smaller grid size, then iteratively process the point cloud again;
6. The loop ends when the grid size reaches a pre-defined smallest size.

4.2.5. Calculation of excavation volume

In this step, based on the obtained road points from step 4, the outlines of road are determined. First, a local feature descriptor is defined based on the neighborhood concept.

As shown in Figure 4.5, if $P = p_1, p_2, ..., p_k$ is the set of neighbors of P_{query} within radius R_{query}. The local vector $\vec{r}_i$ at query point P_{query} is defined as $\vec{r}_i = P_i^k - P_{query}$. The edge descriptor is defined as the sum of these local vectors in Equation 4.6.

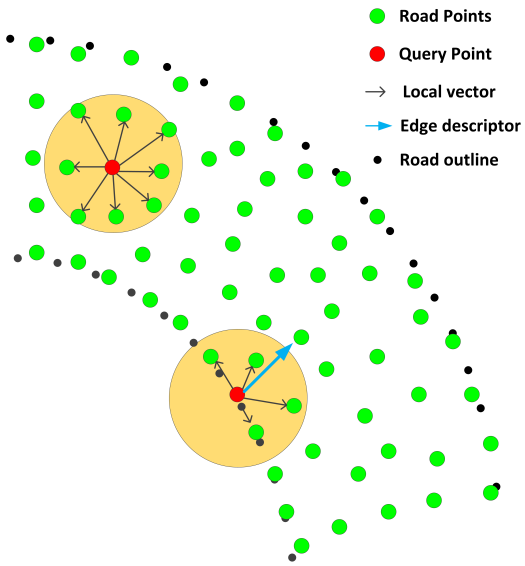


Figure 4.5: Illustration of local neighbourhood feature descriptor.

$$E_{des} = \sum_{i=1}^{k} \vec{r}_i \tag{4.6}$$

In case the query point is indeed an edge point, the value E_{des} is greater than that of a non-edge point in the road point cloud and the direction of the descriptor toward the inner body of the road.

After the road outline is determined, the road central line is estimated based on the location of the road edges. Now suppose the road needs to be widened to four lanes. As a consequence, a certain volume of the road has to be removed or added to extend the flat road surface.

As shown in Figure 4.6, roadsides were divided into slices. For each slice, the volume is computed. Summing all of the sliced volumes together gives the total volume that needs to be excavated or filled. Note that the sign of a sliced volume indicates whether material needs to be removed (positive sign) or added (negative sign). Based on the road central line and edge lines, as well as the expanded width of the road, the expanded edge points are found.

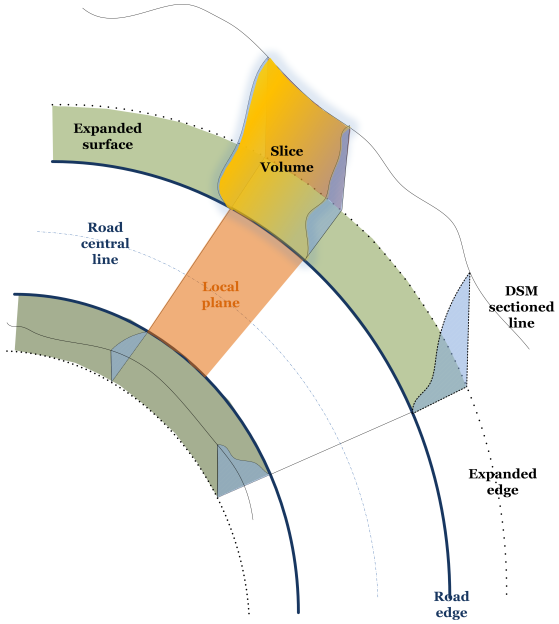


Figure 4.6: Computation of excavation volume.

To minimize the interpolation error, the resolution for both the road parallel and the road perpendicular direction is defined based on the point cloud data resolution, as shown in Figure 4.7.

Suppose V_T^L and V_T^L denote the total volume on the southern and northern side respectively, such that Δv_i^L and Δv_i^R represent the volume of the *i-th* slice on southern and northern side. N_L and N_R denote the number of slices. Thus, the total volume to be moved on each side of the road is determined by Equation 4.7.

$$\begin{aligned} V_T^L &= \sum_{i=1}^{N_L} \Delta v_i^L, \\ V_T^R &= \sum_{i=1}^{N_R} \Delta v_i^R \end{aligned} \tag{4.7}$$

If for example $V_T^L > V_T^R$, we could choose to extend the road on the right side to save time and expenses.

4.2.6. D8 algorithm

The D8 algorithm introduced by O'Callaghan and Mark (1984), is a grid based algorithm and is widely used due to its simplicity. For a given query grid point, the D8 algorithm approximates the primary flow direction by choosing the direction to the neighbor with maximal 2D gradient, as illustrated in Figure 4.8a.

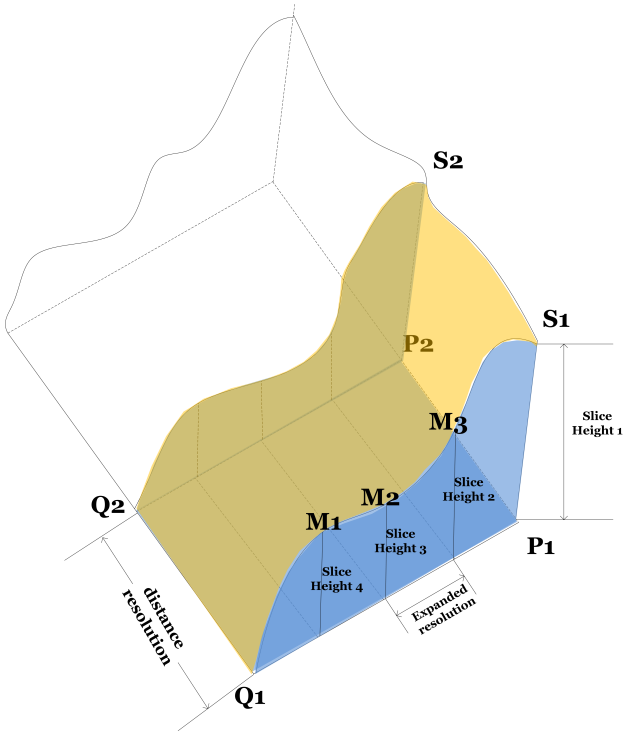
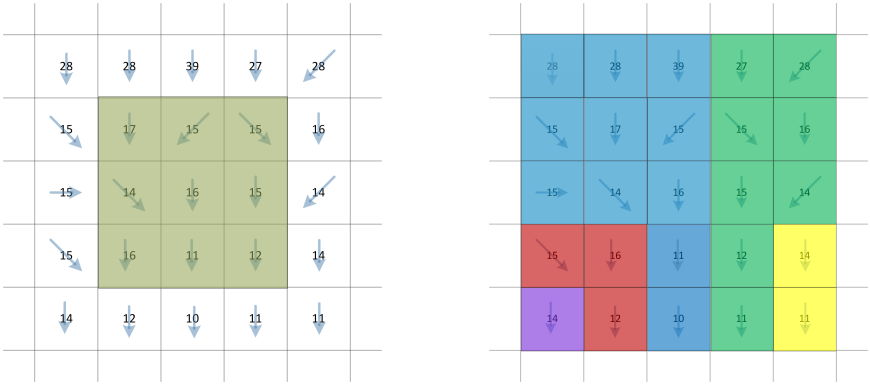


Figure 4.7: Slice volume computational geometry.



(a) (b)

Figure 4.8: Concepts of D8 algorithm. (a) D8 algorithm flow directions.
(b) Upstream catchment area of grid cells.

For example, the flow direction from the central pixel, with value 16, is downward, because the gradient towards the pixel directly below, with value 11, is maximal among

the eight neighbors of the central pixel. In the next step of the D8 algorithm, the flow is followed. In Figure 4.8a, all flow eventually terminates at the pixels in the bottom row.

Applying this method on all the roadside pixels results in a decomposition of the sampled roadside into different catchments. Large catchments correspond to a large local water inflow, as shown in Figure 4.8b, and the area is defined as upstream catchment area. The flow direction is determined for each pixel and pixels that are flowing towards the same bottom pixel, the sink, are assigned in the same randomly allocated color.

In this work, the original down-sampled point cloud data is organized in a uniform grid, and the height assigned to a grid cell is the mean height of all points belonging to the grid cell. Each grid cell is potentially surrounded by eight neighboring grid cells. The gradient for each of these eight directions is obtained using Equation 4.8. Then the D8 algorithm is applied to the gridded point cloud to compute the local flow directions and, by accumulating flow to consecutively compute catchments and sinks.

$$S_i = \frac{H_i - h_q}{d_i} \tag{4.8}$$

where H_i is the elevation of the $i-th$ neighbor of the query point, and h_q is the elevation of the query point itself, while d_i is the horizontal distance from the query point to the $i-th$ neighbor. Note that d_i is $\sqrt{2}w$ in diagonal direction.

4.3. Implementation and testing of the method

4.3.1. Software implementation

The methodology described above is implemented on an ordinary Dell desktop computer, which has an Intel Xeon 3.6 GHz CPU on board and 16 GB random memory. The implementation of the software is in the C++ language. Also, the Point Cloud Library (PCL) statistical outliers filtering tool is used in the processing. The whole processing took 23.184 seconds for the tested data set containing 13,169,989 points.

4.3.2. Data description

The point cloud data studied in this chapter was acquired by the University of Vigo, Spain. The approximate location of the studied road is shown in Figure 4.9.

The entire study area is shown in Figure 4.10. The study area contains a road in a mountainous region. A top view of the data set is shown in Figure 4.10a. The mobile LiDAR system selected for this work was the Lynx Mobile Mapper from OPTECH. The Lynx uses two LiDAR sensors to collect survey-grade LiDAR data at 500,000 measurements per second with a 360° FOV (per scanner) (Puente et al., 2013a,b).

The system incorporates the POS LV 520 unit produced by Applanix, which integrates an Inertial Navigation System with two GNSS antennas, providing an accuracy of 0.015° in heading, 0.005° in roll and pitch, 0.02 m in the X, Y dimension and 0.05 m in the Z axis. All those data are determined by differential GPS post-processing after data collection using GPS base station data. The coordinate system used for this work is UTM-WGS84. The original point cloud data set contains 5,838,794 points and has an average point density of 2084 points per square meter. It covers a 132-m stretch of

Figure 4.9: Approximate location of the studied road.

road. In Figure 4.10c, we can see that the road was constructed in a mountainous area and has steep embankments on southern and northern side as shown in Figure 4.10a.

4.3.3. Geometric computations

Slope computation

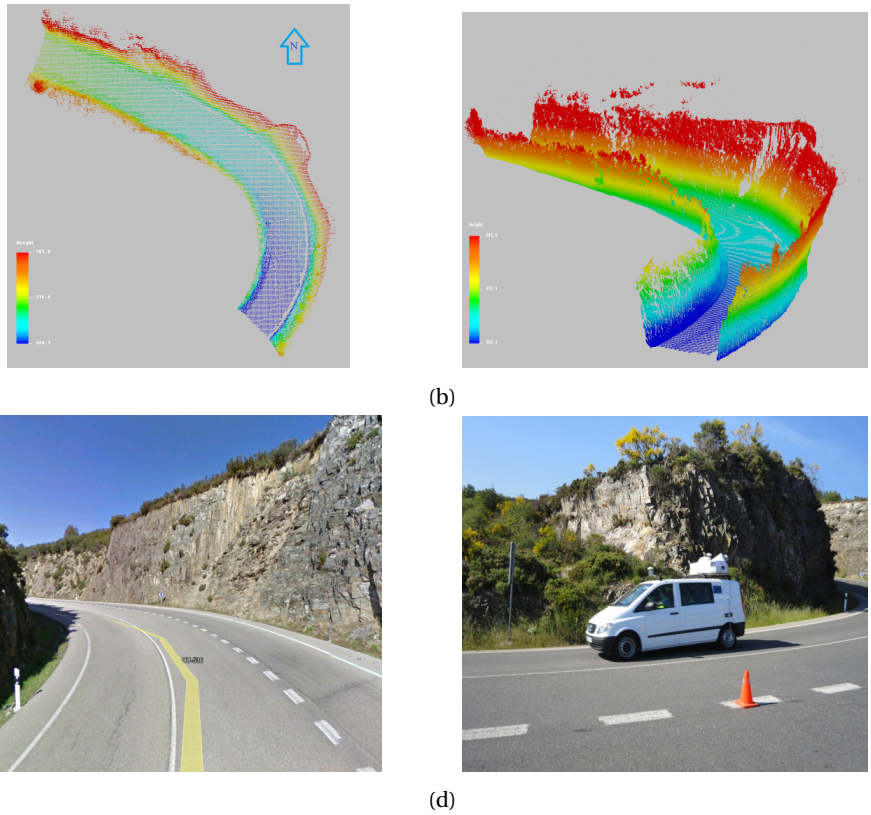
As depicted in Figure 4.11, the slope in this area varies from 0 to 88.1 degrees. The figure is color coded with red indicating a large slope and blue indicating a small slope. Points with small slopes are mainly road points. The points of the road sides have larger slope values corresponding to the steepness of the road side terrain. The dots on the road encircled in red have larger slope value than other road points; these are in fact traffic cones that can also be seen in Figure 4.10d. There is a known landslide site located within the light green circle, which has smaller slope values that stands in contrast to the steep roadside of its neighboring points.

Road detection and segmentation

Road points were identified according to the method outlined in Section 4.2. The minimum virtual grid size was set to 0.1 m, because there is a very high point density in the original point cloud dataset. Additionally, the height threshold was set to 0.3 m and the angle threshold was set to 15 degrees. In this processing, a total of 42,717 road points was abstracted and segmented, as shown in Figure 4.12.

Volume computation

First, based on the segmented road points, the road outline was determined following the method of Section 4.2.5. In this chapter, the local descriptor threshold was set at 1.5, which means that all points with an edge descriptor value greater than 1.5 were regarded as road outline points.



(a) (b)

(c) (d)

Figure 4.10: Original point cloud data set of the study area. (a) Original MLS point cloud data in 3D view. (b) Original MLS point cloud data in 3D side view. (c) Study area in Google Street View. (d) Data collection with MLS.

The abstraction results are shown in Figure 4.13. In total, 429 outline points were identified. Figure 4.13 (a) depicts the abstracted road central line, and Figure 4.13 (b) illustrates in addition the road lines of a projected road expansion by 4 m on each side.

After the extraction of the points above the possible location of the widened road (i.e., 4 m southern and northern of the current road), the vertical distances between points representing the current surface and the expanded road planes were determined, and the excavation volume was estimated. Figure 4.14 shows two profiles of the current surface height at a distance of 4 m southern and northern from the current roadsides. The horizontal axis follows the road starting from its lowest point. In this chapter, the resolution in the road parallel direction was set to 1 m and in the road perpendicular direction to 0.5 m. Figure 4.14 shows the overall height increase of the surface profiles in the road parallel direction.

Figure 4.15 shows the volumes of the slices which were computed as described in Figure 4.7. Because of some low water drain elements on the road side, there are values that are below 0, which means that if the road is widened by 4 m, some of the volume

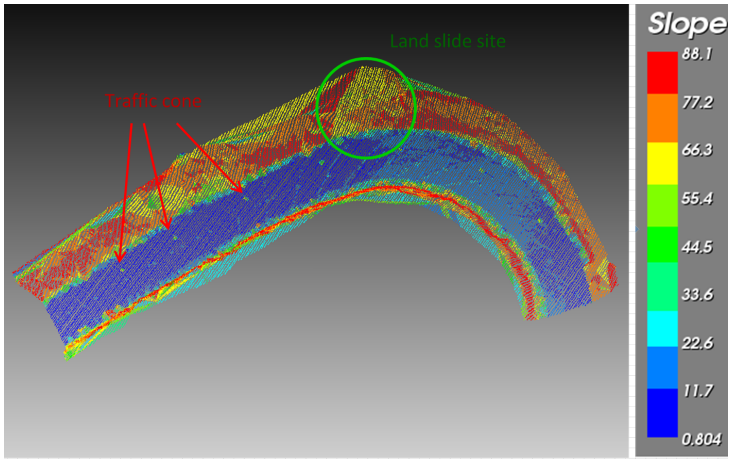


Figure 4.11: 2D slope at each point of the studied road.

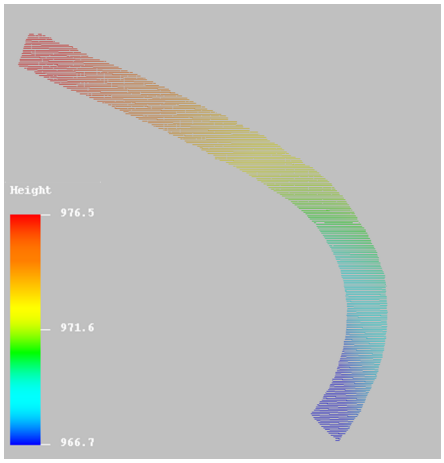


Figure 4.12: Segmented road surface from the original point cloud data set.

should be moved to those locations to match the road surface height. The arrow points to a location that has a greater surface height resulting in a slice with a larger volume.

Figure 4.16 shows the cumulative volume in the road parallel direction. At the southern side of the road, 542.22 m3 needs to be excavated compared to 462.35 m3 on the northern side of the road. Because the studied road is not straight, the road parallel direction distances for the two road sides differ. The northern side is 137.2 m long, whereas the southern side is 124.1 m long. At the location indicated by the arrow in both Figures 4.15 and Figure 4.16, there is one slice with a particularly large volume which causes a steep rise in the cumulative volume. As shown in the cumulative volume estimation results, the material volume that needs to be excavated on the

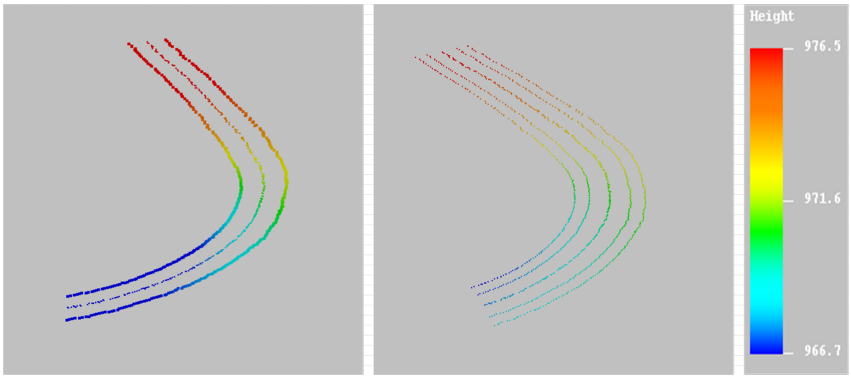


Figure 4.13: Road outline, central line and expanded road outline. (a) Road central line and road outline. (b) Additional expanded road outline.

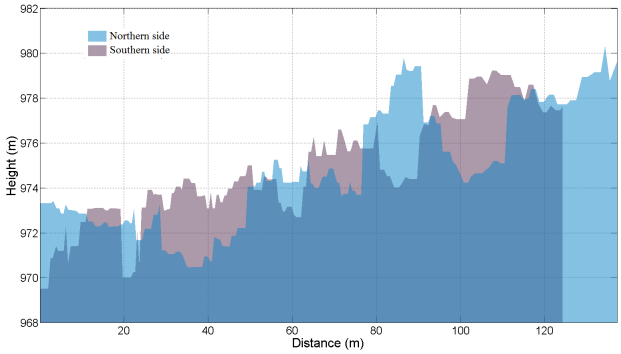


Figure 4.14: Profiled height on the expanded roadside. The light blue and light brown correspond to the northern and southern road side.

southern side is 8% greater than on the northern side. When applied in real applications, such results may help engineers to optimize road widening design to minimize the time and costs of the project.

Roadside points segmentation

Following the methods described in Section 4.2, the point cloud was filtered and voxelized using a uniform width of 0.1 meter. Then the point cloud was segmented and decomposed into three parts: road points, northern roadside and southern roadside points. This is illustrated in Figure 4.17. The points in blue are road points, while the points in red and green are the northern and the southern roadside points respectively.

4.3.4. Catchments estimation results

Before the application of the D8 method to obtain the catchments from the roadside slopes, a uniformed size grid was generated from the point cloud data. In this work,

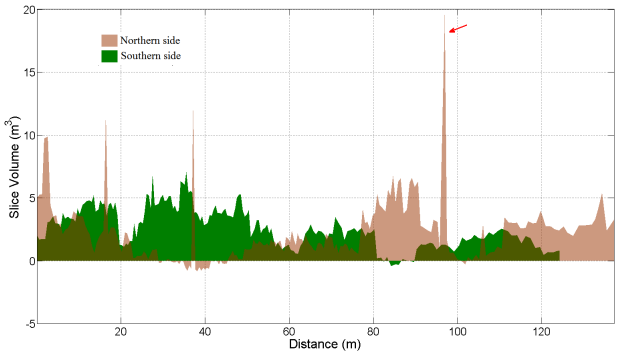


Figure 4.15: Slice volume computed from the expanded road outlines on both sides.

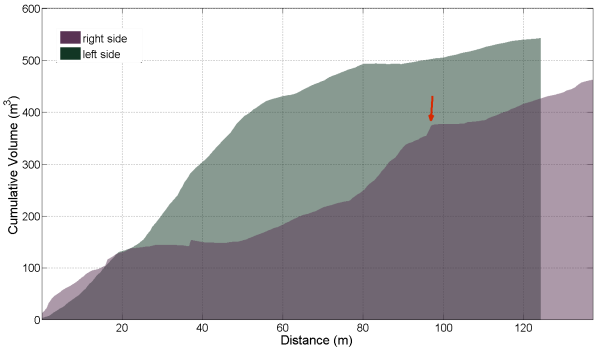


Figure 4.16: Cumulative volume on the extended roadside.

the grid size was preset to 2.0 m. The on-road water flow directions are estimated, as shown in Figure 4.18. In the figure, the flow directions are denoted by arrows. Using the D8 method, the road is divided in catchments, which are indicated in Figure 4.18 by different colors. Dark cell have no outflow.

After the flow directions on the road were determined, the flow directions for off-road point cloud data are also estimated, as shown in Figure 4.19.

In this figure, the sinks are in gray and all cells eventually flowing to the same sink are colorized by the same color. Each sink is labeled by a digit. The number of grid cells having runoff to each labeled sink in Figure 4.19 is given in Figure 4.20. There are 25 sinks on the southern roadside and 29 on the northern roadside respectively.

The results shows for example that the sink labeled as No. 15 on the north roadside, has 44 contributing cells, which indicates that this sink has a lot of potential water inflow. Comparison to the original terrain model in Figure 4.10b shows that this sink is actually located directly below the landslide area also shown in Figure 4.10c. The loca-

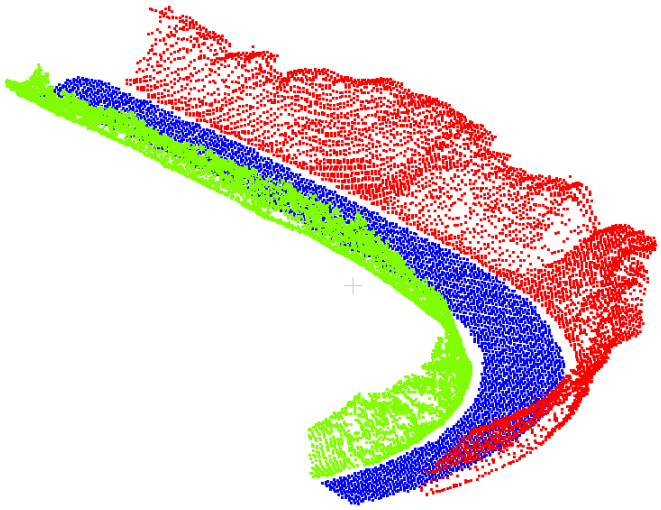


Figure 4.17: Roadside points segmentation result.

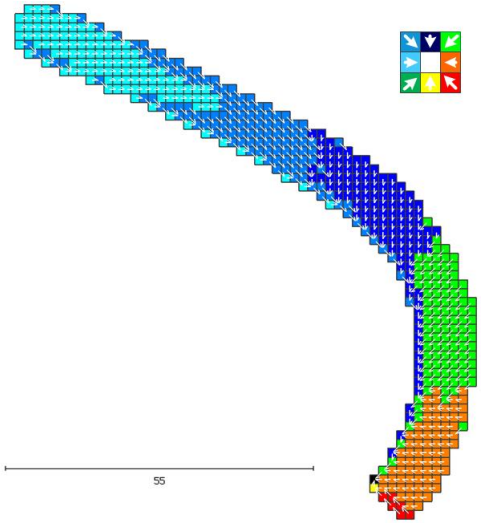


Figure 4.18: Road water flow directions on each grid cells.

tion of this sink is indicated in Figure 4.10a by a green ellipse. The shape of the terrain at this location is indeed such that more water is expected to accumulate. On the other

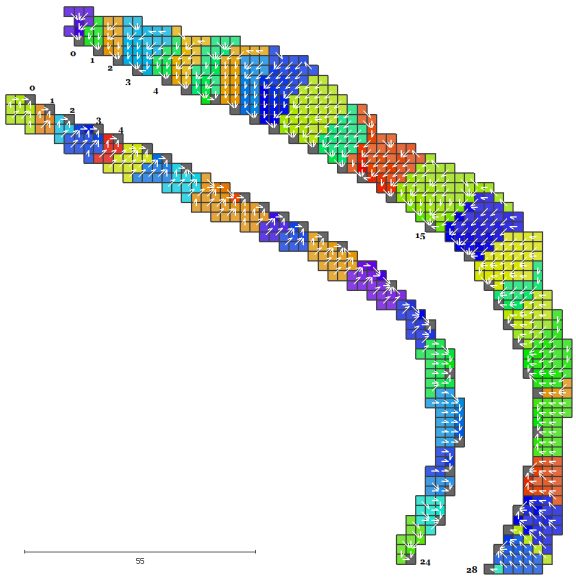


Figure 4.19: Labelled roadside catchment area.

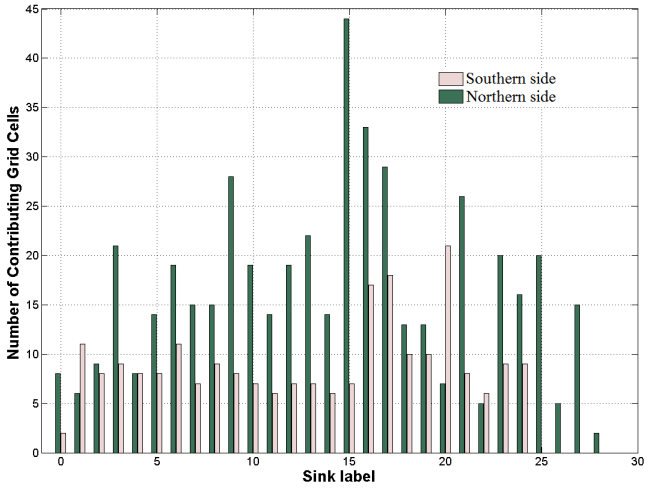


Figure 4.20: Count of catchment area cells for both roadsides.

hand, the sink labeled as 26 has only 5 contributing grid cells. And indeed, at this location, the roadside is very steep and water flows directly on the road. In Figure 4.21, the amount of saturation of the grid cells corresponds to the flow accumulation. That

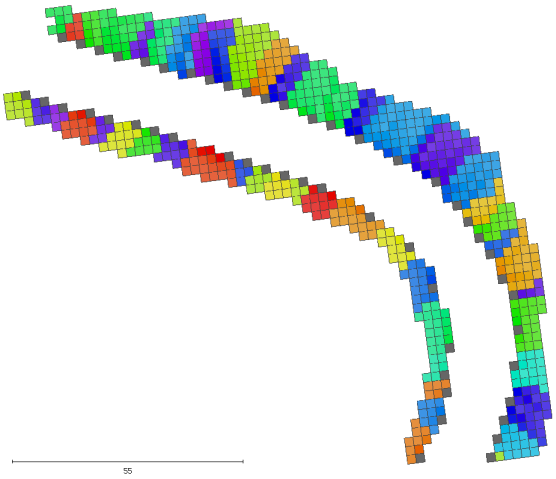


Figure 4.21: Accumulated inflow of each grid cell.

is, a cell with a high color saturation collects water from many cells. This also denotes water flow direction.

4.4. Quality discussion and validation

There is no data available from other sensors that can be used to verify the computed results. Instead the quality of the results is analyzed by considering the quality of the input data in combination with an analysis of how this quality propagates into the final volume computations. In addition, the excavation volumes were determined from a second MLS data set, acquired in a second run by the same system on the same day. Moreover, a possible measurement plan for further validation of the results is sketched.

4.4.1. Discussion on the quality of the results

Since the total volume is computed by summing up slices, the squared total error equals the squared sum of the errors in the determination of each sliced volume. The random error in the computation of a sliced volume consists of a variance component caused by random measurement errors in the original point cloud. This component is denoted as σ_{PTS}. Another variance component corresponds to the surface roughness and is denoted σ_R. Using the law of error propagation, the relationship between these errors is given by Equation 4.9.

$$\sigma^2_{Total} = \sum_{i=1}^{k} \sigma_i^2 = \sum_{i=1}^{k} (\sigma^2_{i,pts} + \sigma^2_{i,r}) \tag{4.9}$$

Here σ_{Total} is the total error of the volume computation, σ_i is the random error in the estimation of the volume of the *i-th* slice, while $\sigma_{i,pts}$ and $\sigma_{i,r}$ denote the point cloud

measuring error and roughness of slice i respectively, k is the number of the slices volumes, here equal to 132.

Thus, the error of the volume of a single slide is studied first. According to the specifications of the Lynx MLS and previous error studies (Puente et al., 2013b), the range precision and range accuracy is 8 mm and ±10 mm, respectively.

As can be seen in Figure 4.22, such single slide is divided into 1-m by 0.5-m blocks in the road parallel and the road perpendicular direction, respectively. In each block, the mean and standard deviation of the points in that block are computed. A slice from the north side slope of the road was randomly selected to compute mean and standard deviation of points in each block of the slice. A side view of both the original and the down sampled point cloud is shown in Figure 4.23.

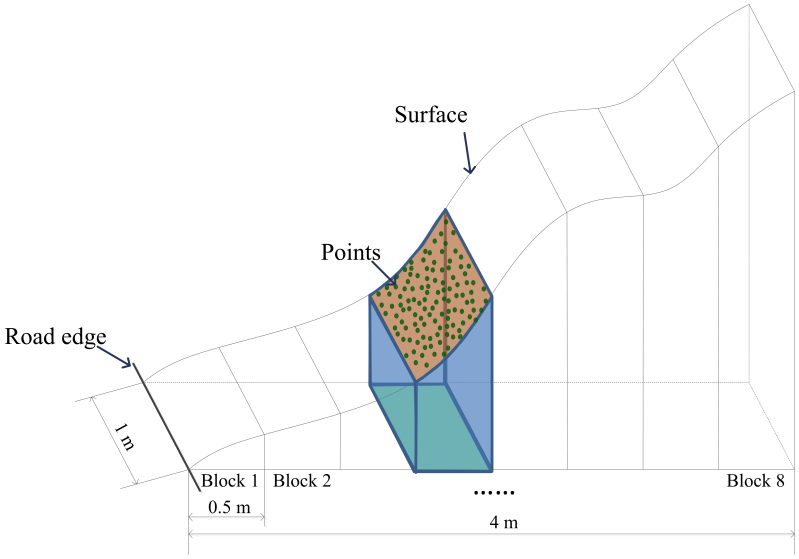
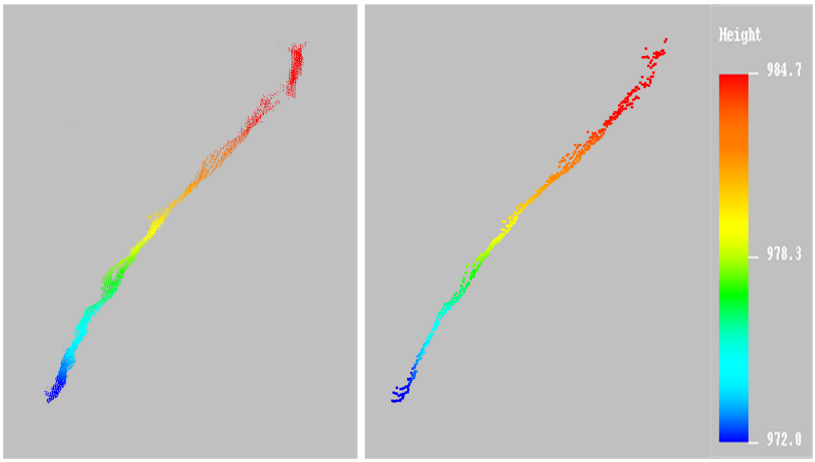


Figure 4.22: Single slice volume computation error analysis.

The resulting standard deviation (std.dev.) values for the eight blocks that together form the slice depicted in Figure 4.10 are given in Table 4.1. The average number of points per block is reduced from 488 to 36. This table also clearly demonstrates the purpose of the down-sampling strategy: close to the road, point density is very high and therefore the reduction in the number of points is high as well. Further from the road, the point density drops and a much larger fraction of the original point is maintained. On top of that, the geometry of the terrain with regard to the lasers on the car has a strong influence on the point density.

To summarize the results from Table 4.1, we determine the differences between the means per block from the original data and the reduced data. The mean of the absolute differences equals 0.18 m. Further validation is needed to verify which means are actually better: the means from the original data are computed based on more points, but some parts of the surface may also be overrepresented in the original point cloud due to local variations in scanning geometry induced by local relief variations. For both the



(a) (b)

Figure 4.23: Side view of randomly selected road side slice. (a) Road side points from original data. (b) Road side slope points from down sampled data.

Dataset		block 1	block 2	block 3	block 4	block 5	block 6	block 7	block 8
Original points	number	839	1070	571	284	200	175	354	415
	mean	972.84	974.94	977.57	979.20	981.26	982.70	984.62	987.02
	std.dev.	0.53	0.71	0.50	0.38	0.40	0.46	0.62	0.69
Down sampled points	number	45	33	33	38	35	40	36	31
	mean	972.43	975.03	977.24	979.19	981.31	982.76	984.54	986.60
	std.dev.	0.51	0.59	0.59	0.57	0.48	0.56	0.65	0.75

Table 4.1: Mean and standard deviation of points per block in meters. First three rows: original point cloud; Last three rows: down-sampled point cloud.

original and the reduced blocks, the std.dev. values are comparable, between 0.5 and 0.6 m. These std.dev. values are larger than the absolute differences between full and reduced data, and also much larger than the quality of the individual points. Therefore, it is concluded that these values are dominated by surface relief which is also clear from Figure 4.23.

Assuming a std.dev. value per block of 0.55 m, the std.dev. per slice equals 1.56m. Assuming 132 slices, this results in a std.dev. for the total volume on one road side of 17.9 m. This std.dev. value corresponds to an error below 4%, when compared to a value of 500 m3 of total excavation volume. As the current error is dominated by surface relief, a reduction in the error could be obtained by decreasing the block size.

4.4.2. Validation using data from a second run

For validating the results shown in Section 4.3, in this paragraph the same method will be applied to a second data set obtained using the same LMMS on the same day. The differences in outcome will be compared to as discussed in Section 4.4.1.

Description of the point cloud obtained in the second run

For the second run, the same system was used but the position of the car on the road was different, as will be shown below. As for the first data set, the data of the second run consists of a geo-referenced point cloud and of a data set giving the trajectory of the LMMS car during data acquisition. The point cloud of the second run cropped to the same piece of road consists of 6,374,830 points, and has a point density of 2,000 points per square meter. A side view of the second run point cloud is shown in Figure 4.24.

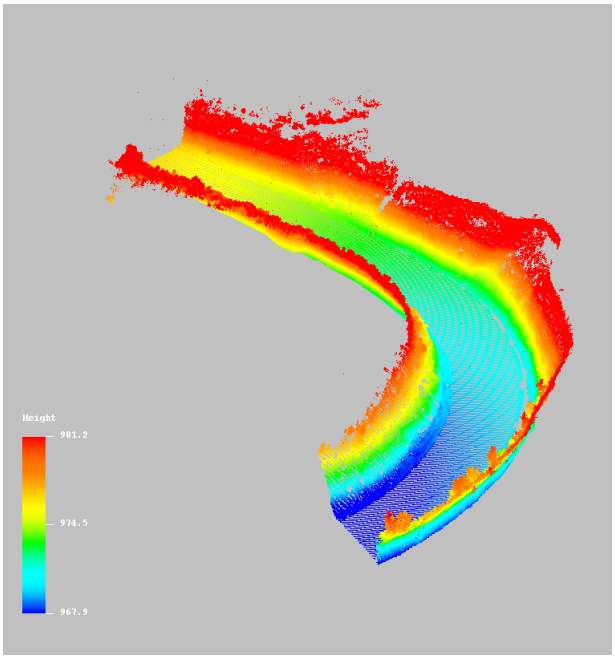


Figure 4.24: Side view of point cloud data from the second run.

Computation results

Following the same methodology as described in Section 4.2, the data of the second run was processed, and the excavation volumes for both road sides were computed. The results are shown in Figure 4.25.

A comparison of the results from both data sets is given in Table 4.2. The results show that the difference in excavation volumes for both sides of the road are within the error budget as derived in Section 4.4.1, which was determined as 4% of the total excavation volume.

Comparison analysis

As can be seen from Table 4.2, there are some differences in the excavation volumes as computed from the original point cloud data and the data from the second run. Recall that volumes are determined from 1 m slices that are further divided in eight

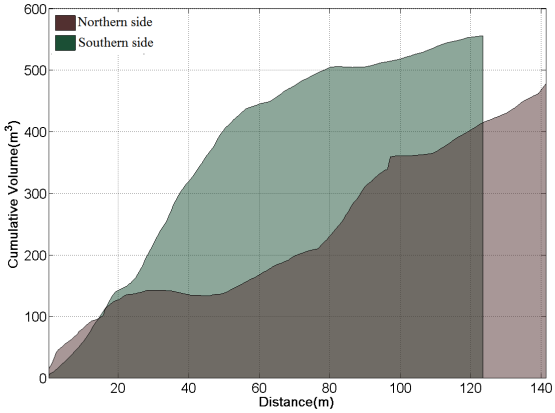


Figure 4.25: Cumulative volume of a roadside extension determined from point cloud data of the second run.

	Southern roadside	**Northern roadside**
Original data (m^3)	542.2	462.3
Data from second run (m^3)	556.3	478.2
Difference (%)	2.53	3.32

Table 4.2: Comparison of excavation volumes determined from original data and data from second run.

blocks, comparing Figure 4.22. To obtain insight in the differences between the outcomes from the first and the second run, Figure 4.26 shows differences in height per block of 1.0 meter in road parallel direction and 0.5 meter in road perpendicular direction. The dotted red line is the abstracted center line of the studied road. The purple and chocolate line in Figure 4.26 depict the trajectories of the LMMS while collecting the original and the second run point cloud data, respectively.

As shown in Figure 4.26, most of the blocks have approximately the same height, which demonstrates that the two data sets are consistent. Only the purple circles indicate locations where local height differences in the order of 1–2 m occur. Examining the two point clouds in detail indicates that at those locations hardly any points were sampled in one of the two runs. This local under sampling is probably caused by limited visibility of the roadside from the location of the LMMS acquisition.

This effect is illustrated in Figure 4.27, which shows a schematized cross section roadside geometry. For the trajectory 1, the purple area is invisible from the LMMS and is therefore not sampled. However, the area can be scanned from trajectory 2. A good solution would be to combine data from both runs such that the two point clouds data can supplement each other in such situations.

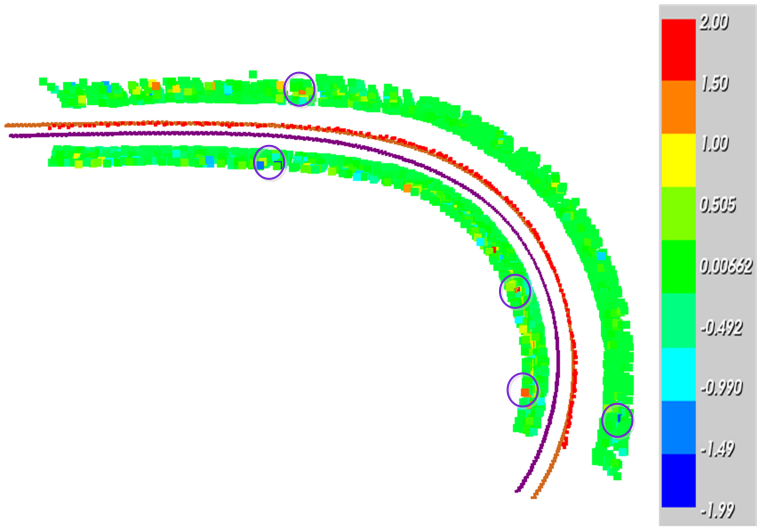


Figure 4.26: Height difference per block between original and second run point cloud data (meter).

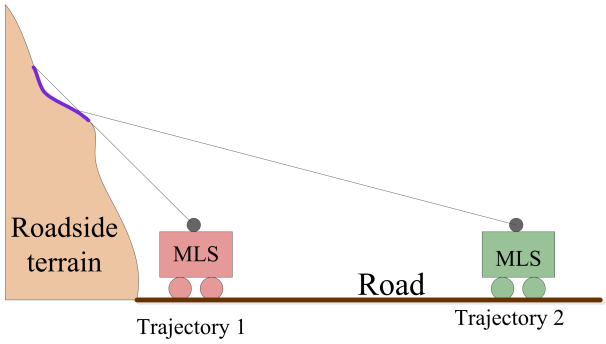


Figure 4.27: Geometry relation between LMMS and steep roadside terrain.

4.4.3. Proposal for further validation

There are several options to further validate the results of the methodology proposed in this chapter in a field experiment. A general idea is to locally use other, preferably superior measuring methods to sample the geometry of a piece of the road and road side considered, and repeat the computations with these superior data. A traditional method would be to use a total station or RTK-GPS to measure some profiles of 3D road surface points in a local geo-referenced datum, and import the obtained data into modeling software such as AutoCAD or 3ds Max, to construct a local road model and compute the volume. This method should give accurate results, but is labor intensive. A total different approach would be to actually perform measurements directly before a planned road extension. In this way, the real volume of the material that is excavated

can be measured and compared to the results of the analysis of the corresponding MLS data.

4.5. Conclusions

This chapter answered the following research question:

How to estimate water flow directions and excavation volume on mountain roads from MLS point clouds?

In this chapter, a method is proposed for the estimation of the excavation volume of a planned road widening from a MLS point cloud. Starting with a MLS point cloud data sampling a mountainous road, we used a uniform-size voxel to down-sample the point cloud data and remove outliers. Then, local normals and 2D slopes were estimated at each resulting grid point to separate road from off-road points. Finally, the volume needed to excavate the road by 4 m on both sides was computed. It was shown on MLS data representing a mountain road in Spain that the volume to be excavated on the southern side differs by 8% to that on the northern side. A more detailed analysis of one slice of data indicates that the error in the estimated excavation volume is below 4%. The results were partly validated by a comparison to results from analyzing a second point cloud obtained by the same system on the same day, but from a different trajectory. The resulting excavation volumes as estimated from both data sets differed by 2.5%–3.5%.

A further step would be to use the proposed method for determining the widening of the road of, e.g., 4 m by x meters on the northern and $(4-x)$ meters on the southern side, with $0 \leq x \leq 4$, that minimizes the moved volume over a stretch of, say, 100 m of road. Further research is also needed to determine an optimal block size: In this chapter, blocks of size 0.5 meter by 1 meter are used; reducing the block size will decrease the effect of surface relief on the error, but will increase the effect of measurement noise and varying point densities.

Since mountainous roads have complicated morphological environments and face threat from landslides and rock fall, there is a need for road and road environment safety inspection and monitoring. To meet this obligation, detailed and continuous road environment surface flow modeling has to be acquired. MLS can acquire point clouds in an efficient way, both from a time and costs perspective. For this reason we have presented a method to estimate roadside properties, which are the gradient and slope, and then the catchments on the roadside slope are computed with the D8 algorithm. The number of cells in each catchments is a measure for the amount of water flow into the corresponding road surface location.

In this thesis, the cell size was set to 2 meters only for the feasibility demonstration of the D8 method in the catchments and runoff estimation. But for practical and high quality purpose, the resolution could be much higher, e.g. up to 25 cm, as long as the point cloud density in the original data is high enough. Also, to validate the catchment estimation results, other data sets could be introduced, like airborne laser scanning data, total station surveying or GNSS profiling of the terrain.

To evaluate the results, other Geography Information System (GIS) software could be used to evaluate the flow direction and compare the results. A future work would be the monitoring of the sink locations, and to inspect if local road erosion is corre-

lated with the size of the inflowing roadside catchment. Note that the D8 method as presented here, requires a non-trivial slope. That is, if the surface off or on the road is locally flat, the method would be stuck. A possible solution is to take the expected speed and direction of water flow into account.

5
Urban and Roadside Tree Individualization

Roadside trees in urban areas are an important component of a community and are efficiently sampled by MLS systems. For monitoring and inventory purposes, the trees need to be individualized. This chapter presents an adjacency analysis based algorithm for roadside and urban tree individualization using voxels from MLS point clouds. First the problems is further specified in the introduction and existing methods are discussed. Then, the newly developed methodology is described in detail. Next, the methods is tested and validation of the results is given.

5.1. Introduction

Trees play an indispensable role in the urban environment and tree management is of great interest for biomass estimation and monitoring environmental changes (Cottone and Ettl, 2001; Zheng et al., 2007; Van Deusen, 2010; Moskal and Zheng, 2012). Traditionally, trees are manually measured in situ, which is time-consuming, costly and susceptible to subjective errors. Besides, adverse site conditions can make access difficult (Hopkinson et al., 2004).

LiDAR has become a well established surveying technique for the acquisition of geo-spatial information (Vosselman and Maas, 2010). Laser scanning uses powerful highly collimated laser light as a probe (Cifuentes et al., 2014). The laser pulse interacts with the measured object and is partly back-scattered to the detector which enables it to acquire the distance between the sensor and illuminated spot (Dassot et al., 2011). Laser scanning obtains dense 3D point clouds, which provide a comprehensive geometrical description of object. Taking advantage of the high scanning speed of LiDAR, dense raw point cloud data of a whole area can be collected in a short time. Combined with automatic point cloud processing techniques, this in principle enables the efficient extraction of geometric tree parameters. In recent years, many studies have investigated the application of LiDAR implemented in airborne laser scanning (ALS), mobile laser scanning (MLS) and terrestrial laser scanning (TLS), for tree and forest applications. A typical processing flow consists of three steps: (i) separate tree points from non-tree points; (ii) identify individual trees among all tree points; and (iii) estimate parameters describing individual tree geometry.

Nowadays, large scale urban tree inventories call for flexible and efficient methods to segment tree points from raw point clouds. Identifying tree points in raw point clouds is a fundamental step in tree modeling using laser scanning. Many algorithms have been developed for detecting and classifying trees from point clouds captured by different sensors. A conventional method is first to segment non-terrain points and then extract all tree points based on the height distribution of the points (Vosselman, 2000; Axelsson, 2000; Sithole and Vosselman, 2004; Kraus and Pfeifer, 1998; McDaniel et al., 2012). This methodology is robust and digital elevation models (DEM) and digital surface models (DSM) can be generated in parallel. Other approaches are region growing (Pauling et al., 2009; Aijazi et al., 2013), feature based tree classification (Lin et al., 2014; Strom et al., 2010; Rutzinger et al., 2010; Yang and Dong, 2013) and canopy model fitting method (Lahivaara et al., 2014). With the availability of small-footprint full-waveform LiDAR systems, algorithms like (Guo et al., 2011; Vaughn et al., 2012; Lindberg et al., 2014) are proposed to exploit the waveform features of the back-scattered waveform to classify tree points. Since more sensors can be integrated on the same

platform, like in (Guo et al., 2011; Vaughn et al., 2012; Karolina et al., 2013), hyperspectral and multi-spectral images are integrated in the classification and extraction of trees (Puttonen et al., 2011). There are also numerous methods available to extract tree points from MLS and TLS point clouds (Belton and Lichti, 2006; Rutzinger et al., 2011; Yang et al., 2012b; Zhong et al., 2013; Sirmacek and Lindenbergh, 2015; Wang et al., 2015).

Individual tree delineation, the second step in the processing chain, aims at separating single trees from the segmented tree points. Since this is the primary focus of this work, existing methods will be discussed in Section 5.2.

Tree modeling has been studied in fields like computer graphics, forestry and remote sensing, for various purposes. Tree parameters have been extracted and models of trees are reconstructed from ALS point clouds (Vosselman et al., 2004). Based on TLS point clouds, forest geometry has been reconstructed for canopy radiative transfer models (Bremer et al., 2015). Also, an automated workflow has been presented to extract 3D tree models from MLS point clouds (Rutzinger et al., 2011). An octree-based space division procedure was introduced to extract tree skeletons (Bucksch and Lindenbergh, 2008). In 2013, Tang et al. proposed an algorithm to reconstruct 3D surface of tree canopy from LiDAR point cloud (Tang et al., 2013). The method first obtains a stack of separated slices corresponding to different height levels. Next the acquired boundaries were combined to form canopies of individual trees. However, those methods are either not scalable and computationally expensive, or consider point clouds obtained from one type of sensors only. The algorithm we propose can deal with different situations and is demonstrated in several case studies involving data from ALS, MLS and TLS systems including challenging scenarios, like separation of trees on steep terrain and trees partly occluded by a wall.

This paper is structured as follows. Section 5.2 discusses existing methods of tree separation in ALS, MLS and TLS point clouds. Section 5.3 presents the newly proposed VoxTree algorithm for individual tree delineation. Then the results and evaluation of the algorithm are presented in Section 5.4. Section 5.5 gives concluding remarks and recommendations.

5.2. Related work and proposed innovations

Existing methods used for individual tree delineation are categorized into two classes, point based approaches and voxel based approaches. The first class deals with all tree points while the latter consider voxel cells containing points. A detailed review of the two approaches is given below.

5.2.1. Point based approaches

Many available algorithms already proved their feasibility to individualize single trees and estimate their parameters from ALS, MLS and TLS point cloud data. In 2006, Solberg et al. presented a method that first generates a canopy surface model from an ALS point cloud. Then based on the surface model single trees were segmented and characterized (Solberg et al., 2006). Based on the vertical distribution of the ALS point clouds, shapes of spruce and pine trees were constructed to individualize and discriminate

these two kinds of trees. Also tree position, height and crown diameter of the individualized trees were estimated (Persson et al., 2002; Holmgren and Persson, 2004). In 2001, Hyyppä et al. proposed to build a terrain model and a canopy model and then generate a 3D tree height model. Based on this tree height model, individual trees were extracted and also parameters, such as tree height, area projected on ground and stem diameter can be derived (Hyyppä et al., 2001). Gorte et al. identified 2D local maxima in 2D point densities to distinguish tree canopies from surrounding objects in 2009. Next, trees were delineated based on the obtained maxima (Rahman et al., 2009). Rutzinger et al. introduced an alpha shape approach for point cloud reduction and tree models are generated consisting of tree crown and a realistic trunk (Rutzinger et al., 2010). Alpha shape is a generalization of convex hull and is able to describe the shape of point entities, however, it also needs to tune the radius (Kirkpatrick and Seidel, 1983). Consecutively in a MLS scanned point cloud, trees are detected and their parameters are estimated by the method in (Weber and Penn, 1995). The accuracy of tree detection was 85% (Rutzinger et al., 2010). In their work the tree trunk and tree crown branch structure were considered. Although this method reduced the volume of the point cloud data and partly preserved the geometry of trees, this method introduced an extra step to determine the alpha shape which also affects its computational efficiency. In 2012, Li et al. presented a new region growing method to segment individual trees from ALS point clouds by taking advantage of the relative spacing between trees (Li et al., 2012). The method segmented 94% of the trees correctly. However, this method was only tested on sparse discrete ALS point clouds. In 2014, Vega et al. proposed an algorithm to segment single trees by obtaining the local maximal point in k nearest neighbor points in the 3D space. Points are processed from the highest to the lowest height value and points are assigned to corresponding tree segments (Vega et al., 2014). This algorithm was tested on three different forest types and 82% of the trees were successfully detected. In 2014, Duncanson et al. presented a method to delineate multi-layered crown for mapping individual tree structure by using a watershed-based canopy height model (Duncanson et al., 2014). The method could identify 70% of dominant trees. The method was able to determine tree parameters, such as tree height, crown radius and crown area. In 2014, Lu et al. developed a bottom-up approach to segment individual deciduous trees with leaf-off LiDAR point cloud data based on the intensity and 3D structure (Lu et al., 2014). The approach was tested on a forest and the results implied that 84% of trees were detected and 97% of the segmented trees were correct.

5.2.2. Voxel approaches

One way to speed up point cloud processing is to consider voxels rather than individual points. In 2008, Wang et al. presented a voxel based procedure to analyze vertical canopy structure of trees and to obtain 3D models of single trees sampled by ALS (Wang et al., 2008). The algorithm first re-samples the input point cloud to voxels and a series of horizontal 2D projected images at different height are generated in the voxel space. Then the main tree canopy layer and the height ranges of the layers are detected according to a statistical analysis of the height distribution of the normalized raw point clouds. Compared to point based methods, this approach improved on efficiency, but

does not consider the relationship between voxel cells. Bienert et al. introduced a voxel based method to analyze wind field models of a TLS scanned forest scenario (Bienert et al., 2010). The stems of trees are automatically detected before the 3D point cloud is translated into a voxel structure representing the forest. Then voxels are clustered based on a region growing concept and finally individual trees are interpreted.

Several algorithms were also designed to process point clouds using voxels, that did not consider trees so far. In 2013, Papon et al. presented a region growing algorithm to perform segmentation for point clouds (Papon et al., 2013). In their study, 3D over-segmentation of point clouds was conducted using the relationship between voxels and the results were then merged to ensure consistency with the spatial geometry of the scene. However, the algorithm was only tested on indoor objects scanned by a TLS. Aijazi et al. presented a method to classify 3D urban MLS point clouds based on voxels, but trees were not studied (Aijazi et al., 2013). Cabo et al. detected pole-like objects from MLS point clouds using voxel based methods (Cabo et al., 2014). The method first simplified the imported huge high density MLS point cloud by a regular voxelization. Then by assessing the local morphology of the voxel cells, trees could be classified and individualized. Nevertheless, the street trees are mainly of the same size and do not overlap much. Moreover, ALS and TLS point clouds were not tested yet. Babahajiani et al. presented an automated method to classify urban environments based on super-voxels (Babahajiani et al., 2015b). In this study buildings, roads, trees and cars were successfully classified. However, delineation of overlapping trees was not studied.

In 2013, Wu et al. presented a voxel based tree detection method to detect street trees from MLS point clouds (Wu et al., 2013). This method is computationally efficient and is able to extract single trees in a street scenario. However, this method focuses on MLS scanned point cloud data, separating trees from ALS and TLS scanned point clouds is not yet considered. Their method searches for trees starting from the bottom layer of a 3D grid, therefore it is unable to detect multiple-stemmed trees or trees that have their trunk occluded. Notably, this method shows its feasibility in separating street trees of similar size, which are along a road and connect mainly in the road direction. Urban trees which vary in size and are closely connected in different directions are not yet evaluated. Their method employed an incremental competing region growing algorithm proposed in (Liu et al., 2006) to separate touching tree crowns. The strategy takes the relative distance to tree centers into consideration rather than the actual connections between voxel cells. This introduces separating errors in case a bigger tree touches a smaller tree, as points of the bigger tree will be wrongly assigned to the smaller tree.

5.2.3. Proposed innovations

Our proposed voxel-based individual tree delineation algorithm is different from the previous methods with respect to the following points:

1. The attribution of the voxel cells to individual trees is based on a novel adjacency analysis, which makes the separation of connected trees more accurate. Notably in the case of connected trees that vary in size.

2. In the voxelization step, the size of the voxel cell is different in the three Cartesian

coordinate axes directions, which makes the algorithm more flexible in complicated scenarios.

3. A 3D clustering step clusters non-empty voxels and immediately separates non-connected components. This allows the algorithm to work efficiently on large data sets.

4. The individual tree delineation is performed both in a *bottom-to-top* and *top-to-bottom* scheme, which not only can separate trees that are occluded by a high wall and do not have their trunks scanned by the LMMS, but also verifies the results in two directions and makes the algorithm more reliable.

5.3. Methodology

The methodology presented in this chapter consists of six steps, as illustrated in Figure 5.1. The first step is pre-processing, which classifies tree points from the raw point cloud. Then the segmented tree points will be imported as input for the next steps. The second step is voxelization, which re-samples the imported tree points to voxel cells corresponding to preset voxel cell sizes. In the third step, connected voxel cells are clustered in 3D space. The fourth step is seed selection over all the clustered cells. Then clusters are individualized in the fifth step. The final step is to evaluate the overall individual tree delineation quality. In the next sub-sections the details of the proposed algorithm are given.

5.3.1. Pre-processing

In this step, tree points are classified and segmented from the original raw imported point cloud. Since this part is not the main focus of this work, this step is done with existing methods. The tree points can either be obtained by automatic filtering methods or with manual segmentation.

In this work the tree points are extracted in two steps. First, the imported original raw point cloud is classified as ground and non-ground points by using the algorithm presented by Kraus and Pfeifer (Kraus and Pfeifer, 1998). Then based on the non-ground points, the tree points are extracted by the algorithm presented in (Sirmacek and Lindenbergh, 2015).

5.3.2. Voxelization

Voxelization of a point cloud means to re-sample its points by voxel cells. The voxel cell designed for the proposed algorithm is a cuboid rather than a cube, which enables voxel cell to have different edge lengths in the three coordinate directions. The voxel cell sizes in the three directions are denoted by W_x, W_y and W_z respectively. The voxelization is performed as described in Section 3.2.4.

5.3.3. Cluster 3D connected cells

To save computation time and deal with trees that have big height differences, the connected cells are clustered after point cloud voxelization. In this study, a 3D seed filling

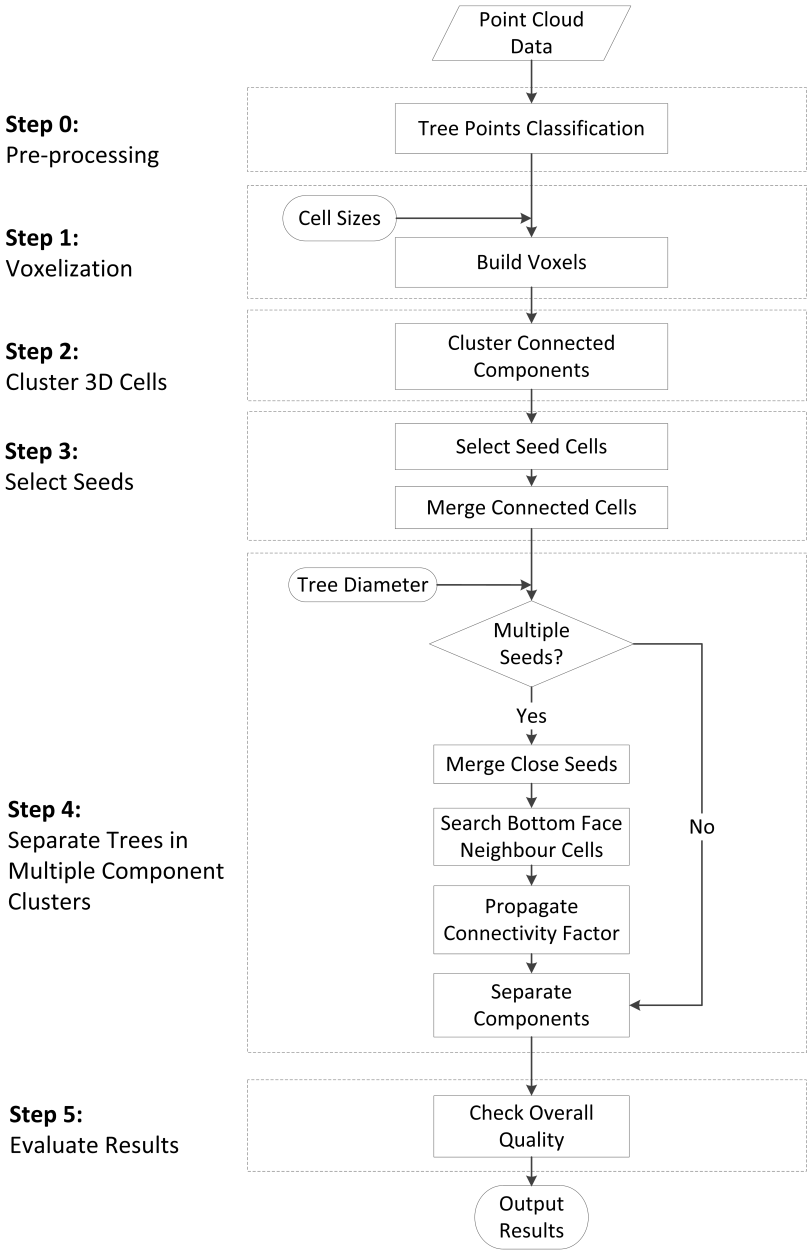


Figure 5.1: Overall methodology of the proposed VoxTree individual tree delineation method.

algorithm from (Yu et al., 2010) is employed to perform the clustering on all the *Positive*, as shown in Figure 3.5, cells before individual tree delineation. First all *Positive*

cells are labeled as *non-visited*. Then for the current cell, its neighbor cells among its 26 3D neighbors are obtained. All positive neighbors, that are *non-visited*, are pushed on a cluster stack. This step is recursively performed until all the positive cells are traversed. The output of this step are the connected cells, which are represented as clusters. In this step, all the cells are traversed only once. Thus the computation complexity of this step is linear, which is $O(N)$ in asymptotic notation (Cormen et al., 2001), if there are N cells.

5.3.4. Select seed cells

The separation of the clustered tree cells into individual trees starts with seed cell identification. A seed will potentially result in one individual tree after separation. Since the procedure of separation from the bottom layer upwards is similar to the separation from the top layer downwards, this section will only describe the methodology from the top layer downwards in detail.

Firstly, a cell is defined as a top cell if and only if this cell has a bottom-face neighbor but has no top-face neighbor cell. Here, bottom-face neighbor of a cell is the cell that connects with the bottom face of the query cell. For example in Figure 3.5, cell B is the bottom-face neighbor of cell C. While C is the top-face neighbor of cell B. Figure 5.2 depicts a scenario with two clusters that contain two connected trees and one individual tree. The colored cells are the initially identified seed cells of the two clusters respectively.

Next connected seed cells are clustered, resulting in the seeds S_1, S_2, S_3 and S_4 in Figure 5.2. Note that at this stage, one seed typically consists of a number of connected cells. The location of a seed is defined as the center of gravity of all the points inside the seed cells.

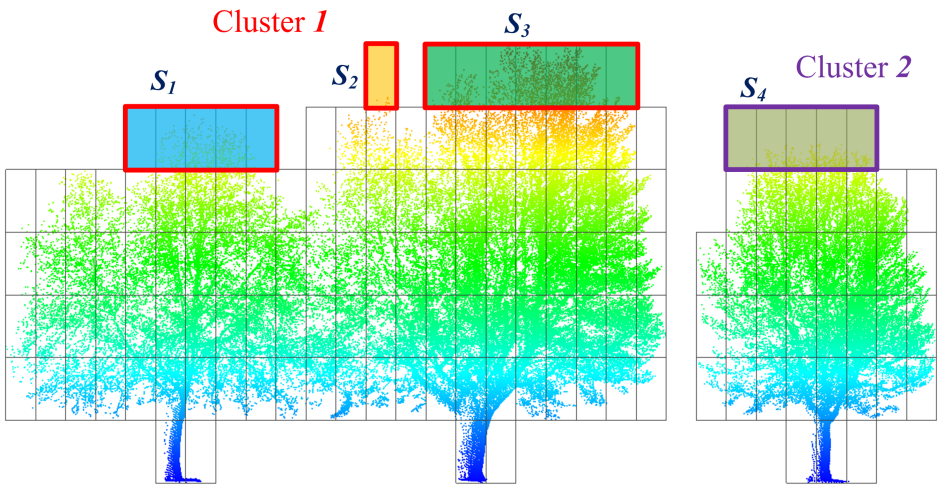


Figure 5.2: Connected cells are clustered as potential seeds of individual trees from the top layer downwards. S_1, S_2 and S_3 are potential seeds of Cluster 1 and S_4 of Cluster 2.

5.3.5. Tree separation

The tree separation procedure starts with the identified potential seeds in Section 5.3.4. The strategy described in this section is to separate trees starting from the potential seeds in the top layer of the cluster downwards to the bottom layer.

Detect multiple component clusters

This section categorizes the clusters as generated in Section 5.3.3 into multiple and single component clusters. A multiple component cluster is expected to have more than one tree. A cluster is considered as having multiple components if and only if this cluster meets the following conditions.

1. The cluster has at least two identified potential seeds.
2. The minimum distances between any two potential seeds are larger than the preset minimum tree canopy diameter.

If a cluster does not meet the conditions, it will be considered as a single component cluster and it is recognized as an individual tree. A detected multiple component cluster will be forwarded to the tree separation step, as illustrated in Figure 5.1.

Seeds inheritance

Merging close by seeds avoids separating one tree with several high branches into more trees. After the potential seed cells are identified, the separation of multiple component clusters starts with merging close by seeds. Figure 5.3 shows a side view of the scenario from Figure 5.2, which has two connected trees.

As the figure illustrates, the voxelization resulted in 7 vertical layers. The identified potential seed cells are firstly clustered and labeled as S_1, S_2 and S_3. Points P_1, P_2 and P_3 are the centers of gravity of the points in each of the seed clusters respectively. The horizontal distances between the centers of gravity of all the potential seeds are computed. As shown in Figure 5.3, the horizontal distances between the three selected seed cells are D_{12}, D_{23} and D_{13}, as computed between points P_1, P_2 and P_3. Close by seeds are merged in an iterative way. First the distances are sorted in ascending order and for the pair of closest seeds it is evaluated whether the horizontal distance is smaller than the preset minimum tree canopy diameter. In Figure 5.3, horizontal distance D_{23} is smaller than the preset minimum tree canopy diameter D_T, thus seed S_2 and S_3 are merged as one seed and labeled as S_2 as illustrated in Figure 5.4. Next distances between seed pairs are recomputed and the distance evaluation is repeated until no distance between seed cells is below the tree canopy diameter threshold.

After merging the close by seeds, their bottom-face neighbors are obtained that inherit the cluster index from their top-face neighbors. As illustrated in Figure 5.4, the bottom-face neighbors of seeds S_1 and S_2 in layer 7 are identified, which are colored in light cyan and orange respectively. Those cells inherit the cluster index and form the new seeds of layer 6. The same operation is conducted when traversing to the next layer downwards.

Similar operations are performed when traversing from the bottom layer upwards. As shown in Figure 5.3, the red cells are seed cells, and the same distance evaluation is conducted on the centers of gravity denoted by points P_4 and P_5. Then similar

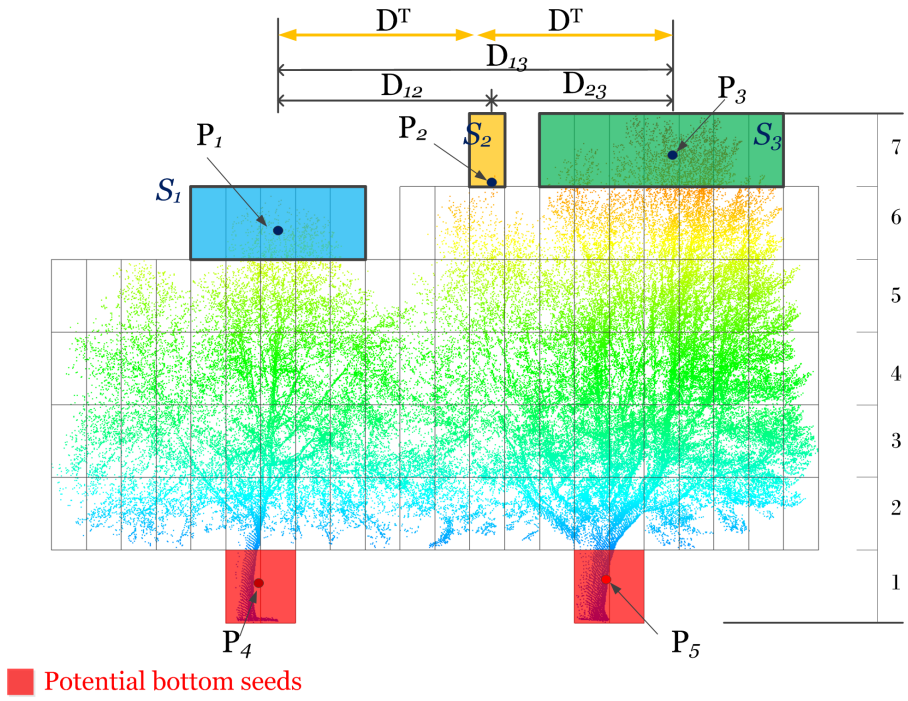


Figure 5.3: Seed cells identification and merging. The distance D_{23} is smaller than the preset minimum tree canopy diameter D^T, thus potential seed S_2 ans S_3 are merged together as one bigger seed.

seed merging and index inheritance operations are performed until the procedure has reached the top layer of the voxel cells.

Assign cells to individual trees

The objective of this step is to assign cells to individual trees. After seed identification and merging, seed indices are inherited by bottom-face neighbor cells and separation continues at the next layer of voxel cells. If at a given layer, cells are connected to only one seed cell, then the cells are assigned to the corresponding tree. However, the situation becomes more complicated when traversing to a layer where all cells are connected and have more seed indices inherited from the top layer seeds. As shown in Figure 5.4, layer 5 will have two inherited tree label indices from layer 6. To separate cells for those kind of layers, an adjacency analysis based cell assigning strategy is presented in this study.

Rather than simply considering the distances, connectivity based separation takes the neighborhood of the cells into account. Figure 5.5 is a top view of layer 5 as illustrated in Figure 5.4. The cells in cyan and orange have indices inherited from top-face neighbor seeds S_1 and S_2 respectively, as depicted in Figure 5.4. The striped orange and cyan filled cells are boundary cells of the two trees in layer 5 and are identified first. Boundary cells of seeds are defined as seed cells that have at least one unassigned cell among their 8 2D neighbors.

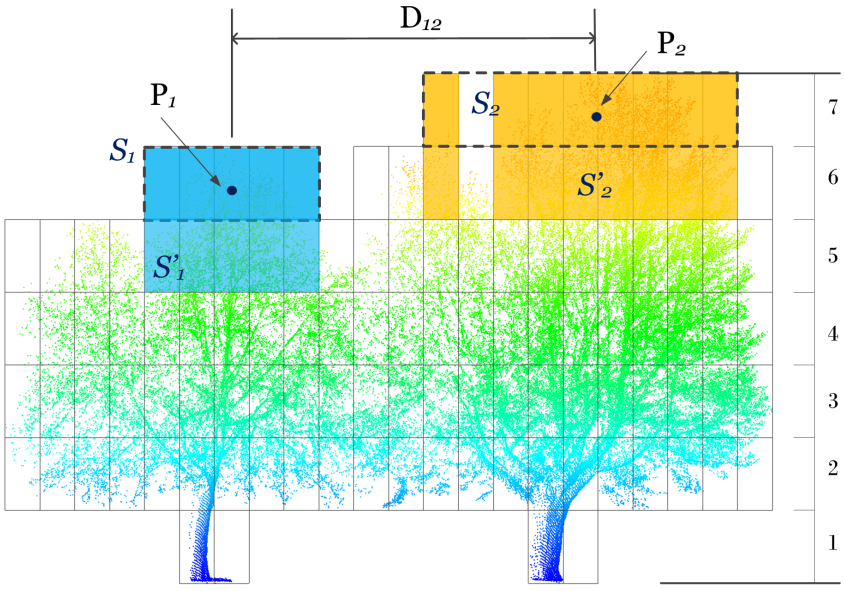


Figure 5.4: Inherit label of trees from top layer. Tree label of the lower layer cells, S_1' and S_2' are inherited from their top-face neighbor seeds, which are S_1 and S_2 respectively.

Unassigned cells will be assigned to a tree according to a connectivity coefficient. First, the connectivity coefficient is set to 1 for all boundary cells. Then the coefficient is propagated to its unassigned neighbor cells. The Connectivity coefficients are computed by Equation 5.1.

$$c(t,s) = C(t,s) \times R(k) \tag{5.1}$$

Here $c(t,s)$ is the connectivity coefficient of a $target$ cell with regard to a $source$ cell. $C(t,s)$ is the neighbor type of the $target$ cell with respect to the $source$ cell and is determined by Equation 5.2.

$$C(t,s) = \begin{cases} 0.50, & t \text{ is face-neighbor of } s \\ 0.25, & t \text{ is edge-neighbor of } s \end{cases} \tag{5.2}$$

A cell t is defined as a face neighbor of *source* cell t, if either t is sharing a face with s, or when all cells on the straight line connecting t with s are face neighbors of s as well. If a cell is not a face neighbor of a *source* cell, it is an edge neighbor of a *source* cell. For example in Figure 5.5, cell d, e, g, m, p and l are face-connected neighbor cells, while f, n, h and o are edge-connected neighbor cells of boundary cell A respectively.

$R(k)$ denotes the attenuation of the influence from the source cell to its unassigned neighbor cells and is defined as $R(k) = \frac{1}{k}$, with k the order of the cell with regard to the $source$ cell. The order of a cell with regard to a *source* cell is defined as the length in number of voxels of the shortest path connecting *source* and *target*. For example in

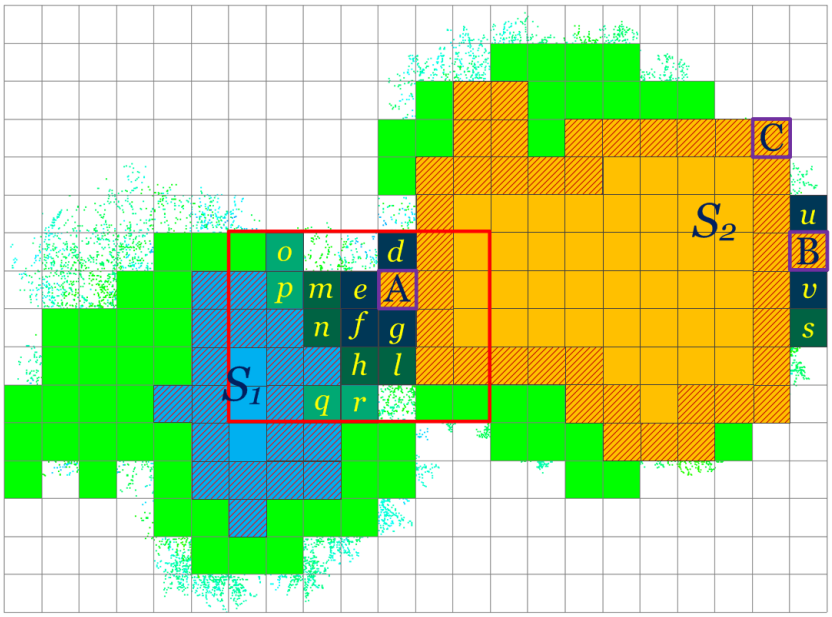


Figure 5.5: Connectivity based cells separation. Boundary cells of seed S_1 and S_2 are obtained will used to compute the adjacent coefficient to all unassigned cells in the same layer, such as d, e, f and g etc..

Figure 5.5, unassigned cells d, e, f and g are directly connected to cell A, so they are *1-st* order cells of A, so $k = 1$. Similarly cells m, n, h and l are *2-nd* order cells and $k = 2$.

All the boundary cells propagate their connectivity coefficient to the connected unassigned neighbor cells of different orders until there is no connected unassigned cell in the same layer. For example in Figure 5.5, cell B has cell u and v as its first order neighbors and cell s as its second order cell. After propagating its connectivity coefficient first to cell u and v, and consecutively to cell s, cell B has finished its connectivity propagation procedure. Cell C has no unassigned connected cells, so no connectivity coefficient with respect to C needs to be computed. After all the trees in one layer propagate their connectivity through their boundary cells to the unassigned cells in the same layer, the unassigned cells then have their accumulated connectivity coefficients corresponding to each tree. The unassigned cells are then assigned to the tree that has the largest accumulated connectivity value. After all the cells are assigned to an individual tree segment, then the points inside all the voxels are exported as individual tree points directively.

Example of connectivity coefficients determination

Figure 5.6 is a zoomed in display of the red rectangle area in Figure 5.5, which shows the accumulated connectivity coefficient values of each cell with respect to the boundary cells of the two trees S_1 and S_2. In Figure 5.6, the value in the upper right corner of each

cell is the accumulated connectivity coefficient value of tree S_2 while the value in the lower left corner corresponds to that of tree S_1.

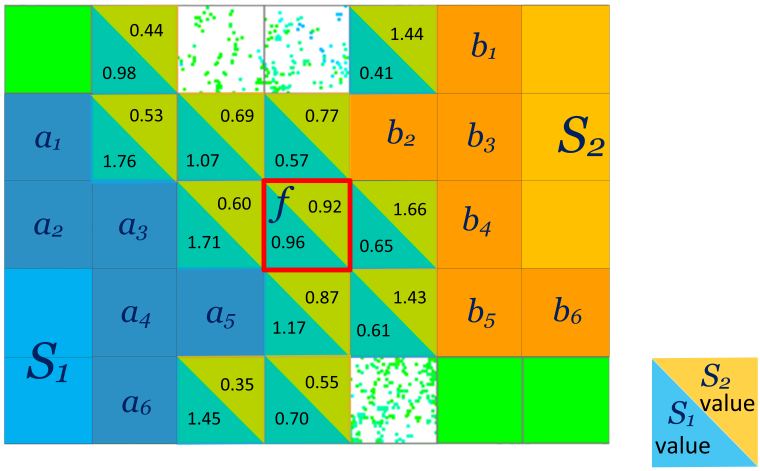


Figure 5.6: Computed adjacent coefficient of the unassigned cells. An unassigned cell will be assigned to a tree that has the biggest adjacent coefficient, such as cell f will be assigned to S_1.

The accumulated connectivity coefficient value of an unassigned cell to a tree in the same layer is computed by Equation 5.3. Note that in Figure 5.6 those values are only computed based on the boundary cells in the rectangle for demonstration.

$$c(t, S_j) = \sum_{i=1}^{n} c_i = \sum_{i=1}^{n} C(t, s_i) \times R_i \tag{5.3}$$

Here $c(t, S_j)$ is the accumulated connectivity coefficient of an unassigned cell propagated from all boundary cells of tree S_j, n is the number of the boundary cells of the tree, c_i is the connectivity coefficient value of the $i-th$ boundary cell, $C(t, s_i)$ and R_i are the neighbor type and the inverse order of the unassigned cell with respect to the $i-th$ boundary cell s_i. For example, the connectivity values of cell f in the red rectangle in Figure 5.6 to tree S_1 are computed as follows:

$$\begin{aligned} c_1 &= \{c_{a1} + c_{a2} + c_{a3} + c_{a4} + c_{a5} + c_{a6}\} \\ &= \frac{0.25}{3} + \frac{0.5}{3} + \frac{0.5}{2} + \frac{0.25}{2} + \frac{0.25}{1} + \frac{0.25}{3} \\ &= 0.96 \end{aligned} \tag{5.4}$$

Here c_{a1}, c_{a2}, c_{a3}, c_{a4}, c_{a5} and c_{a6} are the connectivity coefficients for boundary cells a_1, a_2, a_3, a_4, a_5 and a_6 respectively. Take c_{a1} for example: cell f is an edge-connected, third order neighbor of boundary cell a_1. The influence from cell a_1 is determined as $\frac{1}{3}$. Therefore the connectivity coefficient of cell a_1 with regard to f will be

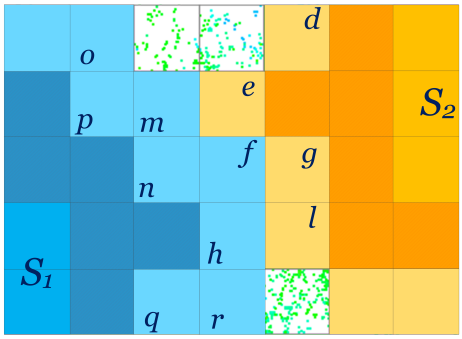


Figure 5.7: Assigning results of the cells. All unassigned cells are assigned to a particular tree as their color indicated.

$\frac{0.25}{3}$ according Equation 5.1. Similarly, the connectivity values to tree S_2 are computed as follows:

$$\begin{aligned} c_2 &= \{c_{b1} + c_{b2} + c_{b3} + c_{b4} + c_{b5} + c_{b6}\} \\ &= \frac{0.25}{3} + \frac{0.5}{1} + \frac{0.25}{2} + \frac{0.5}{2} + \frac{0.25}{2} + \frac{0.25}{3} \\ &= 0.92 \end{aligned} \tag{5.5}$$

Since $c_1 > c_2$ for cell f, it is assigned to tree S_1. The resulting cell assignments are illustrated in Figure 5.7. It can be seen that cell o, p, m, n, h, q, r and f in light blue are assigned to tree S_1, and cell d, e, g and l in light orange are assigned to tree S_2.

Full connected tree separation algorithm

The full procedure for separating connected trees is summarized in Algorithm 1.

In the algorithm, function DETERMINEBOUNDARY() is used to obtain the boundary cells of a 2D seed cluster. The function SEARCHNEIGHBORCELLS() is implemented to perform 2D neighborhood searching and function MAXTREECOEFFICIENT() acquires the index of the tree that has the largest connectivity coefficient for the considered cell. Suppose there are N cells in this cluster of connected trees, first all the cells are at least traversed once, then the unassigned cells are assigned to a correspondent tree.

After a cell is assigned to a tree, the cell is labeled as a component of the tree. After all the cells in the current layer are assigned to the corresponding trees, the indices of the cells are inherited by the bottom-face neighbor cells in the next layer downwards. This operation continues until the bottom layer of the voxel cell is reached.

Algorithm 1 Cell assignment in multiple seeds layer

Input: Tree seeds s and unassigned voxel cell w in the same layer
Output: Cells assigned to the correct tree

```
1: function AssignCells
2:     for each tree seed s_i do
3:         boundary cells b_i ← DetermineBoundary();
4:         for each cell v_j of Boundary b_i do
5:             neighbor cells n←SearchNeighborCells();
6:             while n ≠ empty do
7:                 for each neighbor cell n_k do
8:                     c_i+=C(t, s_i) × R_k
9:                 end for
10:            end while
11:        end for
12:    end for
13:    for each unassigned cell w_m do
14:        id←MaxTreeCoefficient(c);
15:        assign cell w_m to tree s_id;
16:    end for
17: end function
```


5.3.6. Overall quality analysis

This section describes the strategies that evaluate the individual tree delineation quality and the parameters estimation quality in this study. During the tree separation process, whether from top layer downwards or from bottom layer upwards, each individualized tree has a quality flag assigned to it according to the size of the tree. If the individualized tree canopy diameter is bigger or smaller than a preset threshold, the flag is set to 0; else the flag is set to 1. For those trees that were labeled as 0, another separation is conducted in the opposite direction. If this results in a valid separation result (i.e., with flag value 1), this result is then chosen as the final tree separation result. If both traversal directions result in a quality flag of 0, this result is given as output. Such negative results could be an indication for a human operator that further inspection is needed. However, for validation of the separation results, different strategies for different scenarios are given in Section 5.4.

5.3.7. Expected computation efforts

The computational effort of the proposed method depends on the size of the voxel cell and the complexity of the algorithm. Suppose the 3D bounding box of the tree points is D and the voxel cell size is d, then the number of cells is $N = (\frac{D}{d})^3$. Since the cells will be traversed at least once, a lower bound for the computational efforts is $\Omega(N)$. Also the complexity of clustering connected cells in Section 5.3.3 is $O(N)$. However, it is difficult to estimate accurately the complexity of tree separation since the connectivity coefficients determination depends on the number of boundary cells, which deviates in different scenarios. In this study an empirical computation time analysis will be

given in Section 5.4.2.

5.4. Results evaluation

To verify the flexibility and reliability of the proposed algorithm, five tests are performed: (i) separating the same trees sampled by different sensor systems; (ii) separating the same trees sampled by the same sensor, but with different voxel sizes; (iii) separating trees with occluded trunks; (iv) separating trees on steep terrain; and (v) separating trees connected in different directions. Finally the VoxTree method is validated and compared against manually processed ground truth data and an existing method (Wu et al., 2013) using a patch of 11 trees scanned by TLS. The results of these tests are discussed below.

5.4.1. Same scenario from different sensors

In this section, the flexibility of the VoxTree method is evaluated on a group of three trees, which were scanned consecutively by ALS, MLS and TLS. Those data sets sample two connected trees and one isolated tree. The details of the data sets are given in Table 5.1.

	ALS	MLS	TLS
Scanner	Unknown(van der Sande et al., 2010)	Fugro DriveMap	Leica-C10
Number of Points	3,640	112,957	2,346,740
Scanning Date	Before 2011.11.30	2013.11.23	2015.07.23

Table 5.1: Details of the three test data sets

Figure 5.8 shows three sets of point cloud of the same trees. Because of differences in scanning mechanism and acquisition geometry, point distribution and density are dramatically different. Compared to MLS and TLS, as illustrated by the small picture in Figure 5.8 (*b*) and Figure 5.8 (*c*) respectively, the ALS point cloud, as shown by the small picture in Figure 5.8 (*a*), consists of much less points. Note that the left two trees are connected and the right one is isolated.

The delineation results of the ALS, MLS and TLS point clouds are illustrated in Figure 5.8. In this test, the voxel cell size selected was 1.0, 1.0 and 2.5 meters in x, y and z direction respectively. The minimum tree canopy diameter was 7.5 meters and maximum tree bounding box size was set to 5 times of the minimum canopy diameter. The same voxel size was used for all three point clouds. The processing time for ALS, MLS and TLS data sets was 0.241, 0.428 and 0.626 seconds respectively. Note that although the number of points in the TLS point cloud is 644 times larger than that of the ALS data set, the processing time is only 3 times as much. As can be distinguished from the zoomed in areas in Figure 5.8, the two connected trees are well separated for all three data sets.

5.4.2. Same scenario, same sensor, different voxel sizes

To test the scalability in computation time with respect to the voxel size of the proposed VoxTree method, the algorithm is tested on the MLS point cloud with different voxel

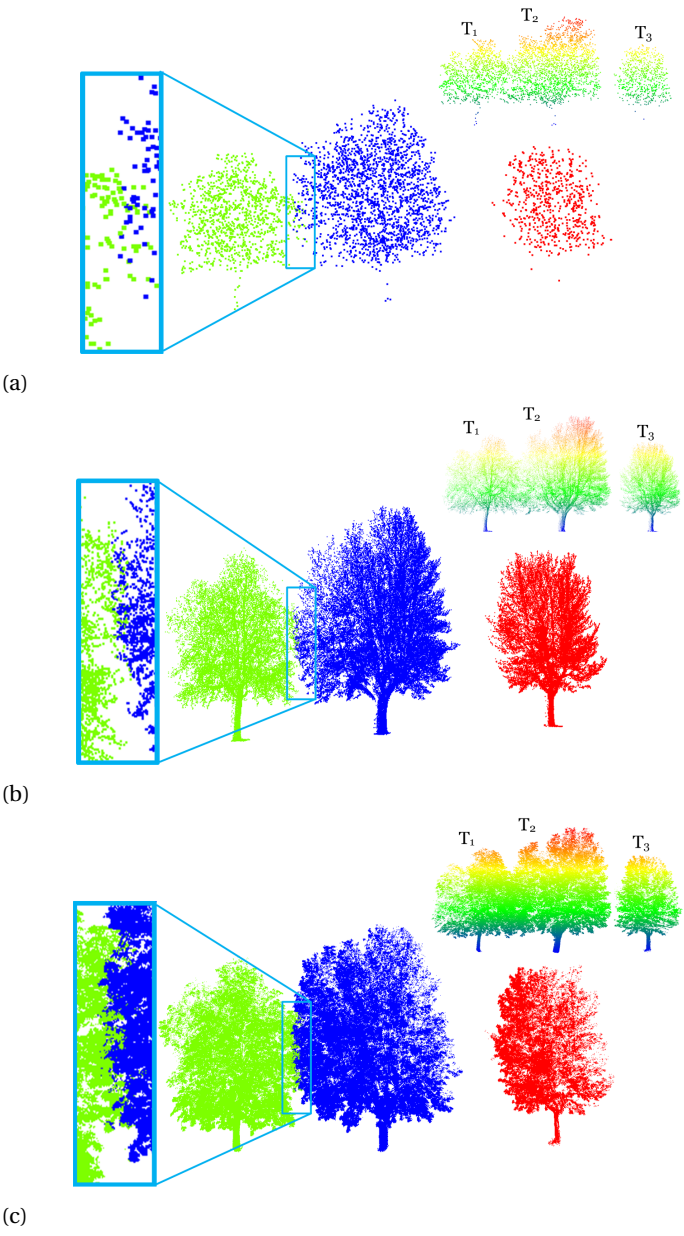


Figure 5.8: Original point clouds sampled by three different sensors and delineation results of the connected trees. (a) ALS point cloud. (b) MLS point cloud. (c) TLS point cloud

sizes. The tree separation results are illustrated in Figure 5.9. As can be noticed in the figure, there are four different voxel sizes used to individualize tree T_1 and T_2. From sub-figures (*a*) and (*b*), the horizontal resolution changed from 2 meters to 1 meter

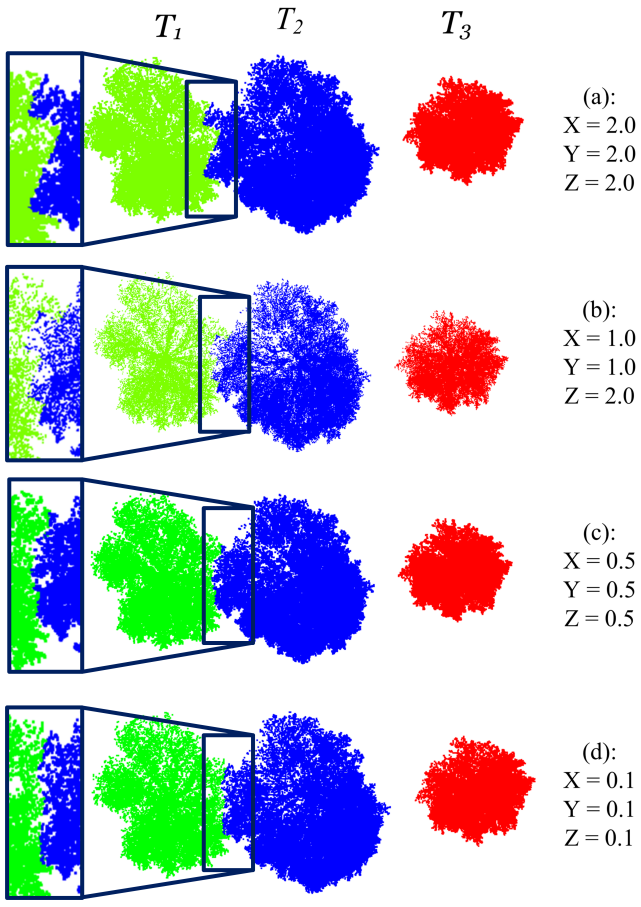


Figure 5.9: Tree separation results with different voxel cell sizes for point cloud sampled by MLS.

while the vertical cell size was kept the same. In this test, the minimum tree canopy diameter was set to $7.5m$ for all the four cases. Sub-figures (*c*) and (*d*) are individual tree delineation results with voxel cell sizes of 0.5 meter and 0.1 meter respectively.

Figures 5.9 (*a*) and (*b*) show that reducing the horizontal voxel size from 2.0 to 1.0 meter results in a finer point assignment in the area where the two trees appear to connect. Figure 5.10 indicates how the computation time increases with decreasing voxel sizes for scenarios (*a*), (*b*), (*c*) and (*d*).

In Figure 5.10, the horizontal axis refers to the number of cells for each of the four tests in Figure 5.9. The vertical axis denotes the processing time in milliseconds. The red line plots $Linear$ complexity results while the blue line is the actual processing time of the four tests. Although there are only four different cell sizes, the trend still verifies that the computation effort is largely in accordance with the theoretical analysis in Section 5.3.7.

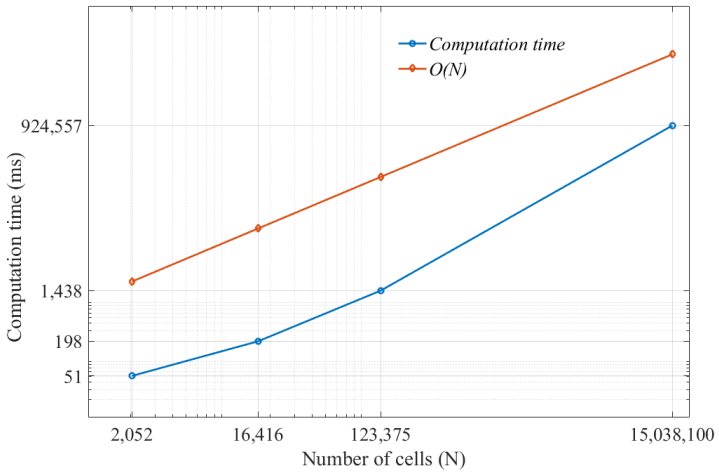


Figure 5.10: Computation time of the four tests.

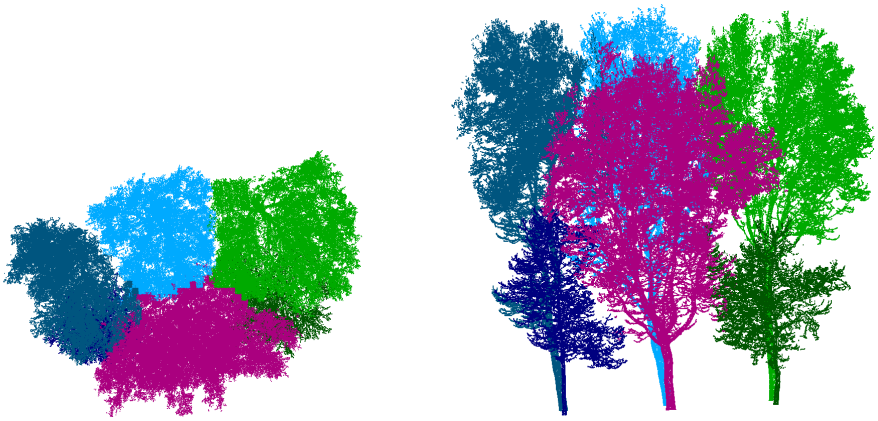
In order to further validate the capability of separating trees in different layers, six closely connected trees, which also consist of four tall trees and two understorey trees, were scanned by a TLS scanner to acquire a point cloud. In this test, the same processing steps were conducted and the voxel cell sizes in the three axis directions were all set to 0.5 m, while the minimum tree diameter was set to 3.5 m. The separation of those trees was conducted from bottom layer upwards and the results are illustrated in Figure 5.11. Figure 5.11a and Figure 5.11b give a top and a side view of the scenario respectively. As illustrated in Figure 5.11, all the six individual trees, including the two understorey trees, can be delineated successfully.

5.4.3. Trees having no trunk points

The proposed VoxTree method is also able to separate trees from both the top layer downwards and from the bottom layer upwards. This enables to individualize trees that are occluded by cars or a wall and therefore do not have their trunks scanned. Figure 5.12 demonstrates a scenario that has four MLS scanned connected trees.

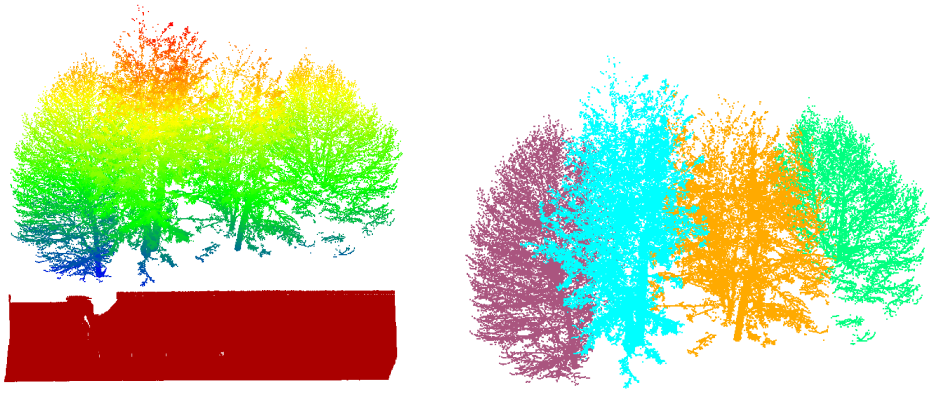
Figure 5.12 (*a*) is a front view of the original point cloud from the MLS. The figure depicts a scenario consisting of a wall and four connected trees. The trees are occluded by the wall and thus do not have their trunks scanned.

In this situation, the proposed algorithm first traverses from the bottom layer upwards which results in one big tree but also the quality flag of the separation is 0 as the size of the bounding vox is above the preset maximum tree diameter of $27.5m$. Therefore for this cluster the algorithm starts separating from the top layer downwards and finally separate the trees with a quality flag 1. The results of this delineation is given in Figure 5.12 (*b*). All the four trees are separated in a qualitative sense and visualized in four random colors. The voxel size used in this test is $40cm$ in the three directions and the minimum tree canopy diameter is set to $5.5m$.



(a) (b)

Figure 5.11: Individual tree delineation results from a group of six trees of different height layers. (a) Separation results from a top view. (b) Separation results from a side view.



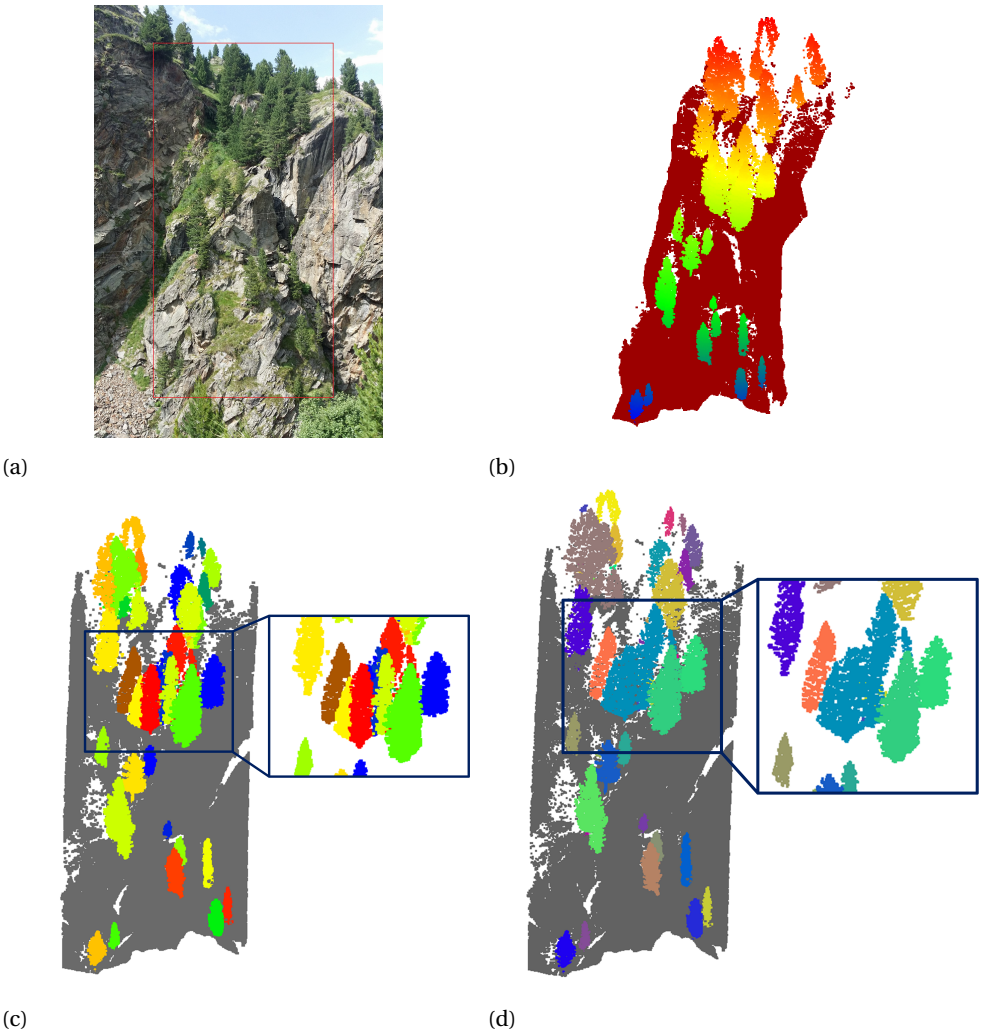
(a) (b)

Figure 5.12: Individual tree delineation results of the four occluded trees that have no trunk points. (a) Original MLS scanned point cloud of four trees occluded by a wall. (b) Individual tree delineation results of the patch of four connected trees.

5.4.4. Trees on steep terrain

This section demonstrates the capability of the proposed algorithm to individualize trees on steep terrain. Figure 5.13a shows steep terrain at Obergurgl in Austria. There are several spruce trees on the steep slope and some of them are connected. This area was scanned on July 7^{th} 2015, with a Riegl VZ-400 TLS scanner. The area in the red rectangle is selected for testing. The original point cloud of this area has 121,039 points and 30,732 tree points remain after segmentation. In Figure 5.13b, terrain points are

dark red while tree points are colorized by height from red to blue.



(a) (b)

(c) (d)

Figure 5.13: Individual tree delineation on a steep terrain at Obergurgl in Austria from TLS point cloud. (a) A picture of the testing area. (b) Original segmented tree points and ground points. (c) Individual tree delineation results of trees on this steep terrain area. (d) Tree separation results of Wu's method.

After segmenting the tree points from the original point cloud, the proposed VoxTree algorithm is applied. Figure 5.13c is the individualized trees in this area while Figure 5.13d gives the results from Wu's method (Wu et al., 2013). Ground points are dark red and individualized trees are in random colors. There are 36 individualized trees in total. As can be observed from the zoomed in area in the orange rectangle, the connected trees are correctly individualized. As Figure 5.13d shows, the existing method is unable to separate the trees of which have the same bottom height.

5.4.5. Trees connected in different directions

Another test is performed on a ALS point cloud data set, sampling trees connected in different directions. What's more, the size of the trees are different. The point cloud is a part of the AHN2 project of the Netherlands (van der Sande et al., 2010). The original

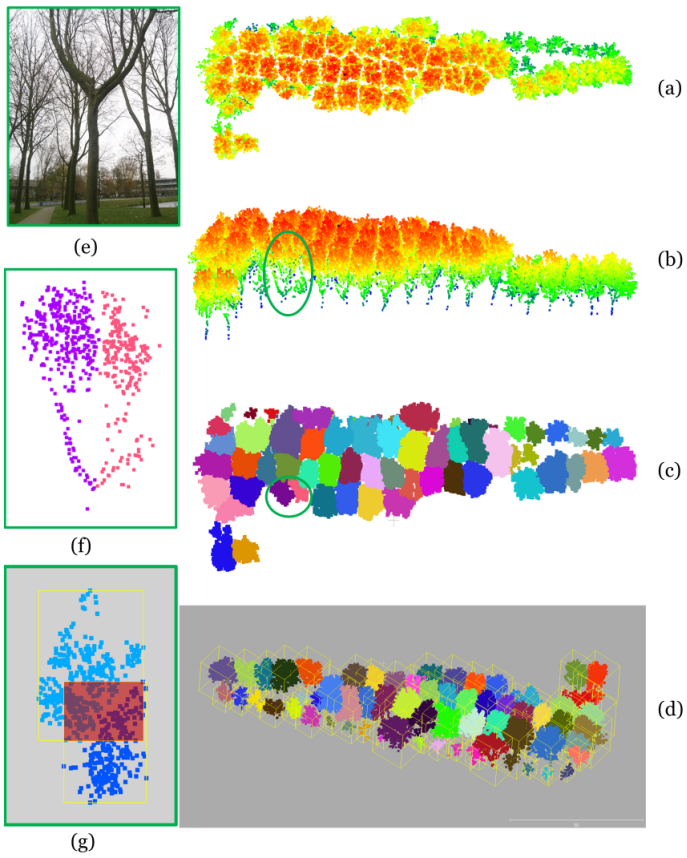


Figure 5.14: Original ALS point cloud and individual tree delineation results. (a) Original ALS point cloud from a top view. (b) Original ALS point cloud from a side view. (c) Results of individual tree delineation from a top view. (d) Bounding boxes of all individualized trees. (e) Picture of the tree in green circle in (b). (f) Resulted points of the tree in green circle. (g) Top view of the bounding box of the tree in green circle.

tree points are shown in Figure 5.14, (*a*) and (*b*). There are 36,005 points in total and the average point density in 3D is 5 pts/m^3. Manual in situ counting showed there are 63 trees in total. Because the point cloud is scanned from above, there are less points on the tree trunks than on the canopy.

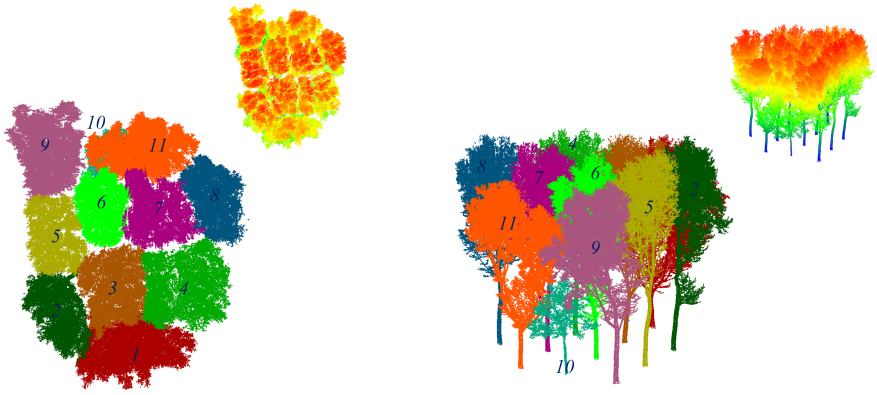
The cell sizes used in the algorithm were $20cm$ in horizontal direction and $40cm$ in vertical direction. The minimum tree canopy diameter was set at $6.5m$. The separation of the trees runs smoothly and the delineation results are given in Figure 5.14 (*c*), as a top view of the trees. 66 trees are identified by the VoxTree algorithm. Compared to

the real number of trees, 3 trees too many are identified. The sub-figures on the left are magnifications corresponding to the area indicated by the green ellipse in the main figures. This is one of the scenarios where the VoxTree algorithm splits one tree into two parts. The reason is that this tree has two big branches which are well-sampled by the ALS data, while at the same time its trunk is only sampled by one points. When either separating from the bottom layer upwards or from the top layer downwards, the two quality flags of the separation are both 1and therefore results in two trees.

Figure 5.14 (*d*) is a visualization of the individualized trees with their bounding boxes. The bottom left sub-figure is a magnification of the area in the green ellipse in Figure 5.14 (*b*) and (*c*). The lower left image is a top view of the 3D bounding boxes two touching trees. The red rectangle indicates the overlapping area of the two bounding boxes. A bigger overlapping area implies the two separated trees have large connected area. Since the errors in the delineation of connected trees mainly occurs in trees of highly overlapping occasions. This analysis suggests that non-optimal separation results can be found by considering the intersecting boundary boxes before a detailed inspection.

5.4.6. Cross validation with ground truth

To evaluate the accuracy of the proposed method for individual tree delineation, a patch of 11 connected trees was scanned with a Leica C10 TLS scanner. The points of those trees were manually separated and used as ground truth for the accuracy evaluation. The original point cloud and the segmented ground truth are shown in Figure 5.15.



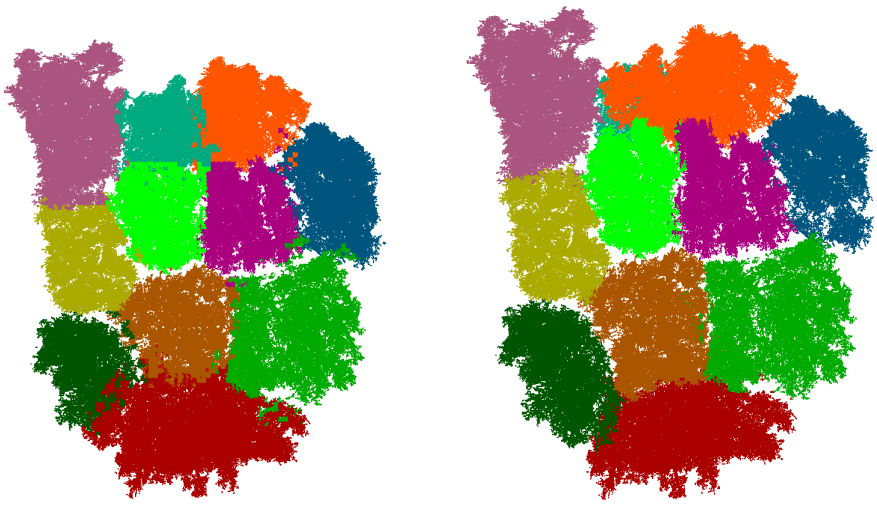
(a) (b)

Figure 5.15: Original TLS point cloud and manually separated ground truth of a patch of 11 connected trees. (a) A top view of the original TLS point cloud and the manually separated ground truth. (b) A side view of the original TLS point cloud and the ground truth separation.

The top right small images in Figure 5.15 (*a*) and (*b*) are the original point cloud from a top and a side view, The main images of Figure 5.15 (*a*) and (*b*) are the manually separated trees from two different view perspectives. Note that each of the 11 trees is

labeled.

In this test, the method proposed by (Wu et al., 2013) was implemented for comparison. For both methods, Wu's and VoxTree, the same voxel cell size of $30cm$ was used. The individual tree delineation results of the two methods are shown in Figure 5.16.



(a) (b)

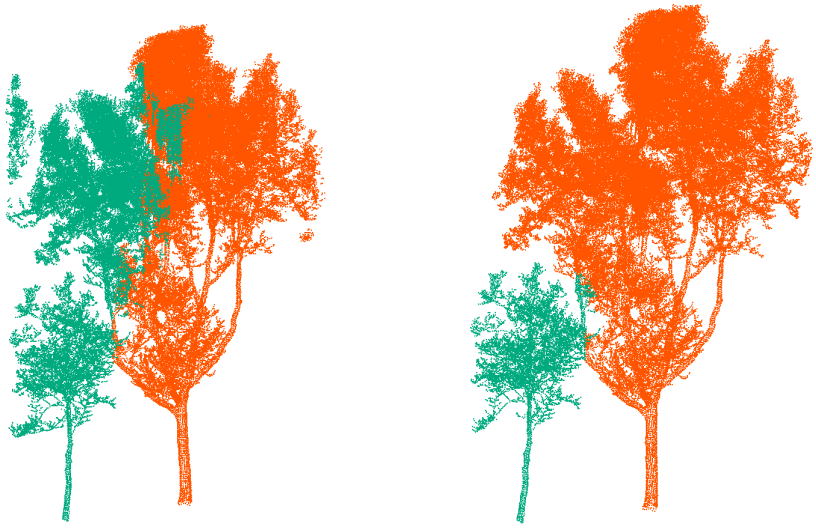
Figure 5.16: A top view of the individual tree delineation results of the two methods from TLS point cloud. (a) A top view of the separation results of Wu's method. (b) A top view of the separation results of the proposed VoxTree method.

In the figure, (*a*) is a top view of the separation results from Wu's method and (*b*) is the results of the proposed VoxTree method. It can be seen that the method proposed in (Wu et al., 2013) works very well and most of the points were correctly assigned to a particular tree. The VoxTree method is good at separating connected parts of trees and generates therefore better results. To quantify the delineation accuracy of the two methods, the manually separated ground truth of the trees was used as reference and the separation Cohen's Kappa (Cohen, 1960) of the two methods are computed. For this purpose each of the 11 trees was considered as a classification class. Using the reference data, a 11×11 confusion matrix was determined shown in Table 5.2. From this confusion matrix Cohen's κ was determined for both Wu's method and VoxTree. The κ of Wu's method is 89% and the resulting κ value of VoxTree is 94% for this specific patch of trees. Table 5.2 gives the tree separation details of the proposed VoxTree method.

Figure 5.17 illustrates a scenario where the VoxTree method outperforms Wu's method (Wu et al., 2013). The Figure displays two connected trees of different height. Figure 5.17 (*a*) shows the results from Wu's method. A big proportion of points from the bigger tree were assigned to the smaller tree. However as shown in Figure 5.17 (*b*), even though points from a branch of the bigger tree were wrongly assigned to the smaller tree, the VoxTree method shows big improvement with respect to Wu's method. As can be seen

Individual tree delineation		Trees (manually separated ground truth)											Row total (Num. of Pts.)
		1	2	3	4	5	6	7	8	9	10	11	
Trees (separation results of VoxTree)	1	84265	7651	6618	153								98687
	2		43318	743		90							44151
	3			52683	2163		26						54872
	4	1388		108	59105			554	922				62077
	5			9		38985	269						39263
	6					101	49730	1243					51074
	7				390		240	54208	47			283	55168
	8							1197	60048			5	61250
	9					3044	105			86840			89989
	10						2697	1851		1684	**11739**	508	18479
	11							1605	699		115	**120593**	123012
Column total (Num. of Pts.)		85653	50969	60161	61811	42220	53067	60658	61716	88524	11854	121389	698022

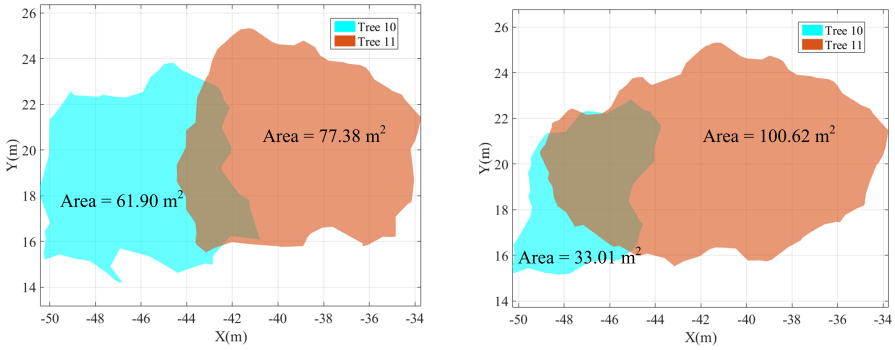
Table 5.2: Details of the tree separation results of the VoxTree method.



(a) (b)

Figure 5.17: Tree separation details of the existing method and the proposed VoxTree method from TLS point cloud. (a) Separation results of Wu's method. (b) Separation results of the proposed VoxTree method.

in Table 5.2, the VoxTree method wrongly assigned 115 points from tree 10 to tree 11, and 508 point s from tree 11 to tree 10. For Wu's method these numbers are 196 and 86236 respectively. The subsequent benefit is significant when estimating the canopy area based on the individual tree delineation results of the two methods. Figure 5.18 shows the estimation results of the canopy projected area on ground by using the resulted points of Tree 10 and Tree 11 in Figure 5.15. The areas of the two trees from the two resulted points were approximated by alpha shape with same radius of 0.8 m. Figure 5.18a and Figure 5.18b are the estimated canopy area from the two results respectively. As can be seen the estimated area of Tree 10 are 61.90 and 33.01 m^2 and Tree 11 are 77.38 m^2 from the two results respectively.



(a) (b)

Figure 5.18: Canopy area estimation from the resulted points of Tree 10 and Tree 11. (a) Canopy area estimated by the resulted points from Wu's method. (b) Canopy area estimated by using the resulted points from the proposed method.

5.5. Conclusions and Recommendations

This chapter answered the following research question:

How to individualize roadside trees from MLS point clouds?

In this chapter a voxel based scalable individual tree delineation method is proposed. The method first clusters cells based on connectivity, then a novel cell adjacency analysis approach is introduced to separate connected trees. It is scalable in the sense that it can operate at different levels of detail. A series of tests have been conducted to verify the flexibility and reliability of the proposed VoxTree method in different scenarios.

First a scenario is presented consisting of three point clouds from different sensors, which are ALS, MLS and TLS respectively, sampling the same three adjoining trees. This test confirms that the VoxTree algorithm is capable of dealing with point clouds from different platforms and different point densities. To validate the performance of the algorithm for different voxel scales, the same MLS point cloud is processed at different scales. Four different scales are used and the three individual trees were delineated to different levels of detail. Consecutively two challenging scenarios were tested to verify the capability of the proposed method dealing with trees that have no trunks points and trees on steep terrain. Then a patch of 63 trees connecting in different directions was analyzed and 66 trees were found. By examining erroneous cases manually, the errors were mainly caused by local under-sampling of the trunks in the point clouds. Finally another 11 trees were scanned using a TLS scanner and the proposed VoxTree algorithm was compared to an existing method and manually separated ground truth. The results shows that VoxTree outperformed the existing method and improved the

separation accuracy κ from 89% to 94%.

Although the presented tests have illustrated the advantages and reliability of the proposed VoxTree algorithm, still some weaknesses were noticed as well. First, the input of this algorithm is tree points only, thus the tree points need to be segmented from the original point clouds. Second, this algorithm also needs preset parameters, which are voxel sizes and minimum tree canopy diameter. The maximum tree size was set automatically as 5 times the minimum tree canopy diameter. The voxel sizes in the three Cartesian directions affect the level of detail of individual tree delineation results and computation time. The minimum tree canopy diameter will influence the seed merging step. The method works better for trees that have similar canopy diameters. Over- or under-individualization may occur when tree sizes are strongly varying. In addition, the application of the proposed algorithm in forest scenarios and in situations where trees are also connected with understorey vegetation is not yet validated.

A recommendation is to tile input tree points according to approximate tree size. This is expected to generate more reliable delineation results. Also, the proposed method is based on voxels, which correspond to uniform 3D grids. Since most of the space is not occupied by tree points, uniform 3D re-sampling to voxels introduces memory redundancy in the processing. Therefore an additional recommendation is to organize the cells using octrees rather than voxels as a next step work.

6

Automatic Roadside Furniture Recognition

Roadside furniture, such as lamp poles, traffic signs and traffic lights, are key elements of the road environment. Inventory and monitoring of those objects is crucial for ensuring the normal function of roads. MLS systems sample the road environment efficiently, still, until now, it was hardly possible to automatically identify roadside objects from the shape information available in these point clouds. This chapter therefore presents an automatic algorithm for roadside object identification. The proposed methods notably exploits a newly introduced 3D shape descriptor, SigVox, that has been especially designed for this purpose, but it is expected to have many other applications. This chapter first introduces the problem properly before discussing related work. Next, the proposed method is explained in detail. Finally, it is evaluated on two times 4km of MLS data sampling the TU Delft campus.

6.1. Introduction

Urban roads are of crucial importance in modern society by reducing the distance between people and services, and produce economic and social benefits (Vanier, 2006). The condition of the urban road furniture, i.e. street lamps, traffic lights, traffic signs, bus station signs and billboards, needs to be inspected and documented regularly to avoid potential risks caused by wear, vandalism or accidents (Halfawy, 2008). Furthermore, high precision urban maps are extensively demanded in various fields, such as smart cities (Nebiker et al., 2010; Batty et al., 2012), autonomous driving (Li et al., 2004; Schreiber et al., 2013) and intelligent transportation systems (Bishop, 2000; Agamennoni et al., 2011; Ivan et al., 2015), etc. Efficient and frequent updating of the urban road inventory is essential to ensure the overall technical and social function of a city. Currently, safety inspections on roadside furniture are conducted by manual in situ examination or semi-automatic interpretation of collected imagery and video data (Pu et al., 2011). These methods are valid and practical in identifying defect roadside furniture. However, the methods are labor intensive and expensive. Therefore, the methods are not optimal for striking a balance between maintaining safety and reducing expenses.

In the last decades new techniques based on photogrammetry and remote sensing have been developed for obtaining accurate 3D urban measurements (Haala and Brenner, 1999; Ellum and El-Sheimy, 2002; Frueh and Zakhor, 2003; Over et al., 2010; Mc Elhinney et al., 2010; Puente et al., 2013a). Among the developed techniques, Mobile Laser Scanning (MLS) systems, which combine Light Detection And Ranging (LiDAR), Global Navigation Satellite Systems (GNSS) and an Inertial Navigation System (INS), are able to obtain dense and highly accurate point measurements (Vosselman and Maas, 2010). A MLS system continuously samples the road environment and the geometry of objects is captured in form of point clouds with versatile information, i.e. precise coordinates, intensity of laser pulse and RGB color. In recent years, the collected point cloud data have been used in various applications, such as 3D tree detection and modeling (Rutzinger et al., 2010; Zhong et al., 2013; Wu et al., 2013; Lindenbergh et al., 2015), road surface extraction (Jaakkola et al., 2008; Mc Elhinney et al., 2010; Pu et al., 2011; Wang et al., 2013; Guan et al., 2014), curbstone identification (Zhou and Vosselman, 2012; Yang et al., 2012a, 2013; Kumar et al., 2014), road corridor objects classification (Pu and Zhan, 2009; Puttonen et al., 2011), change detection (Qin and Gruen, 2014)

and mountain road monitoring (Wang et al., 2014; Díaz-Vilarino et al., 2016). Particularly, the obtained high density point clouds enable the detection and identification of objects along road corridors. Pole-like objects, such as tree trunks, lamp poles and traffic light poles can be identified and extracted (Brenner, 2009; Golovinskiy et al., 2009; Lehtomäki et al., 2010b; Pu et al., 2011; Yang et al., 2012b; Cabo et al., 2014; Yang et al., 2015). However, still lacking are methods for recognition, identification and grouping specific types of roadside object from MLS point clouds.

In this chapter, an automatic urban roadside object recognition method is presented for MLS point cloud data. The proposed method starts with raw point cloud data obtained by a MLS system. First, the point cloud is tiled along the road direction following the trajectory of the MLS system. Next, the point cloud tiles are divided in ground and non-ground points. The non-ground points are organized in an octree data structure and connected voxels are clustered. Consecutively, a newly proposed 3D SigVox shape descriptor of the objects of interest, such as different types of street lamps and traffic signs, is constructed. Finally, objects in the clustered point clouds are recognized by SigVox descriptor enabled template matching.

The contribution of this work to the state of the art is as follows: (i) It introduces a novel 3D multi-scale shape descriptor, that is easy to compute and powerful for shape detection; (ii) It gives a work-flow to use this shape descriptor to identify different types of lamp poles and traffic signs; and (iii) In doing so, it shows how to efficiently handle large MLS point clouds, e.g. by using a suitable tiling strategy. What we consider as a notable innovation is that this method, for the first time, use shape descriptors of complete objects to match repetitive objects in large point clouds.

The remainder of the article is organized as follows. Section 6.2 presents related work in object recognition, followed by a detailed description of the proposed method in Section 6.3. In Section 6.4 the proposed method is demonstrated and validated on a MLS point cloud sampling 4km of road environment. Finally, conclusions are given in Section 6.5.2.

6.2. Related work

MLS systems efficiently sample the surface of objects along a road and record the measurements as dense and accurate point clouds (Puente et al., 2013a; Barber et al., 2008; Haala et al., 2008; Cahalane et al., 2010). The acquired measurements, which typically consist of 3D coordinates, intensity of laser pulse and RGB color information, enable the recognition of roadside objects. A variety of methods has been presented on this topic. The available methods for object recognition can roughly be classified into three categories: (i) model fitting based (Pu et al., 2011; Rutzinger et al., 2010; Lehtomäki et al., 2010b; Brenner, 2009; Cabo et al., 2014; Xiao et al., 2016); (ii) semantic based (Fan et al., 2014; Yang et al., 2015; Babahajiani et al., 2015a); and (iii) shape based (Golovinskiy et al., 2009; El-Halawany and Lichti, 2011; Velizhev et al., 2012; Bremer et al., 2013; Yang and Dong, 2013; Li, 2013; Rodríguez-Cuenca et al., 2015). This section first reviews the related work corresponding to the aforementioned methods on object recognition in Section 6.2.1, Section 6.2.2 and Section 6.2.3 respectively. Finally, a comparison of some related methods is given in a table.

6.2.1. Model fitting methods

A model fitting based object recognition method in general starts with segmenting and clustering the point cloud, followed by fitting the point segments to known geometric models, such as cylinders and planes. Brenner developed an algorithm for pole extraction from MLS scanned point clouds. The method first assumes that the basic characteristic of a pole is that it is upright. There is a kernel region where laser scanned points are present and an outside region where no points are present (Brenner, 2009). Point segments are analyzed in cylindrical stacks and when a certain minimum number of stacks is detected, the segment is considered as a pole. The final step is to estimate the exact position of the pole.

In 2010, Lehtomäki et al. presented an algorithm to detect pole-like objects in a road environment using MLS point clouds. The algorithm first segment scan lines and then remaining point groups are clustered. Consecutively adjacent clusters that are closely connected or overlapping in horizontal profiles are merged. Based on these merged point clusters, cylinder fitting is performed to detect poles along the road direction. The method was able to find 77% of the poles if compared to a manual data analysis with a correctness of 81% (Lehtomäki et al., 2010b).

In 2011, Pu et al. presented a method to recognize basic structures from MLS point clouds for road inventory. The method first roughly classified tiled raw point clouds into ground and non-ground objects. Then geometric attributes, i.e. size, position, orientation, RGB color and material of objects' surface were characterized and organized per segment. Consecutively objects with planar features were approximated by planar models, such as rectangles, circles and triangles. Pole-like objects were sliced vertically and for each slice a 2D enclosing rectangle was derived. Next the differences of those rectangles, i.e. position and size differences, between neighboring slices were checked, and if they were within a defined threshold then similar slices were accumulated. Finally, if the number exceeded a maximum length, a point segment was considered a pole-like object (Pu et al., 2011). This method was capable of recognizing building walls and pole-like objects such as lamp poles and tree trunks. The final results showed that 86% and 64% of poles and trees respectively were correctly recognized.

In 2014, Cabo et al. introduced an algorithm that automatically detects pole-like street furniture objects from MLS point clouds. Rather than directly considering each point as the aforementioned methods did, this algorithm first simplified the point cloud into voxels. Then each 2D vertical layer of the constructed 3D regular voxels was analyzed and potential elements were selected by an isolation criterion. The isolation criterion was evaluated based on fitting two 2D rings of different radii. If a candidate voxel cluster is enclosed between the inner and outer ring, then it is considered as a potential pole object. The algorithm was tested on point clouds of four test sites and was able to recognize all the target pole-like objects except of severely occluded ones (Cabo et al., 2014).

In 2016, Xiao et al. did not consider pole-like objects but introduced a method to detect street-side vehicles with a deformable vehicle model using MLS point clouds (Xiao et al., 2016). This method first classified raw point clouds into ground, buildings and street objects. Then geometric features were extracted from obtained street objects. Next, these features were fit to an explicit model. The vehicle recognition accuracy

could reach up to 95%.

6.2.2. Semantic methods

Semantic methods for object recognition usually define a set of rules based on prior knowledge of the objects. Then based on these rules, objects are extracted and recognized. In 2014, Fan et al. introduced an algorithm for identifying man-made objects along urban road corridors from MLS point clouds (Fan et al., 2014). The method assumes that, firstly, man-made objects have a regular geometry whereas vegetation has a complex geometry; secondly, different urban man-made objects are characterized by the point extension and the height above the ground level. With the above rules, the method divides a MLS point cloud into three layers with respect to vertical height. In each layer, seeds of man-made object are indicated by a line filter in the foot prints of off-ground objects. Further classification is performed on those seeds by checking in which layers the seed points of an object are found. Finally, points belonging to respective objects are retrieved based on the classified seed points. The capability of extracting man-made objects was found to have a detection rate of 83%.

In 2014, Teo et al. proposed a similar method as Cabo et al. in (Cabo et al., 2014) to detect pole-like object from MLS point clouds (Teo and Chiu, 2015). After removal of building facade points, the point cloud was re-sampled by voxels which were used to obtain a coarse segmentation. Next, fine-segmentation is conducted based on point to point distances which enables the separation of overlapping objects. Based on a series of predefined rules, pole-like objects were detected in a hierarchical way. The method was tested on two point clouds and the results showed that the correctness of the pole-like detection were 97.8% and 96.3% respectively for those test data sets. However, the method cannot classify different types of pole-like objects.

In 2015, Babahajiani et al. presented a method to recognize objects in 3D point clouds of urban street environments (Babahajiani et al., 2015a). The method starts with automatically extracting ground points. Building facades are detected using binary range images. Then the remaining points are voxelized and transformed to super voxels. Consecutively, boosted decision trees are employed to train and classify the extracted local 3D features of the voxel cells. The output of the classification is labeled with semantic classes. This method is evaluated on a challenging fixed-position TLS point cloud and a MLS point cloud. The global accuracy and per-class accuracy were about 94% and 87% respectively.

Yang et al. proposed an automatic algorithm for hierarchical extraction of urban objects from MLS point clouds (Yang et al., 2015). The method segments MLS points into ground and non-ground points. Based on the non-ground points, multi-scale supervoxels are generated. For each supervoxel, its geometric nature is determined by PCA. Then the multi-scale supervoxels are segmented with regard to their geometric type. In addition, the saliency of the segments is also calculated. Furthermore, seven semantic rules are defined corresponding to seven types of object, i.e. building, utility poles, traffic signs, trees, streetlamps, enclosures and cars. The method was validated on two MLS point clouds and the results demonstrate that the object extraction and classification accuracy of the proposed method was better than 91%.

6.2.3. Shape based methods

Shape based methods consider the explicit or implicit shapes of point clusters or segments. Then shape features are calculated to classify and identify objects from MLS point clouds. In 2009, Golovinskiy et al. presented a shape-based recognition method for analyzing 3D MLS point clouds of urban environments (Golovinskiy et al., 2009). The method first determines the location of each potential object, then those objects are segmented from the original point cloud. Features are extracted from these object segments. Classification of those segments is performed based on the extracted features. The evaluation demonstrated that this method is able to recognize 65% of the objects.

In 2011, El-Halawany et al. proposed a pipeline for roadside pole detection from MLS point clouds (El-Halawany and Lichti, 2011). The algorithm first calculates the eigenvalues of the covariance matrix of a local neighborhood using a KD tree. Then eigenvalue-based segmentation is conducted and linear objects are extracted by region growing. The final recognition results are evaluated by cylinder fitting and an eigen-radius relation. Velizhev et al. proposed an implicit shape model based automatic method for object localization and recognition in 3D outdoor scenes from MLS point clouds (Velizhev et al., 2012). The method consists of two steps. First a list of hypotheses on objects is determined by connected component extraction. Then objects are recognized using local descriptors and a voting-based localization method. The method was validated on a MLS point cloud and the recognition precision was 68% and 72% for cars and light poles respectively.

In 2013, Bremer et al. presented a method based on eigenvalues and graphs to extract objects from MLS point clouds (Bremer et al., 2013). First the method calculates a 3×3 covariance matrix for each point from a local neighborhood and eigenvalues and eigenvectors are derived. Then those eigenvalues are characterized and classified. By connected component segmentation and clustering, ground and building facades are separated. Finally pole objects including trees are separated using a Dijkstra region growing approach.

In 2015, Yu et al. proposed a street light pole extraction algorithm for MLS point clouds based on pairwise 3D shape context (Yu et al., 2015). The method first detects road curbstones based on a series of profiles perpendicular to the road direction. Curblines are extracted to divide a point cloud into road and non-road points. Ground points are further segmented from non-road points using a voxel based height filter. Next, non-ground points are clustered as separated object segments. Finally a point based 3D shape context, i.e. the fast point feature histogram (FPFH) proposed in (Rusu et al., 2009), was used to match the objects of interest. The method was tested on a MLS point cloud and street poles were robustly extracted with a completeness over 99% and a correctness of 97%.

Rodríguez-Cuenca et al. presented an automatic pole-like object detection and classification method from MLS and TLS point clouds based on an anomaly detection algorithm. The method first extracts ground points and then based on an anomaly detection algorithm, vertical objects are detected as point clusters. Then the detected vertical objects are classified as either man-made poles or trees. The testing results demonstrated that the detection rate was 96% and the classification rate was 95%.

Method	Shape descriptor	Identified objects	Used input	Voxel
Yang et al.	no	buildings,traffic signs, trees, street lamps, cars,enclosures	(x,y,z,Intensity,r,g,b)	no
Yu et al.	point based	bus stations, light poles traffic poles	(x,y,z)	no
Cabo et al.	no	poles	(x,y,z)	yes
Teo et al.	no	pole-like objects	(x,y,z)	yes
SigVox	object based	different lamp poles and road signs	(x,y,z)	yes

Table 6.1: A comparision of four existing methods with the proposed method.

Table 6.1 summarizes some key methods and compares their approach to the proposed SigVox method. So far, only (Yu et al., 2015) also matches different instances of the same type of furniture. Our proposed method has the same goal, but we propose the use a shape descriptor that operates at object scale.

6.3. Methodology

As illustrated in Figure 6.1, the proposed algorithm for automatic urban roadside object recognition consists of four consecutive steps:

1. Pre-processing. First the input raw point cloud is tiled with regard to the scanning trajectory. Next, the tiled point cloud is divided in non-ground and ground points.

2. Voxelization and building SigVox descriptors. Non-ground points are voxelized using an octree and connected voxels are clustered. Examples of objects of interest are manually selected for training and SigVox descriptors are constructed to form a template list.

3. Similarity Matching. Each of the clusters is automatically examined if it is a candidate for the selected objects of interests. If yes, then its SigVox descriptor is built and its similarity to the different training objects is computed. The cluster is then assigned to the best matched training object.

4. Validation. The recognition results are analyzed with regard to ground truth data and their accuracy is determined.

6.3.1. Pre-processing

The pre-processing in this work consists of two parts, i.e. tiling of the raw point cloud, and separation of ground and non-ground points, as indicated in Figure 6.1. One scan of a MLS point cloud data set usually is too large to process on a normal desktop computer. Thus the raw point cloud is divided into tiles of suitable size. Furthermore, the focus of this work is on the non-ground objects rather than the ground points. Therefore, the tiles of the point cloud are further segmented into ground and non-ground points.

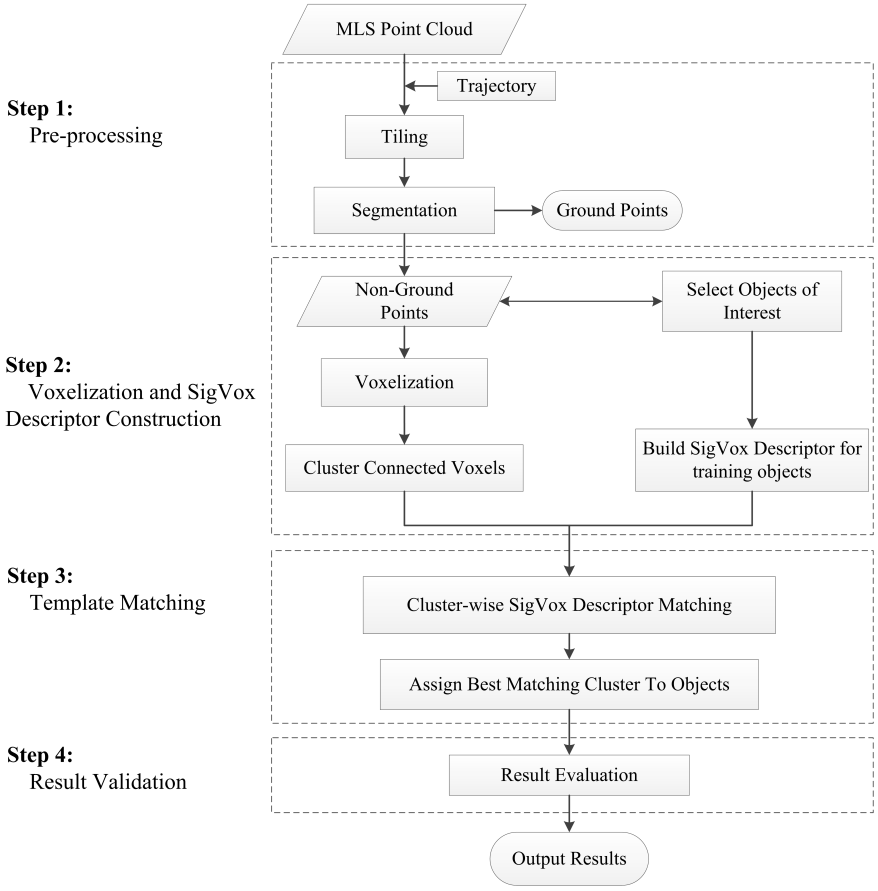


Figure 6.1: Overall methodology of the proposed algorithm. The method starts with a MLS point cloud and results in a list of roadside objects.

In this work, the scanning trajectory of the MLS system is used to partition the original MLS point cloud. Trajectory data is obtained by the Position and Orientation System (POS) of the MLS system, which consists of a series of 3D positions recorded at high frequency. The original MLS point cloud is tiled along the scanning trajectory.

Figure 6.2 shows an example tiling of a raw point cloud acquired by a MLS system. The red line is a segment of the so-called Smoothed Best Estimation of Trajectory (SBET) of the MLS system and the purple arrow indicates the scanning direction. In this example, three tiles are generated and overlapping areas are indicated. During tiling, the 3D trajectory is projected on the horizontal plane. For each tile, the length along trajectory, i.e. the distance between *Starting point* and *Endpoint* along SBET in Figure 6.2, and the width across trajectory are flexible. Figure 6.3 is a magnification of the 2D polygon in Figure 6.3. Points P_1, P_2 and P_3 are points of SBET and d is the width of the polygon in the across trajectory direction. The 2D boundary of each tile

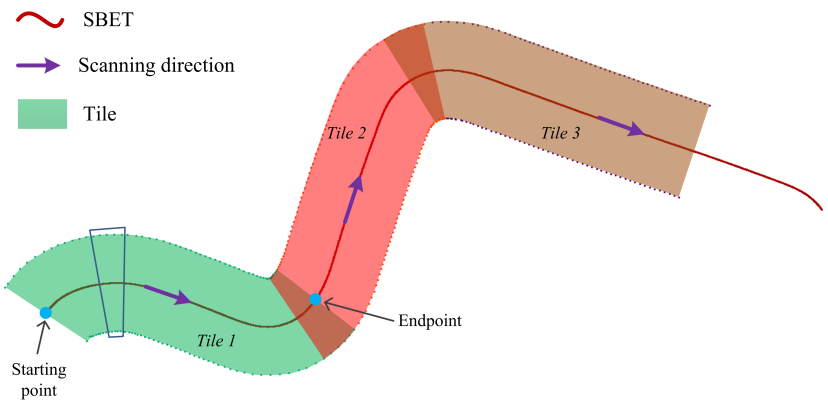


Figure 6.2: Tiling of original MLS point cloud data along the road corridor direction. Different colors indicate different tiles.

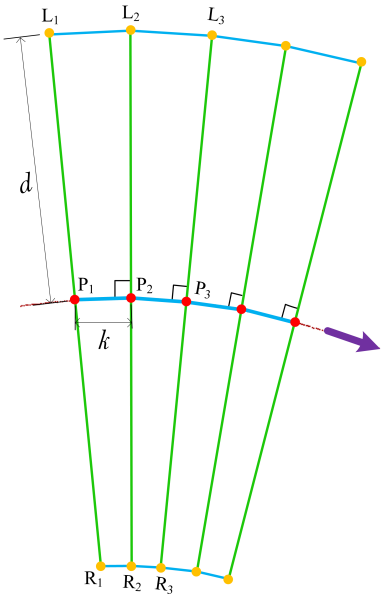


Figure 6.3: Geometry of a tile polygon along the scanning trajectory. The P_i denote given trajectory points. The L_i and R_i are computed tile boundary points and are defined on the basis of a width parameter d.

is obtained by accumulating polygons of the defined resolution k. In the first polygon $L_1R_1R_2L_2$, edge L_2R_2 is perpendicular to line segment P_1P_2. Consecutively, edge L_3R_3 of the second polygon $L_2R_2R_3L_3$ is obtained. All those concatenated 2D polygons form the boundary of *Tile 1*. Finally, all points that project within the boundary of the tile are output as belonging to that tile.

The next step of pre-processing is to identify the non-ground points in the tiled point clouds. In this study, the algorithm proposed by Pfeifer (Pfeifer, 2001) is used. The non-ground points will be forwarded to the next step. Meanwhile, points from objects of interest, such as street lamp poles and traffic signs, are selected and stored separately for training purposes.

6.3.2. Voxelization

The voxelization is performed on the non-ground points only. In this step, non-ground points are re-sampled to voxels organized by an octree data structure. Next connected voxels are clustered.

The octree data structure is extended to 3D from the 2D quadtree as introduced by Klinger (Allen, 1971). The octree data structure connects each branch of the tree node with a 3D Cartesian node, which is also defined as a voxel. This tree node can be subdivided recursively into 8 branches, i.e. octants. Figure 3.7 demonstrates an octree based Cartesian spatial subdivision and its hierarchical data structure. This data structure enables efficient neighborhood searching by building up a series of look up tables. In this work, the neighborhood searching strategy proposed by Payeur is implemented (Payeur, 2006).

6.3.3. Clustering and Selecting Candidate Clusters

For the voxelization of the non-ground points, the bounding box of a tile of non-ground points is considered as the root node of the octree structure. The root node is subdivided recursively until the preset subdivision criteria are met. Consecutively, connected voxels are clustered based on a 3D seed filling algorithm proposed by Yu et al. (Yu et al., 2010). Then the points inside each voxel cluster are stored as point clusters. The subsequent point clusters are denoted by C^i, $i = 1,2,...,k$. Here k is the number of obtained point clusters. It is expected that many of these point clusters correspond to a roadside object like a street pole or traffic sign.

Next the 3D bounding box, i.e. $B(C^i)$, of each point cluster C^i, is obtained and compared with the 3D bounding boxes of the selected training objects T^j, $j = 1,2,...,h$. Here h is the number of training objects of interest. If the relative difference between the 3D bounding box of the *j-th* point cluster and the 3D bounding box of a training object is small, the point cluster is considered as a candidate of the *j-th* object. The relative difference $D_{i,j}$ between two 3D bounding boxes is computed by Equation 6.1.

$$\begin{aligned} D_{i,j} &= \sqrt{\frac{\left\| B(C^i) - B(T^j) \right\|}{\left\| B(T^j) \right\|}} \quad (i = 1,2,...,k; j = 1,2,...,h) \\ &= \sqrt{\frac{[B(C^i)_x - B(T^j)_x]^2 + [B(C^i)_y - B(T^j)_y]^2 + [B(C^i)_z - B(T^j)_z]^2}{[B(T^j)_x]^2 + [B(T^j)_y]^2 + [B(T^j)_z]^2}} \end{aligned} \tag{6.1}$$

Here, $B(C^i)$ and $B(T^j)$ are the 3D bounding boxes of a candidate cluster and a training object respectively. Suppose there are k candidate clusters and h training clusters. B_x, B_y and B_z are the sizes of the bounding box in the three coordinate directions. When

$D_{i,j}$ is smaller than a preset threshold, then a point cluster is regarded as a candidate of the *j-th* training object $C(T^j)$. Due to possible variations in the orientation of the roadside object, the orientation of bounding box will vary as well. Thus, the threshold used here for candidate object selection needs to be big enough to avoid omission.

6.3.4. SigVox descriptor construction

In this section, the approach for determining the dimensionality of a voxel is given. Consecutively, the concept of the 3D SigVox descriptor and the methodology to construct SigVox descriptors is presented.

Dimensionality Analysis

Non-ground objects were clustered as voxel clusters in Section 6.3.2. Consecutively the dimensionality of each voxel in these clusters is determined by PCA. The dimension of a voxel is determined as follows: Suppose $p_i = (x_i \quad y_i \quad z_i)^T$ are the coordinates of a point p_i inside the voxel cell, then the center of gravity $\overline{p}$ of all the points p_i inside the voxel is determined by Equation 6.2.

$$\overline{p} = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{6.2}$$

Here n is the number of points inside the voxel. The 3D structure tensor M of the points is defined by Equation 6.3.

$$M = \frac{1}{n} Q^T Q \tag{6.3}$$

Here $Q = (p_1 - \overline{p}, p_2 - \overline{p}, ..., p_n - \overline{p})^T$. M is a symmetric matrix and can be decomposed as $M = RIR^T$. Here R is a rotation matrix and I a diagonal positive definite matrix. The elements of I are the eigenvalues of matrix M. The three eigenvalues are positive, are denoted by $\lambda_1, \lambda_2, \lambda_3$, and are sorted such that $\lambda_1 > \lambda_2 > \lambda_3$. The corresponding eigenvectors are v_1, v_2, v_3 respectively.

In this chapter, voxel cells are categorized into three types: linear, planar and scatter. The three cases are defined as follows: (i) If for the eigenvalues of a voxel it holds that $\lambda_1 >> \lambda_2$, then this voxel is defined as a linear, or $1D$ voxel cell. For a linear voxel cell, eigenvector v_1, which corresponds to eigenvalue λ_1, is the significant eigenvector and indicates the significant direction of the points inside the voxel cell. (ii) If $\lambda_2 >> \lambda_3$, then the voxel cell is defined as a planar, or $2D$ cell. In this case eigenvector v_3, the normal vector of the plane, is the significant eigenvector. (iii) If $\lambda_1 \approx \lambda_2 \approx \lambda_3$, then this cell is defined as a scatter cell, or $3D$ cell. A scatter cell does not have a significant direction and will not be considered. Here >> means *much larger* and is implemented by a preset threshold. In this chapter, the linearity is examined first and then planarity. The thresholds for linearity and planarity are denoted by T_l and T_p respectively. After this procedure, all voxel cells in a cluster have a geometric flag denoting its dimensionality and if applicable, a significant eigenvalue and eigenvector as its properties.

EGI Descriptor

In this section, the approach to construct the proposed SigVox 3D descriptor for both training objects and candidate voxel clusters is demonstrated. The SigVox descriptor

is inspired by the existing EGI descriptor, proposed by Horn (Horn, 1984). The EGI descriptor is approximated by an icosahedron. Rather than computing local normals from a query point with a radius as in (Horn, 1984), the significant eigenvectors obtained in Section 6.3.4 are used to construct a 3D EGI descriptor (Wang et al., 2016). An approximated sphere, i.e. an icosahedron in this work, is used to assign significant eigenvectors to its surface and is also defined as the *Eigen-Sphere* of a voxel cluster.

The full sphere is approximated by an icosahedron. As illustrated in Figure 6.4, the relative position of the icosahedron with regard to the Cartesian coordinate axes is given by its standard position in this work. In Figure 6.4, point O is the origin of the

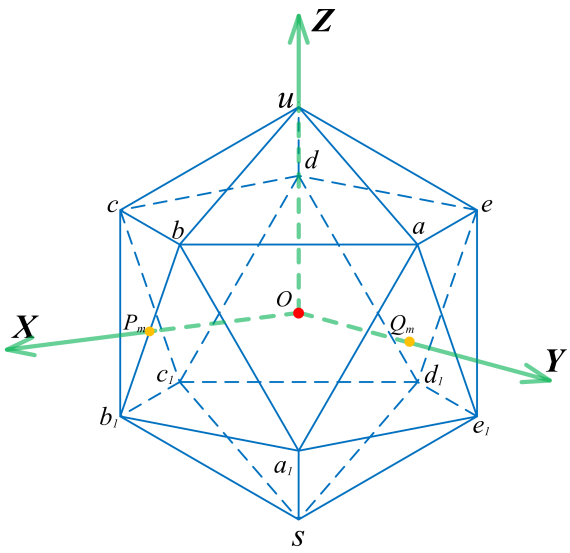


Figure 6.4: A uniform icosahedron used to construct the EGI descriptor. Such icosahedron provides a coarse sphere approximation.

coordinate system and the geometric center of the icosahedron. The X axis intersects one icosahedron edge at point P_m and the Y axis penetrates through one triangle patch at point Q_m, while the Z axis passes through vertex u. This is an uniform icosahedron, as the distances of its 12 vertices to the origin O are equal to 1.

The next procedure is to assign the obtained significant eigenvectors of all voxels in a cluster to the triangles on the boundary of the icosahedron. In Figure 6.5, triangle *uae* is one of the 20 boundary triangles of the icosahedron in Figure 6.4. Vector $\vec{v}$ intersects triangle *uae* at point P. Suppose the three vertices correspond to vectors $\vec{v_n}$, $\vec{v_a}$, $\vec{v_e}$ respectively, then the coefficients k_n, k_a, k_e in the linear combination in Equation 6.4 must be all positive (Preparata and Shamos, 1985). This is used to assign significant eigenvectors to the correct triangle.

$$\vec{v} = k_n \vec{v_u} + k_a \vec{v_a} + k_e \vec{v_e} \tag{6.4}$$

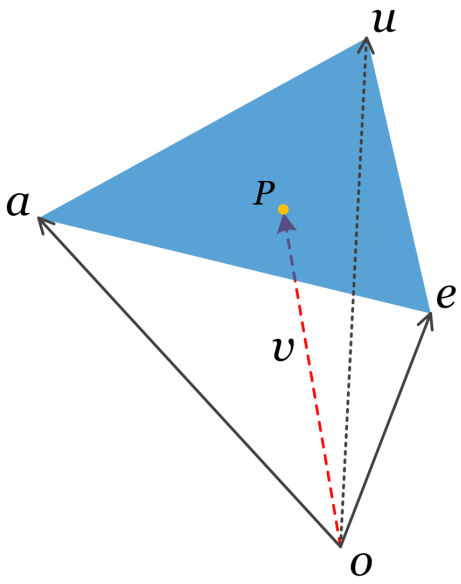


Figure 6.5: Assign a significant eigenvector $\vec{v}$ to a triangle *aeu* on the boundary of the icosahedron.

Indeed, to determine the coefficients in Equation 6.4, the function can be decomposed and rewritten to Equation 6.5.

$$\begin{pmatrix} k_n \\ k_a \\ k_e \end{pmatrix} = \begin{pmatrix} \vec{v_u} & \vec{v_a} & \vec{v_e} \end{pmatrix}^{-1} \cdot \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{6.5}$$

Here $\vec{v_n} = (x_n, y_n, z_n)^T$ are the coordinates of vertex v_n and v_x, v_y, v_z are the coordinates of vector $\vec{v}$.

The 3D EGI descriptor of the candidate cluster is constructed by assigning the significant eigenvectors of all voxels in the cluster to its *Eigen-Sphere* by determining for each eigenvector which triangle of the icosahedron it intersects using Equation 6.5. This means that only the voxel eigenvector corresponding to the biggest eigenvalue is assigned for a linear voxel, while for a planar voxel, only its normal vector is assigned. Note that the eigenvectors obtained by PCA are not direction definite, e.g. vector $\vec{v}$ and its antipodal vector -$\vec{v}$ are both applicable. In this work, for the purpose of symmetrical concern, both a vector and its antipodal vector are assigned to the *Eigen-Sphere*.

For each significant eigenvector assigned to a particular triangle, a weight value W^k is stored, indicating the percentage of points it contains of the cluster it is from. That is, the weight W^k of a significant voxel contributing to the *i-th* triangle, is computed by Equation 6.6.

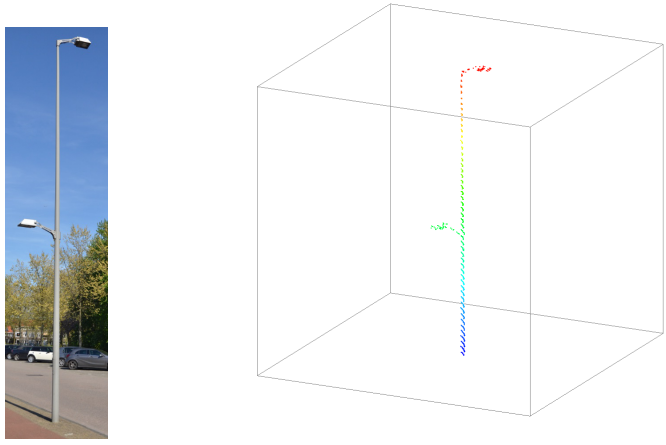
$$W^k = \frac{N_k}{N} \tag{6.6}$$

Here N_k is the number of points in the considered voxel while N is the total number of points in the cluster.

The aim of the weights is to avoid ambiguity in dimensionality determination in Section 6.3.4. Division of the point cluster of an object using an octree will separate different parts of the object in a somewhat arbitrary way, depending on the particular orientation of the object. For example, a voxel that contains only one border of a plane will appear linear. Such voxels will lead to ambiguity in the dimensionality as a whole. The weight will take the number of points in the voxels into consideration. This weight is therefore designed to reduce the sketched ambiguity.

SigVox Descriptor

In this work, a significant eigenvector based shape descriptor using multiple levels of voxel is proposed, which is denoted by SigVox. SigVox is constructed based on the recursive subdivision of each candidate point cluster using the octree. At each level of subdivision, the geometric dimensionality feature of each voxel cell is computed as described in Section 6.3.4 and their significant eigenvectors are obtained.

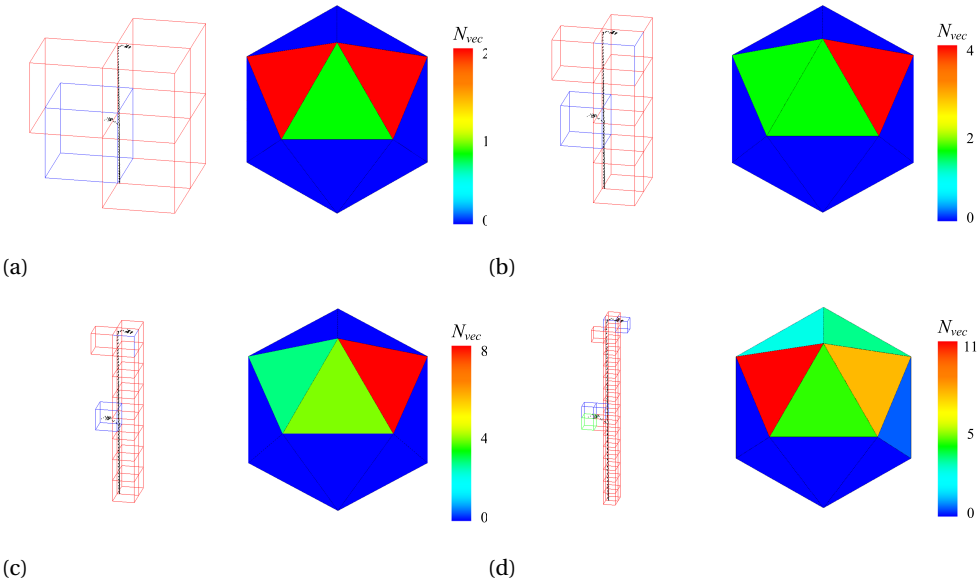


(a) (b)

Figure 6.6: A street lamp pole and its point cloud. (a) A typical street lamp. (b) Point cloud of the lamp pole and its original octree octant.

Figure 6.6a shows a typical type of street lamp in this study. Figure 6.6b is the corresponding point cloud of the pole and its original octree octant, which is also the root node of the octree.

Figure 6.7 demonstrates the recursive subdivision of the street lamp pole by an octree at four levels and the corresponding *Eigen-Spheres*. In each sub-figure, the left figure denotes the subdivision of the point cluster while the right figure is the corresponding EGI descriptor represented by an *Eigen-Sphere*. In the sub-figures, linear, planar and scatter voxels are denoted in red, green and blue respectively. The triangles of the icosahedron at each subdivision level are colored according to the number of



(a) (b)

(c) (d)

Figure 6.7: Recursively subdividing a lamp pole at four levels by an octree and the corresponding *Eigen-Shpere*. (a) Subdivision at level 1. (b) Subdivision at level 2. (c) Subdivision at level 3. (d) Subdivision at level 4. In the different subdivisions, red, green and blue octants indicate linear, planar and scatter voxels respectively.

collected significant eigenvectors. The number of voxels at the four subdivision levels are 6, 10, 18 and 36 respectively. The number of significant voxels is 5, 8, 16 and 33 respectively. For example, in Figure 6.7b the subdivision is at level 2 and there are 10 voxels. The two blue voxels are scatter voxels. Thus there are $10-2=8$ significant voxels and their eigenvectors are assigned to the three red and green triangles of the icosahedron.

For each candidate point cluster, their SigVox is defined as an ordered series of significant eigenvector based EGI at different levels of subdivision, as given by Formula 6.7.

$$SigVox = \{E_1, E_2, ..., E_n\} \tag{6.7}$$

Here, E_1, E_2, E_n are the corresponding EGI descriptors at level 1, 2 and n.

In this work, the similarity between each candidate point segment and its corresponding template segment will be determined by comparing the distance between their SigVox descriptors for a preset number of levels.

6.3.5. Descriptor Matching

This section will first introduce the distance between a pair of SigVox descriptors of two point clusters. The second part describes the transformation method that is used to evaluate the similarity between two EGI descriptors, i.e. the similarity at a fixed scale.

Distance between SigVox descriptors

The distance between the SigVox descriptors of candidate point clusters and template clusters is determined by accumulating the difference between the assigned vectors and their weights for each EGI triangle at each recursive subdivision level. An icosahedron has a total of 60 symmetry transformations (Conway et al., 2008). Since both the significant vector and their antipodal vector are assigned to the icosahedron surface, only 30 symmetries have to be considered in practice.

Suppose P_c and P_t denote a candidate point cluster and a template cluster respectively. Let E_c^l and E_s^t denote their EGI descriptor at level l respectively. Their similarity at level l is defined by Equation 6.8

$$\begin{aligned} S_l^{c,t} &= min\{\hat{S}_l^{c,t}\}_j \, (j = 1,2,...,30) \\ &= min\left\{\sum_{i=1}^{20}(N_{vec,c}^{i,l} * W_c^{i,l} - N_{vec,t}^{i,l} * W_t^{i,l})^2\right\}_j \end{aligned} \tag{6.8}$$

Here, $S_l^{c,t}$ denotes the similarity between point cluster P_c and template cluster P_t at level l. This final similarity is the best match among a total of 30 comparisons. That is, each $\hat{S}_l^{c,t}$ gives the similarity for one among a total of 30 similarity transformations of the icosahedron. $N_{vec,c}^{i,l}$ and $N_{vec,t}^{i,l}$ are the number of eigenvectors that intersect the *i-th* triangle of the EGI from P_c and P_t at level l respectively. $W_c^{i,l}$ and $W_t^{i,l}$ are the weights of the *i-th* triangle from P_c and P_c at level l. Here j denotes the *j-th* symmetry of the icosahedron. The similarity is defined by the minimum distance of the similarity among all 30 icosahedron similarities.

The multiple scale distance between a pair consisting of a point cluster P_c and template cluster P_t is simply the sum of the similarities at all subdivision levels, as denoted by Formula 6.9.

$$S^{c,t} = \sum_{l=1}^{n} S_l^{c,t} \tag{6.9}$$

Here $S^{c,t}$ is the distance between the SigVox descriptors of point cluster P_c and template cluster P_t. The preset maximum subdivision level is denoted by n.

Transformation of *Eigen-Spheres*

While determining the similarity between two *Eigen-Spheres* descriptors of the same level, i.e. the $\hat{S}_l^{c,t}$ in Formula 6.8 in section 6.3.5, the *Eigen-Sphere* of a candidate is transformed 30 times corresponding to the 30 symmetries of the icosahedron up to antipodal identification. The transformation is performed as described in Algorithm 2.

In Algorithm 2, the input is a pair of *Eigen-Spheres*, i.e. E_l^c and E_l^t respectively. In this step, the *Eigen-Sphere* of template cluster, denoted by E_l^t, is kept stationary and only E_l^c, which is the EGI of the candidate point cluster is transformed. The algorithm first puts the two *Eigen-Spheres* in standard position as given in Figure 6.4. Note that *Vertex u* is at the positive direction of Z axis when the *Eigen-Sphere* is in standard position. Then the E_l^c is rotated consecutively for five times around Z axis by $\frac{2\pi}{5}$ rad to compute the first 5 $\hat{S}_l^{c,t}$. The direction of rotation is performed in a right-handed

Algorithm 2 Algorithm for relatively transform the two *Eigen-Spheres* to determine similarity at subdivision level l.

Input: A pair of *Eigen-Spheres*, i.e. E_l^c and E_l^t.
Output: Similarity value at level l, i.e. $\hat{S}_l^{c,t}$.

```
function COMPUTESIMILARITY
    Place E_l^c and E_l^t in the standard position;
    for m = 0 → 5 do
        for n = 0 → 4 do
            Determine the (m*5+n)-th Ŝ_l^{c,t}
            Rotate E_l^c about Z axis with 2π/5 rad;
        end for
        m = m + 1;
        Determine spherical coordinates of the m-th vertex of E_l^c, (1,θ_m,φ_m);
        Rotate the E_l^c about axis Z with 90 − θ_m degree;
        Rotate the E_l^c about axis X with φ degree;
    end for
    Return the minimum one among the obtained 30 Ŝ_l^{c,t};
end function
```

manner as demonstrated in Figure 6.8. This five values correspond to the five symmetries when *Vertex u* is at the north pole. The next step is to compute the five similarity values when the next vertex is at north pole position. To transform an arbitrary vertex to the north pole, for example *Vertex b*. First the spherical coordinates are computed, i.e. $(1,\theta_b,\phi_b)$, as shown in Figure 6.8. Next the E_l^c is first rotated around axis Z by $\theta_b^{'} = 90-\theta_b$ degree and then rotated around X axis by ϕ_b degree. Subsequently *Vertex b* is transformed to the positive Z axis. When the next vertex is transformed to the positive Z axis, the 5 similarity values are computed. The algorithm runs until all the 30 $\hat{S}_l^{c,t}$ corresponding to the 30 symmetries are all determined. Then the minimum one is obtained and returned as the similarity value of the pair of EGI descriptors and returned.

Many objects have a dominant geometric dimensionality. For example, poles are dominantly linear. For such objects, a local coordinate frame could be acquired by PCA, by aligning the object along the first eigenvector. In this way, the number of symmetries to be considered in the similarity determination can be significantly reduced. Note that in case that object has no dominant dimensionality, this possible refinement is not applied.

Object Recognition

This section describes the strategy of assigning the obtained candidate point clusters $\{C^i\}$, $i = 1,2,...,k$ to a specific kind of target object, which is represented by $\{T^j\}$, $j = 1,2,...,h$.

For a specific object of interest, i.e. $\{T^j\}$, its candidate point clusters are first obtained as described in Section 6.3.2. Then the obtained candidate point clusters, i.e. $\{C^i\}$, are subdivided by an octree into several levels. The SigVox descriptors corre-

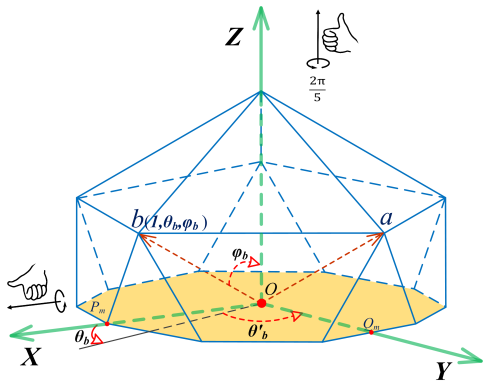


Figure 6.8: Relative transformation of the E_l^c with regard to E_l^t. This figure indicates how a candidate cluster is rotated to match a training object.

sponding to those levels is constructed. Next the similarity between the MS-EGI descriptors of an object of interest and candidate point clusters are determined.

To determine whether a candidate point cluster should be assigned to a specific object of interest, a preset similarity threshold is used. If the determined similarity between two SigVox descriptors, which is denoted be $S^{c,t}$ in Equation 6.9, is below the threshold and this similarity is minimal among different training objects, then the point cluster is assigned to the object of interest. Finally all those point clusters are exported separately.

6.3.6. Evaluation of the similarity

The quality of the object recognition method proposed in this work is evaluated in two ways. First, a similarity score will be determined by the similarity matching of each pair of SigVox descriptors. The similarity score denotes how confident a recognition is. The second way to evaluate the quality of object recognition is in-situ inspection of the results. Finally the results are summarized in a confusion matrix where identification results are compared to ground truth.

The similarity score is computed after the similarity of the SigVox descriptors between all the candidate point clusters and a training cluster are determined. In this procedure, a confidence value will be computed that expresses the quality of the object recognition.

Suppose there are n levels of subdivision for a pair of SigVox descriptors. Then there are n preset thresholds. If among all those n levels of similarity, i.e. n pairs of *Eigen-Spheres*, there are m pairs within the threshold, then this candidate point cluster will have a similarity score $F = \frac{m}{n}$. For example, suppose a point cluster is subdivided by an octree into 4 levels, then its SigVox descriptor consists of four EGI descriptors, one for each scale. When comparing the SigVox descriptors of the point cluster and its template point cluster, if 3 of them are within the thresholds, then its matching score is

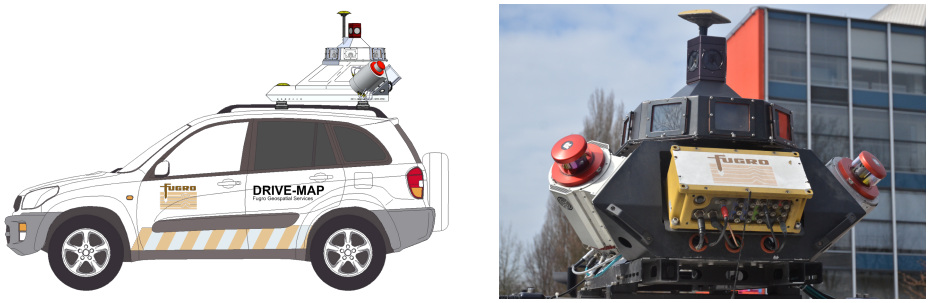
0.75.

6.4. Results and Evaluation

In this section, the proposed method is tested and validated on two MLS point clouds sampling a stretch of 4 km of urban road. First, a brief introduction of the used MLS system and a description of the used point cloud data is given. Then, the results on one of the test runs from each processing step, including pre-processing, voxelization and clustering, and object recognition, are presented and discussed. Next, the results from processing the second test run are presented. This second run samples the same road environment, but the data was acquired in opposite driving direction. This second run data was processed using the same settings. The recognition results from the two point clouds were compared and analyzed. Finally, the recognition accuracy of the proposed method was evaluated against manual in-situ inspection results.

6.4.1. Data Description

The point clouds tested in this work are acquired on March 22, 2016, by the Fugro Drive-Map MLS system, which is shown in Figure 6.9. Figure 6.9a shows the MLS system as a whole while Figure 6.9b is a close up view of the sensors. The specifications of



(a) (b)

Figure 6.9: Drive-Map MLS system from Fugro. (a) A side view of the MLS system. (b) A view of the sensor configuration. This system was used to acquire the point clouds used in this work

the MLS system are given in Table 6.2.

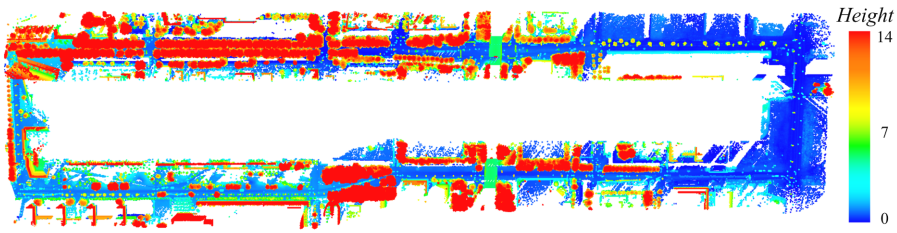
Laser Pulse Rate	1,333,000 p/s
Ranging Accuracy	<2 cm
Maximum Range	100 m
Scanner	Riegl VQ 250
Swath Angle	360 degrees
Panoramic Camera	Ladybug 3

Table 6.2: Specification of the Drive-Map mobile laser scanning system.

The point clouds acquired by the MLS system obtained in two opposite directions

have an average point density of 1,500 points per square meter. The point clouds cover a stretch of approximately 4 km of urban road. The number of points are 72,165,310 and 68,228,118 respectively. The MLS trajectory and the first point cloud are illustrated in Figure 6.10. In Figure 6.10a, the red line indicates the trajectory of the MLS system during the first acquisition. Figure 6.10b denotes the original point cloud of the first acquisition colored by height.

(a)



(b)

Figure 6.10: Overview of the scanning trajectory and point cloud. (a) Top view of the scanning trajectory of 4km long. (b) Original point cloud colorized by relative height.

The parameters and thresholds used in the tests are given in Table 6.3. Note that there are three parameters in the *Tiling* step. The *width* indicates the distance across trajectory boundary of the tiles, *length* indicates the distance along trajectory and *overlap* is the size of the buffer area between two consecutive tiles. The *Voxelization* level gives the maximum subdivision level of the octree. *Dimensionality* consists of linearity and planarity, which have dominant direction as described in Section 6.3.4. The SigVox 3D descriptor has two parameters: level indicates the number of scales, while similarity gives the threshold distance used to accept a candidate object as matching a training object.

6.4.2. Pre-Processing of the Point Cloud

As the original point clouds are too large to process on a normal desktop PC and points further away from the scanning trajectory are less interesting in this study, the original point clouds are divided into smaller tiles. In the re-tiling step, points that are further

Parameter	Value	
Tiling	width	20 (m)
	length	200(m)
	overlap	5 (m)
Voxelization level		9
Dimensionality	linearity (T_l)	10
	planarity (T_p)	20
Bounding box ($D_{i,j}$)		5.0(%)
SigVox	level	4
	similarity ($S^{c,t}$)	3.0

Table 6.3: Parameters and thresholds used in the processing of the two point clouds.

than 20 meter from the trajectory are removed. The length along the trajectory is 200 meters for each tile with an overlap of 5 meters between consecutive tiles. After re-tiling, ground and non-ground points are separated. The results of the re-tiling and separation of ground and non-ground points are given in Figure 6.11.

Figure 6.11a shows the bounding box of each of the 20 newly generated smaller tiles. As shown in this figure, each tile is labeled by a unique tile index. Figure 6.11b shows the separation results of the 20 tiles, in which the blue and red points denote the separated ground and non-ground points respectively. Table 6.4 gives the number of points corresponding to each pre-processing step for both data sets. As can be noticed 33,339,127 points are left after re-tiling the first point cloud. Consecutive segmentation results in 18,562,951 ground and 14,776,176 non-ground points respectively.

Point cloud	Original	After retiling	Ground Points	Non-Ground Points
1^{st} Run	72,165,310	33,339,127	18,562,951	14,776,176
2^{nd} Run	68,228,118	31,060,145	16,288,775	14,771,370

Table 6.4: Number of points after each pre-processing step.

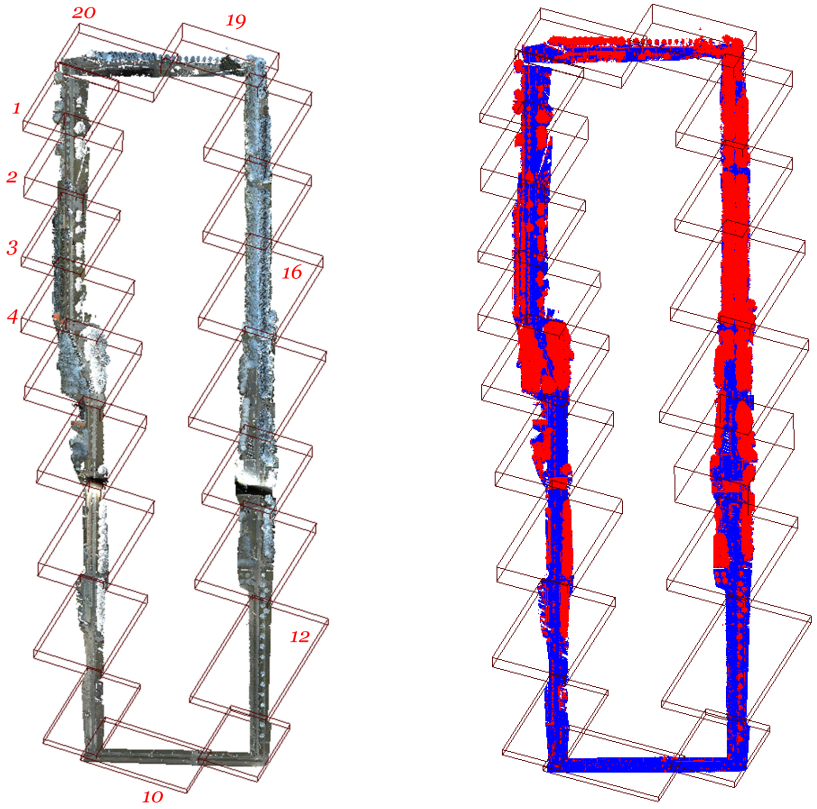
After the pre-processing of the original point clouds, non-ground points are used as input for the next step, i.e. voxelization and connected component clustering.

6.4.3. Voxelization and Clustering of Non-Ground Points

The non-ground points of each tile are organized in an octree data structure. This section describes the results of the voxelization and clustering of the non-ground points.

The voxelization of the non-ground points is conducted recursively in each tile, as described in Section 6.3.2. In this work, the criterion to stop subdivision is the minimum voxel size. The recursive subdivision of the octants in the octree is terminated whenever the voxel size is smaller than $10cm$. After voxelization, connected voxels are clustered. Consecutively, the points inside the voxels of a cluster are extracted to form a point cluster. Next, for each obtained point cluster, its 3D bounding box is obtained.

Figure 6.12 shows the results of voxelization and clustering of the non-ground points from tile 3 in Figure 6.11a. Figure 6.12a shows the non-ground points colorized by



(a) (b)

Figure 6.11: Re-tiling and non-ground point separation results of the first point cloud. (a) The point cloud was divided in 20 tiles and one tile contains approximately 4 million points. (b) Separation results: ground points are plotted in blue, and non-ground points in red. During the tiling points further than 20m from the trajectory are removed.

height. Figure 6.12b demonstrates an octree subdivision at level 9 corresponding to voxels of 39.4 cm. Figure 6.12c illustrates the results of clustering connected voxels. The clusters are colorized by random colors. Also, the points inside the voxels of each cluster are extracted to form corresponding point clusters. The 3D bounding boxes of those extracted point clusters are given in Figure 6.12d. Those 3D bounding boxes will be used to select candidate point clusters from a selected object of interest, as described in Section 6.3.2.

6.4.4. Object Recognition

Results of object recognition are presented in this section. In this chapter, 6 different street lamp poles and 4 different traffic signs were selected as objects of interest. Points corresponding to example objects were manually extracted as template point clusters

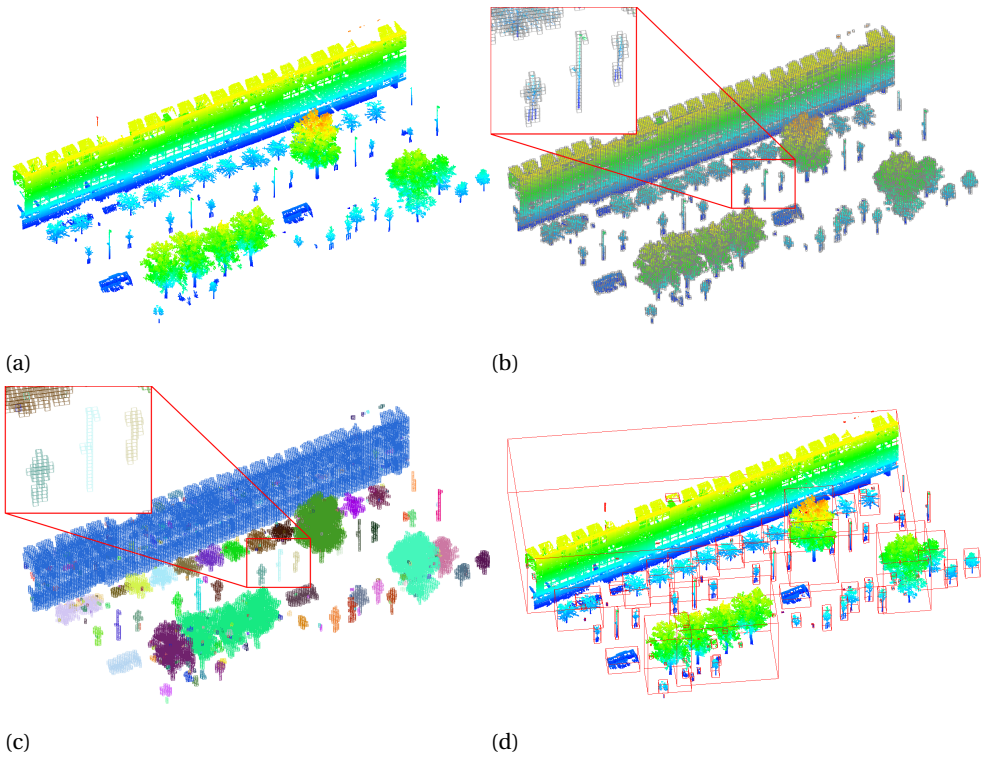


Figure 6.12: Results of voxelization and clustering of a tile of non-ground points. (a) Original non-ground points of tile 3. (b) Voxelization results of the non-ground points. (c) The connected voxels are clustered. (d) 3D bounding boxes of the considered point clusters. The resulting clusters are compared with the training objects, provided their bounding boxes do not deviate too much.

and object recognition was conducted on the rest of the point cloud. Figure 6.13 illustrates the selected objects. In the figure, the top row are photos of the selected street lamp poles and their sampled point clusters, while the bottom row shows the considered road signs and their corresponding point clusters.

Note that in the current research, one example object of interest was used to generate a training template. So far, the choice of this particular example has not been evaluated. For this there are some options. Firstly, select a few samples of the same type of object, then register the point clouds of those samples to form an average point cloud of the object. Consecutively, the SigVox feature template is generated based on the average point cloud. Secondly, select several samples of the same type of object and compute the point to point distance between those sample point clouds. The point cloud that has the smallest average point to point distance to the other samples is selected to generate the template for object recognition.

After importing the point clusters of the selected objects of interest, their 3D bounding boxes are obtained, as given in Table 6.5. Then, for candidate object selection, the 3D bounding box of each object of interest is compared with that of the voxel clusters

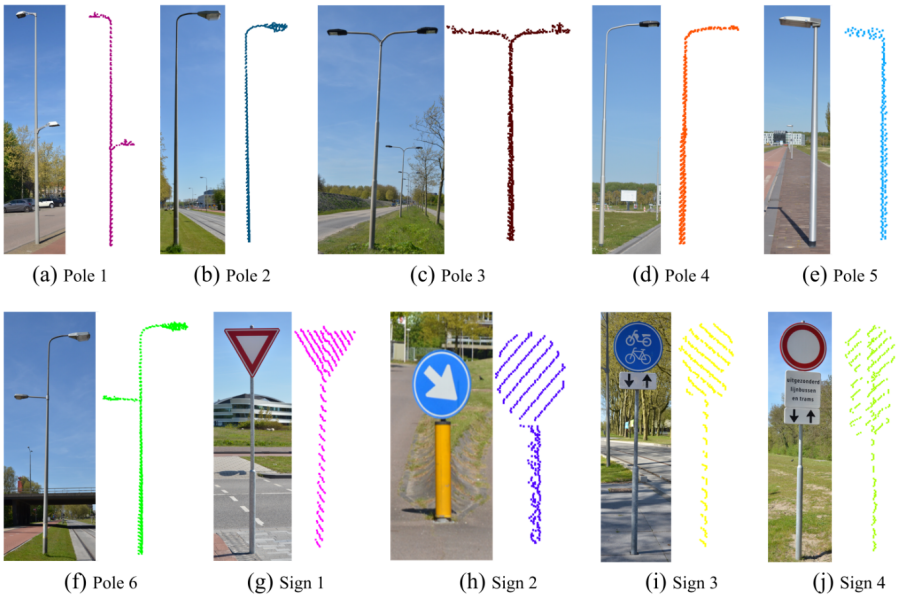


Figure 6.13: Images and point clouds of the selected objects of interest. (a), (b), (c), (d), (e) and (f) are photos and points of six lamp poles, while (g), (h), (i) and (j) are the considered traffic signs and their corresponding point clusters. The image shows that not all objects are sampled equally well by the MLS.

Object	Bounding box (m)		
	length	width	height
Pole 1	1.6	0.4	8.7
Pole 2	1.2	0.8	7.7
Pole 3	3.5	0.5	7.5
Pole 4	1.9	0.5	9.6
Pole 5	0.7	0.3	3.4
Pole 6	3.4	0.4	9.7
Sign 1	0.9	0.1	2.9
Sign 2	0.4	0.2	1.2
Sign 3	0.6	0.2	2.5
Sign 4	0.7	0.2	3.1

Table 6.5: Dimensions of the 3D bounding boxes of the objects of interest.These dimensions are used to select candidate objects which speeds up processing.

obtained in Section 6.4.3. In this work, the threshold for the similarity of 3D bounding boxes ($D_{i,j}$) is set to 5.0%. Take *Pole 2* and *Pole 4* in Figure 6.13 for example, the distance of their bounding boxes calculated from Equation 6.1 is 4.4%. Thus, not only the poles of type *Pole 2* in the test data will be selected as candidates, but also the poles of type *Pole 4*. In point cloud of the first run, there are 37 and 5 poles of type *Pole 2* and *Pole 4* are obtained as candidates of *Pole 2*.

In the next step, the SigVox 3D descriptors of *Pole 2* and all the 42 selected candidate

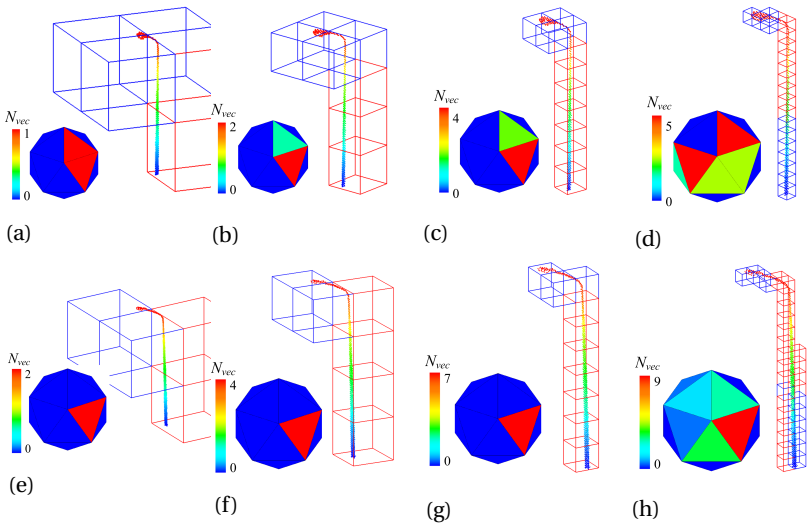


Figure 6.14: The voxel subdivision and the correspondent SigVox 3D feature descriptor of *Pole 2* and *Pole 4*of the four levels. (a), (b), (c) and (d) are the voxel subdivision and SigVox feature descriptor of *pole 2* at level 1, 2, 3 and 4. (e), (f), (g) and (h) are the voxel subdivision and SigVox feature descriptor of *Pole 4* at level 1, 2, 3 and 4. The red, green and blue voxel indicate the linearity, planarity and scatter of the points inside each voxel.

point clusters will be compared. Figure 6.14 shows the subdivision at 4 levels of *Pole 2* and *Pole 4* and the corresponding SigVox descriptors. Next, the similarity distances of between SigVox descriptors of the training point cluster of *Pole 2* and the obtained 42 candidates are determined. The resulting similarity distances for each of the 4 subdivision levels are shown in Figure 6.15. The histograms show that there is a clear difference between the similarity distance *Pole 2* and *Pole 4*. In this work, the similarity distance threshold is set to 3.0. Thus only candidates that have a similarity distance below 3.0 will be assigned to type *Pole 2*. In the point cloud of the first run, 37 poles of type *Pole 2* are correctly identified. This procedure is performed for all selected objects of interest.

For better visualization, the object recognition results are presented in two figures: Figure 6.16 and Figure 6.17. The object recognition results of the north part of the study area from the first point cloud are given in Figure 6.16. In Figure 6.16a, the green icons denote the correctly recognized objects. The red icons depict items that are not correctly identified. In Figure 6.16b, ground points are colored light blue and non-ground points gray. The successfully recognized objects are colored in correspondence to Figure 6.13. Figure 6.16b shows a scenario that has three lamp poles of type *Pole 1* in Figure 6.13. The proposed method identified two of them. The pole in black was missed because it is close to a bus stop. Therefore its points were in a cluster together with the bus stop points. As a result, the bounding box of this cluster was too far away from the bounding boxes of the training objects. Two road signs, of type *Sign 3* and *Sign 4* in Figure 6.13, which are highlighted by rectangles, are correctly recognized in Figure 6.16b. Figure 6.16c shows a scenario where seven lamp poles of type *Pole 1* and four road signs

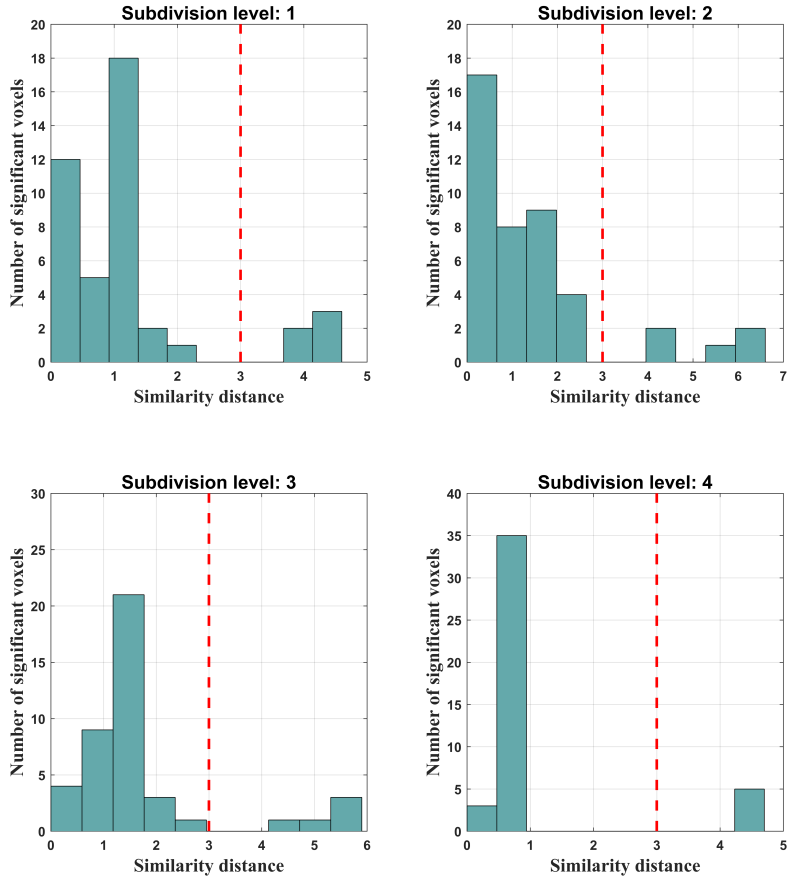
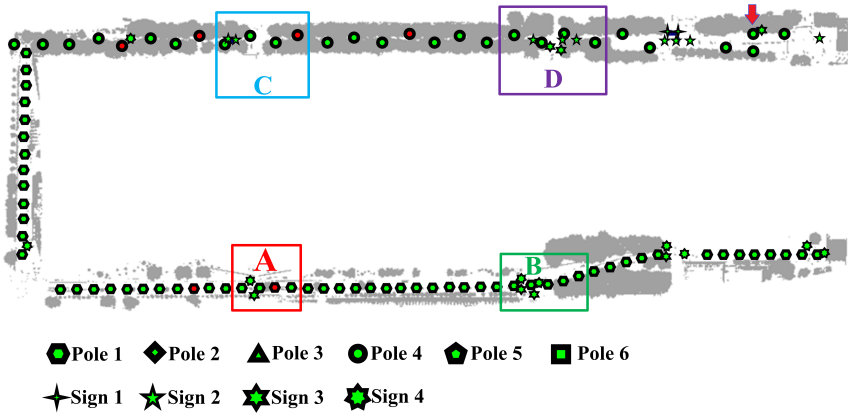


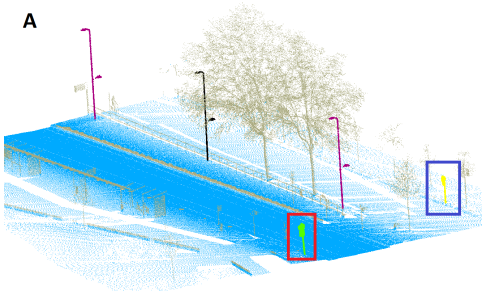
Figure 6.15: Similarity distances of the SigVox descriptor between template *Pole 2* to the obtained 42 candidates at 4 levels of subdivision. The red dash line in each subfigure denotes the threshold of similarity distance.

of type *Sign 3* and *Sign 4* were correctly recognized. In Figure 6.16d, there are actually four lamp poles of type *Pole 4*. However, one lamp pole, the black one, was not recognized because it is connected with the overhead roadside tree and was therefore not selected as candidate. In Figure 6.16e, four street lamps of type *Pole 4* and 5 road signs of type *Sign 2* and *Sign 3* were all correctly recognized.

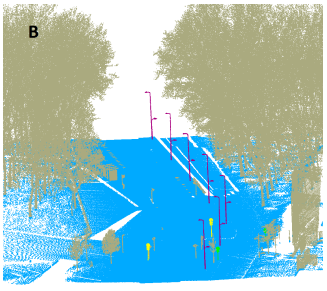
Figure 6.17 shows the recognition results from the south part of the study area. Figure 6.17a is a top view of the area. Figure 6.17b is a zoom in of Zone E in Figure 6.17a. Three poles of type *Pole 4* and *Pole 5* were successfully recognized. Also, two road signs of type *Sign 1* and one road sign of type *Sign 3* were correctly identified. However, one



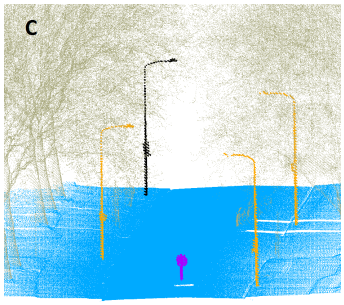
(a)



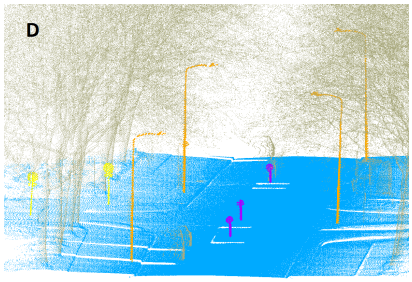
(b)



(c)

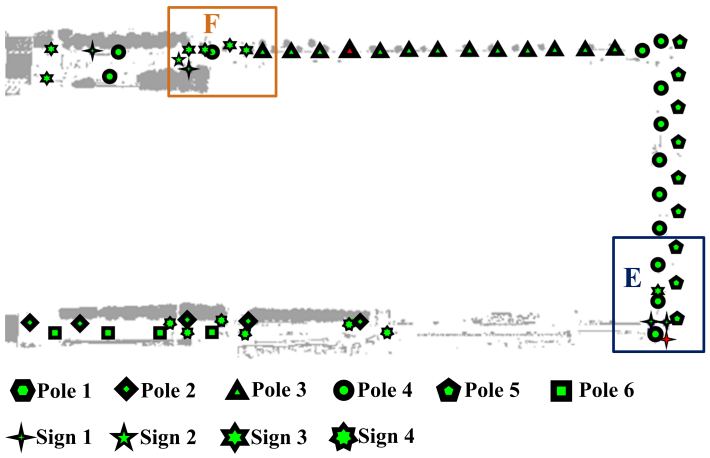


(d)

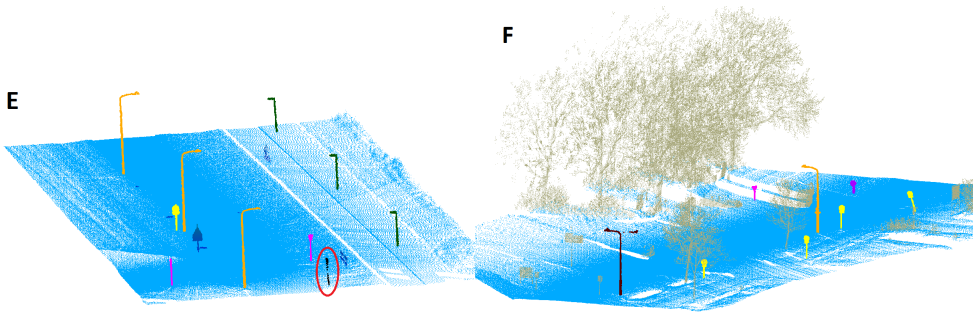


(e)

Figure 6.16: Street object recognition results from the north part of the study area. Different icons indicate different lamp pole and traffic sign types. Successfully identified objects are indicated in green, missed objects are colored red. (a) Overall results of the recognition. (b) Zoomed in view of area A. (c) Zoomed in view of area B. (c) Zoomed in view of area C. (d) Zoomed in view of area D. Each object type is colored in a different color.



(a)



(b) (c)

Figure 6.17: Street object recognition results from the south part of the study area. (a) Overall results of the recognition in the south part of the study area. (b) Zoomed in view of area E. (c) Zoomed in view of area F. The red ellipse indicates a street sign that was not detected by the workflow.

road sign of type *Sign 1* was not identified. This is because as a result of occlusion, only part of its shape is represented by the available points. In Figure 6.17c two lamp poles of type *Pole 3* and *Pole 4* were successfully identified. Four road signs of type *Sign 3* were correctly recognized. Also one sign of type *Sign 1* and one of type *Sign 2* were identified successfully.

6.4.5. Evaluation of Object Recognition Results

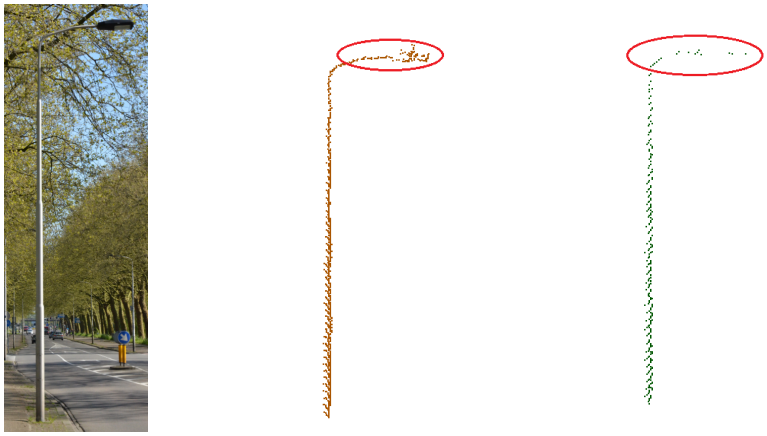
To validate the reliability and accuracy of the proposed road object recognition method, a second point cloud of the same area was collected, but from an opposite driving direction. This point cloud was processed by the same work flow. The ground truth of the street lamp poles and road signs was also collected by visual in situ inspection. The recognition results of the second point cloud and ground truth of the road objects are given in Table 6.6.

Object	Pole 1	1st Point Cloud	2nd Point Cloud	Ground Truth	Ground Truth Total
Street Lamp Pole	Pole 1	56	56	58	130
	Pole 2	37	39	41	
	Pole 3	12	12	12	
	Pole 4	5	5	5	
	Pole 5	9	9	9	
	Pole 6	4	4	4	
Road Sign	Sign 1	6	5	7	51
	Sign 2	10	9	10	
	Sign 3	15	16	16	
	Sign 4	16	18	18	
Object Total		170	173	181	181

Table 6.6: Results of road object recognition from the two point clouds and in situ inspected ground truth.

As illustrated in Table 6.6, 123 and 125 from a total of 130 street lamp poles are correctly recognized in the two point clouds. Thus the accuracy of street lamp poles recognition rates are 94% and 96% for the first and the second point cloud. There are 47 and 48 road signs correctly recognized from a total of 51 road signs. Therefore the accuracy of road sign recognition for the two point clouds is 92% and 94%. The overall accuracy of the road structure recognition are 94% and 96% for the two point clouds.

There are a few cases where poles are correctly identified in the point cloud of the first run, but missed in the second run. These inconsistencies are caused either by occlusions, or by a too large distance of an object to the scanning trajectory. These factors result in incomplete sampling of the objects and resulting in deviating 3D bounding box sized. As a result their clusters were not selected as a candidate object. An example is the road sign of type *Sign 1*in Figure 6.17b. However, if the full shape of an object is sampled, still the object can be successfully identified even if there exists a big difference in point density. Notably, the proposed method is able to identify poles sampled at different point density. Figure 6.18a shows the lamp pole denoted by the red arrow in Figure 6.16a. As the distances of the pole to the scanning trajectory are different, the sampled point density is different as well. Figure 6.18b and Figure 6.18c are the point



(a) (b) (c)

Figure 6.18: A lamp pole of type *Pole 4* represented in two point clouds obtained from opposite driving directions. (a) A photo of the pole. (b) Points from the first point cloud. (c) Points from the second point cloud. In the second run, the pole was sampled by 273 points compared to 954 points in the first run, which strongly affects the local point density as visible in the area marked by the ellipse.

clouds of the pole sampled from two opposite driving directions, consisting of 954 and 273 points respectively. There is a big difference in point density at the top of the pole as indicated in the figures. Still, the pole was correctly recognized in both point clouds.

Recognition may fail if an object is too close to another object. For example, Figure 6.19 shows a scenario where a lamp pole is connected to a road side tree. As a consequence the lamp pole is not separated in the clustering step. Subsequently it was not considered in the candidate selection step and finally not recognized. Further work should consider the separation of apparent connected objects.

6.5. Discussion and Conclusions

6.5.1. Discussion

In this section, the sensitivity analysis of the used parameters is firstly given. Then, future work on some aspects of the proposed method are discussed.

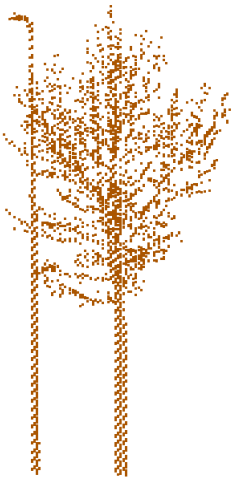
Sensitivity analysis

The parameters used in this work are given in Table 6.3. This section gives a short analysis of the influence of the parameters to the results.

1. Voxelization level. The level of voxelization corresponds to the size of the voxels, the deeper the level, the smaller the voxel size. The parameter should be set considering the minimum distance between objects of interest and the surroundings, as well as the average point density.

2. Dimensionality. The linearity and planarity thresholds will define the dimensionality of a voxel using PCA. The smaller the thresholds, the more significant

(a)



(b)

Figure 6.19: A scenario where a lamp pole was not identified. Because the tree and the pole are too close, they are clustered together, which negatively effects the shape encoding. (a) A lamp pole connected to a road side tree. (b) Point cloud of the lamp pole and the tree.

voxels will contribute their eigenvectors to the SigVox descriptor. The optimal thresholds should also consider the noise level of the point clouds.

3. Bounding box. This parameter is used to remove point clusters that have a deviating 3D bounding box compared to those of the training objects at an early stage. This avoids considering all the obtained point clusters and speeds up the processing. However, a too small bounding box buffer will cause omissions.

4. SigVox. The number of required levels of the SigVox descriptor depends on the complexity of the geometric shape of the selected objects of interest. A too small number of levels will result in robustness issues. The similarity distance threshold depends on the similarity of the considered objects. Smaller thresholds may leads to omissions in the recognition results.

Future work

Still some aspects of the presented method should either be further considered or improved.

1. Object separation. As the results in Section 6.4.4 shows, objects were unsuccessfully recognized because of their proximity to other objects. To enhance performance, separation of objects of interest needs to be further improved.

2. Candidate selection. In this work, candidate point clusters are selected by comparing their 3D bounding boxes with the 3D bounding boxes of the example objects of interest. However, there may be situations in which poles are inclined.

Such cases are notably interesting for street inventory management. Probably the method proposed in this work will not identify inclined street poles, because of the initial bounding box selection step.

3. Approximation of EGI. The EGI descriptor is approximated by an icosahedron, which has 20 boundary triangles. If the shape of an object is extremely complicated, the icosahedron may not be sufficient for representing the shape of the object. However, the icosahedron can be further tessellated by incrementally subdividing one triangle into four smaller triangles until the approximation meets the requirement.

4. Threshold selection. The used parameters are mainly set with regard to the experimental results in this work. A next work should consider automatic threshold selection.

6.5.2. Conclusions

In this chapter, the following research question is answered:

How to automatically identify objects in the urban road environment in MLS point clouds?

Firstly, an automatic method for roadside object identification is proposed and validated. The method consists of four steps, i.e. pre-processing, voxelization and SigVox descriptor construction, template matching, and result validation. The proposed method was tested on two point clouds sampling the same stretch of 4 km of urban road obtained by a MLS system driving in opposite direction. In this study, 6 different types of street lamp poles and 4 types of road signs were selected as objects of interest and the SigVox descriptor of those objects were constructed as template objects. The recognition was performed by computing the distance between the SigVox descriptors of template objects and candidate point clusters. The recognition results of the two point clouds were compared to ground truth data of the street objects obtained by in situ visual inspection. The comparison results show that the overall accuracy of road structure recognition is 94% and 96% for the two point clouds. To the best of our knowledge, this is the first time that a shape descriptor, describing complete objects is used to efficiently extract repetitive objects in large point cloud scenes.

7
Conclusions

This chapter first summaries the conclusions of this thesis addressing the research question as proposed in Chapter 1. Answers to the sub-questions are provided. Next, recommendations for future work related to the methodology presented in this thesis are given.

7.1. Conclusions

As introduced in Chapter 1, the main research question of this thesis is:

How to extract efficiently geometric information on the road environment from huge MLS point clouds?

Five sub-questions were proposed to answer the main question. In summary, huge point cloud data sets collected by MLS systems are organized by scalable and efficient data structures, voxels and octrees. Rather than manipulating the data sets point by point, data processing and analysis were performed on 2D raster grids, and 3D voxels at a variety of scales. The 2D grids were used to estimate surface direction. Adjacency in clustered 3D voxels combined to shape analysis at voxel level turned out to be a powerful way to extract, separate and even identify objects in 3D space. Each proposed method was evaluated by tests and the feasibility, i.e. reasonable computation time, number of parameters and output quality, was validated as well. Detailed answers to the five sub-questions are given below:

1. **What is the current status of MLS systems and the data processing work-flow?**

   This question is answered in Chapter 2. First, the components of a typical MLS system were introduced and their roles were discussed. Then, an overall review of state-of-the-art of the data processing work-flow of MLS point clouds was provided. In summary, the typical MLS system hardware consists of laser scanners, cameras and positioning and orientation systems. The laser scanners provide distance measurements. The positioning and orientation systems determine the location and orientation of each emitted laser pulse. By combining the measurements of the components, the 3D coordinates of sampled objects are acquired. Next, a review on the processing of point cloud data sets collected by a MLS system is provided. In general, the work-flow starts with filtering and segmentation of the raw point clouds. In the filtering step, the ground and non-ground points, as well as noise and outliers are removed. In the segmentation step, points representing different sampled objects are separated. The last stage is applications. The obtained point clusters and segments are used for different purposes, such as extraction of road geometry, curbstone and road markings, railway, highway and tunnel monitoring and different environmental applications.

2. **How to organize efficiently the huge MLS point clouds?**

   MLS systems are efficient in data acquisition. Typically, the systems are able to collect 3D points at a speed of approximately 1 million points per second. What's more, besides the 3D points, the scanning trajectory, images and videos are also acquired. Thus, the collected data sets at a city level is huge. It is a challenge to handle 3D data

sets of such large volume, whether in storage and loading or to manipulate, i.e. point querying and modeling. An intuitive approach is first to subset the huge data sets into smaller tiles. In some commercial software, like micro-station of TerraSolid, re-tiling is performed by cropping point clouds with regard to geo-referenced grids. Although the size of those grids can be set by users, division is somehow arbitrary and the scanning geometry is not easy to incorporate. This results in many tiles having few points and several tiles having a large number of points. In this work, a re-tiling strategy based on the scanning trajectory was introduced. Since point clouds are obtained along a scanning trajectory, point density is varying from high to low with increasing distance from the trajectory. The dimensions of the tiles, i.e. length and width along and across trajectory respectively, are pre-set by the user based on the application considered. Points of each tile are organized with different data structures for specific applications. Even if the number of points in a tile is much smaller than that in the original point cloud, the number of points is still often too large to manipulate efficiently. In this thesis, voxels and octrees were used to re-sample the re-tiled points. Their properties, such as agility in dimensions and efficiency in neighborhood querying, are exploited in the design of algorithms for different applications in this work.

3. **How to estimate the excavation volume on mountain roads and water flow direction from MLS point clouds?**

   This question is answered in Chapter 4. Mountainous roads are lifelines of rural areas but they are prone to natural hazards, such as landslides, rock fall and water erosion. Because of the complicated morphological environment, it is expensive to monitor the status of those roads. Also, engineering work on mountainous roads, i.e. road widening, needs data to estimate the excavation volume that has to be removed. MLS systems are able to efficiently acquire dense and accurate point clouds of the road environment. In Chapter 4, a strategy for automatic estimation of excavation volume and water-flow directions was presented. The method starts by down-sampling the original point clouds using a uniform-size voxel. Then, local normals and 2D slopes were estimated at each resulting point to separate road and non-road points. For excavation volume estimation, a digital surface model was generated using the obtained terrain points. Next, the road was sliced along the scanning direction and the excavation volume of each slice was accumulated. The estimation is obtained by adding the accumulated volumes on the two roadsides. To estimate water flow directions, the obtained terrain points were used firstly to generate 2.5D grids. Next, a gradient analysis was employed to determine water flow direction. Finally, by accumulating the number of in-flow grid cells, the locations were estimated where erosion is more likely. The feasibility of the presented strategy was verified on a stretch of mountainous road. The proposed strategies on excavation volume and water flow estimation are generally applicable in aspects that both the methods follow the common data processing workflow as presented in Chapter 2 and only has several parameters that related to computation time and estimation quality.

4. **How to individualize roadside trees from MLS point clouds?**

   One answer to this question is given in Chapter 5. Monitoring and documenting parameters such as height, diameter at breast height and canopy diameter of roadside

trees are non-trivial tasks. Not only the number of trees is large, but also trees are growing and thus have to be measured regularly. Furthermore, roadside trees are often almost touching each other which makes the surveying of the parameters harder. The method takes tree points as input, starts with re-sampling the tree points by voxels, followed by clustering based on voxel adjacency. Consecutively, a vertical transect starting from suitable seed voxels in combination with a novel adjacency analysis is used to assign voxels of the same tree into one class. Finally, points within those voxels are identified and thus the trees are individualized. The method is scalable as the voxel sizes in the three coordinate axis directions are changeable depending on the requirements on quality and computation time. The method was verified by a series of tests.

5. **How to automatically identify objects in the urban road environment in MLS point clouds?**

   In Chapter 6, the answer to this question is given. To automatically identify roadside objects from the 3D MLS point clouds is challenging. The possible difficulties lie in the variety of objects, the complex shape of objects and imperfections in the data. In this thesis, an automatic method for identifying objects sampled by MLS point clouds is presented. The method takes raw MLS point clouds and trajectory files as input. Retiling is performed before ground points are removed and voxelization is performed on the non-ground points using an octree. Consecutively, connected voxels are clustered together to represent potential individual objects. Then, a 3D significant eigenvector based shape descriptor using voxels (SigVox) is introduced. Notably, the SigVox is scalable and based on the entire shape of objects. Similarity matching of SigVox is performed between training objects and the candidate individual point clusters. The proposed method was validated on point clouds of 4km of urban road around the TU Delft campus and 10 types of objects of the road were identified with promising accuracy.

The main research question of this study was addressed by answering the five subquestions consecutively. In this study, the huge point clouds collected by MLS systems were firstly organized by either voxels or octrees. This enables efficient processing and storage based on voxels instead of individual discrete points. Both urban and mountain road environments, which consists of road surface, roadside terrain, roadside trees and furniture, were considered in this study. Geometric information of those road environments, such as terrain volume, water flow directions, location or shape of individual trees, lamp poles and traffic signs, were studied and analyzed. Different strategies were proposed and presented in 2.5D and 3D respectively for efficient extraction of the aforementioned geometric information. Gradient and normal based methodologies were introduced to estimate terrain volume and water flow direction in 2.5D. Algorithms based on 3D adjacency and intrinsic shape analysis at object level were presented to extract geometric information in 3D. The tests verified the general applicability and performance of the corresponding presented algorithms.

In general, the proposed methods can be directly further extended to point clouds obtained by other mobile laser scanners, such as backpack systems, compare Figure 2.6,

as well as TLS and ALS systems. Often, the presented methodology can be used seamlessly. For example, the excavation volume estimation algorithm, Chapter 4, can be used to estimate landslide volumes from ALS point clouds. Note that this has not been tested. The algorithm for tree individualization from MLS point clouds, Chapter 5, can also be applied to separate forest trees sampled by TLS systems as already shown in Chapter 5. The presented method for object recognition is able to conduct registration of TLS point clouds as well (Wang et al., 2016). Thus, the application scenarios of the proposed methods are not restricted to road environments.

7.2. Recommendations

On several aspects the methods presented in this thesis can be further investigated and probably improved.

Self-adaptive parameter setting

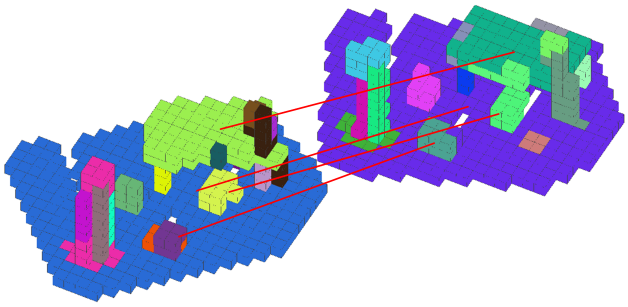
In this work, the proposed methods are all automatic, i.e. excavation volume and water-flow direction estimation, voxel based tree individualization and 3D SigVox based feature matching for object identification. There are parameters that needs to be specified by the user as they define the quality of outputs. For example, voxel size is a key parameter that influences the processing time and quality of results in water-flow direction estimation, tree separation and clustering in object identification. However, other parameters, such as the mean tree diameter in tree individualization and thresholds for feature similarity in object identification, could probably be determined automatically during processing. Those parameters can be self-adaptively determined either by machine learning strategies or statistical analysis.

Switch from voxels to octrees

In this work, some processing is performed based on voxels. Voxels have their advantages, such as the possibility of having agile voxel sizes in three axis directions, the ease to construct them and to use them in neighborhood searching. However, using voxels often introduces memory redundancy. During a voxelization step, the total number of voxels is determined by the bounding box of the entire input point clouds and pre-set voxel sizes. Gaps in the point cloud still occupy memory. Therefore, efficient use of voxels requires either an additional tiling step, or the use of relatively big voxels. An octree is a hierarchical data structure that determines and incorporates spatial occupation during voxelization. Furthermore, neighborhood searching can be performed by looking up the addresses of the neighboring nodes. Replacing voxels in tree individualization for example will allow to input larger point cloud tiles and have smaller voxel sizes if required.

Assess properties of the identified objects

In Chapter 6, the presented 3D SigVox feature matching method is able to identify objects on the roadside. However, the application of the SigVox feature descriptor and the identified objects can be further extended. Figure 7.1a shows an example of coarse point cloud registration in voxel space based on the correspondences obtained by SigVox feature matching. Road furniture documentation, such as furniture location,



(a)

(b) (c)

Figure 7.1: Further applications of the SigVox feature descriptor and the identified objects.

orientation and functionality and monitoring the state of individual furniture can be conducted based on the identified objects. Figure 7.1b illustrates three identified lamp poles of the same type. However, height and orientations of the poles are different due to inaccurate installation, traffic accidents or wind. Determining the vertical angles of the poles may give an indication of poles that need maintenance. Figure 7.1c shows an example of a similar scenario for road signs.

High performance computing

So far, all the processing and testing of the algorithms proposed in this thesis were performed on a desktop PC. The work-flows were not yet parallelized. For work at industry level, point clouds to be processed may sample complete cities and will be huge. Processing of those kind of data sets may require longer computation time and larger storage space. High performance computing will speed up the processing either by running the algorithms on a super-computer or parallelizing the work-flow. The parallelization can be carried out by first re-tiling the point clouds, then assigning a number of tiles to a core for processing and finally merging the results from the tiles. Also, as

the points in tiles are organized by an octree, parallelization can also be conducted by distributing points in each sub-octant of an octree to cores and finally merging the results.

Bibliography

Abellán, A., Calvet, J., Vilaplana, J. M., and Blanchard, J. (2010). Detection and spatial prediction of rockfalls by means of terrestrial laser scanner monitoring. *Geomorphology*, 119(3-4):162–171.

Abo-Akel, N., Filin, S., and Doytsher, Y. (2009). Reconstruction of Complex Shape Buildings from Lidar Data Using Free Form Surfaces. *Photogrammetric Engineering & Remote Sensing*, 75(3):271–280.

Agamennoni, G., Nieto, J. I., and Nebot, E. M. (2011). Robust inference of principal road paths for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):298–308.

Aijazi, A., Checchin, P., and Trassoudaine, L. (2013). Segmentation Based Classification of 3D Urban Point Clouds: A Super-Voxel Based Approach with Evaluation. *Remote Sensing*, 5(4):1624–1650.

Alho, P., Vaaja, M., Kukko, A., Kasvi, E., Kurkela, M., Hyyppä, J., Hyyppä, H., and Kaartinen, H. (2011). Mobile laser scanning in fluvial geomorphology: mapping and change detection of point bars. *Zeitschrift für Geomorphologie, Supplementary Issues*, 55(March):31–50.

Allen, K. (1971). Patterns and search statistics. In Rustagi, J. S., editor, *Optimizing Methods in Statistics*, pages 303 – 337. Academic Press.

Alliez, P., Berberich, E., and Fabri, A. (1997). CGAL - the computational geometry algorithms library. `http://www.cgal.org/index.html`. Accessed: 2016.10.10.

Anonymous (1957). *Review of Geodetic and Mapping Possibilities*. Cooperative Society for Geodesy and Cartography.

Anselin, L., Syabri, I., and Kho, Y. (2006). GeoDa: An introduction to spatial data analysis.

Aschoff, T., Thies, M., and Spiecker, H. (2004). Describing forest stands using terrestrial laser-scanning.

Axelsson, P. (2000). DEM generation from laser scanner data using adaptive TIN models. *International Archives of Photogrammetry & Remote Sensing*, 33:110–117.

Babahajiani, P., Fan, L., and Gabbouj, M. (2015a). Object recognition in 3d point cloud of urban street scene. In *Computer Vision - ACCV 2014 Workshops*, volume 9008 of *0302-9743*, pages 177–190. Springer International Publishing, Singapore.

Babahajiani, P., Fan, L., Kamarainen, J., and Gabbouj, M. (2015b). Automated super-voxel based features classification of urban environments by integrating 3d point cloud and image content. In *2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 372–377.

Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122.

Baltsavias, E. (1999). Airborne laser scanning: basic relations and formulas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):199–214.

Barber, D., Mills, J., and Smith-Voysey, S. (2008). Geometric validation of a ground-based mobile laser scanning system. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):128–141.

Barboza, D. C. and Clua, E. W. G. (2011). GPU-based data structure for a parallel ray tracing illumination algorithm. In *Brazilian Symposium on Games and Digital Entertainment, SBGAMES*, pages 11–16.

Batty, M., Axhausen, K. W., Giannotti, F., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., and Portugali, Y. (2012). Smart cities of the future. *European Physical Journal: Special Topics*, 214(1):481–518.

Belton, D. and Lichti, D. (2006). Classification and segmentation of terrestrial laser scanner point clouds using local variance information. *Iaprs, Xxxvi*, 5(1):44–49.

Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517.

Bentley, J. L. (1990). K-d trees for semidynamic point sets. *Proceedings of the sixth annual symposium on Computational geometry*, pages 187–197.

Beserra Gomes, R., Ferreira da Silva, B. M., Rocha, L. K. d. M., Aroca, R. V., Velho, L. C. P. R., and Gonçalves, L. M. G. (2013). Efficient 3D object recognition using foveated point clouds. *Computers & Graphics*, 37(5):496–508.

Bi, H., Ao, Z., Zhang, Y., Zhang, K., and Tang, C. (2014). *Organization of LiDAR Point Cloud Based on 2D*, pages 2161–2168. Springer New York, New York, NY.

Bienert, A., Queck, R., Schmidt, A., Bernhofer, C., and Maas, H.-G. (2010). Voxel Space Analysis of Terrestrial Laser Scans in Forests for Wind Field Monitoring. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences, Commission V Symposium*, XXXVIII, P(Section 6):92–97.

Biosca, J. M. and Lerma, J. L. (2008). Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):84–98.

Bishop, R. (2000). A survey of intelligent vehicle applications worldwide. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*, pages 25–30.

Bitenc, M., Lindenbergh, R., Khoshelham, K., and van Waarden, A. P. (2011). Evaluation of a LIDAR land-based mobile mapping system for monitoring sandy coasts. *Remote Sensing*, 3(7):1472–1491.

Boavida, J., Oliveira, A., and Santos, B. (2012). Precise long tunnel survey using the riegl vmx-250 mobile laser scanning system. In *Riegl International Airborne and Mobile User Conference*, Orlando, Florida.

Börcs, A., Nagy, B., and Benedek, C. (2015). *Fast 3-D Urban Object Detection on Streaming Point Clouds*, pages 628–639. Springer International Publishing, Cham.

Boulaassal, H., Landes, T., Grussenmeyer, P., and Tarsha-Kurdi, F. (2007). Automatic Segmentation of Building Façades using Terrestrial Laser Data. *The International Archives of Photogrammetry and Remote Sensing*, XXXVI:65–70.

Boyko, A. and Funkhouser, T. (2011). Extracting roads from dense point clouds in large scale urban environment. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6 SUPPL.).

Bremer, M., Schmidtner, K., and Rutzinger, M. (2015). Reconstruction of forest geometries from terrestrial laser scanning point clouds for canopy radiative transfer modelling. In *EGU General Assembly Conference Abstracts*, volume 17 of *EGU General Assembly Conference Abstracts*, page 11819.

Bremer, M., Wichmann, V., and Rutzinger, M. (2013). eigen value and graph-based object extraction from mobile laser scanning point clouds. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume II-5/W2, pages 55–60, Antalya, Turkey. ISPRS Workshop Laser Scanning 2013.

Brenner, C. (2005). Building reconstruction from images and laser scanning. *International Journal of Applied Earth Observation and Geoinformation*, 6(3-4):187–198.

Brenner, C. (2009). Extraction of features from mobile laser scanning data for future driver assistance systems. In *Lecture Notes in Geoinformation and Cartography*, pages 25–42.

Briese, C., Pfeifer, N., and Dorninger, P. (2002). Applications of the robust interpolation for DTM determination. *International Archives of Photogrammetry and Remote Sensing*, pages 55–61.

Browell, E. V. and Center., L. R. (1977). *Analysis of laser fluorosensor systems for remote algae detection and quantification*. Washington, D.C. :National Aeronautics and Space Administration ;. http://www.biodiversitylibrary.org/bibliography/4830 — Cover title. — Prepared at Langley Research Center.

Brown, R. A. (2014). Building a Balanced k-d Tree in O(kn log n) Time. *ArXiv e-prints*.

Bucksch, A. and Lindenbergh, R. (2008). CAMPINO - A skeletonization method for point cloud processing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):115–127.

Bufton, J. (1989). Laser altimetry measurements from aircraft and spacecraft. *Proceedings of the IEEE*, 77(3):463–477.

Cabo, C., Ordoñez, C., García-Cortés, S., and Martínez, J. (2014). An algorithm for automatic detection of pole-like street furniture objects from Mobile Laser Scanner point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87:47–56.

Cahalane, C., McCarthy, T., and McElhinney, C. (2010). Mobile Mapping System Performance - An Initial Investigation into the Effect of Vehicle Speed on Laser Scan Lines. In *Proceedings of Remote Sensing and Photogrammety Society Annual Conference*, pages 1–8.

Castillo, E. (2013). Point Cloud Segmentation via Constrained Nonlinear Least Squares Surface Normal Estimates. *Recent UCLA Computational and Applied Mathematics Reports*, pages 1–6.

Chao, Y., Wu, T., Wang, X., and Zheng, G. (2015). The computation of delaunay triangulation of lidar point cloud based on gpu. In *2015 23rd International Conference on Geoinformatics*, pages 1–4.

Chen, F. and Lu, C.-T. (2008). *Nearest Neighbor Query, Definition*, pages 782–783. Springer US, Boston, MA.

Chen, Q., Gong, P., Baldocchi, D., and Xie, G. (2007). Filtering Airborne Laser Scanning Data with Morphological Methods. *Photogrammetric Engineering Remote Sensing*, 73(2):175–185.

Chen, Y., Zhang, L., and Xu, H. (2011). *Algorithm Research on Delaunay TIN Generation and Real Time Updating*, pages 751–757. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chengming, Z. C. Z. and Zhiyong, T. Z. T. (2009). The Collision Detection Algorithm Based on the Combination of Two-Dimensional and Dynamic Octree. *2009 International Conference on Environmental Science and Information Application Technology*, 1:1–7.

Chrószcz, A., Lukasik, P., and Lupa, M. (2016). Analysis of performance and optimization of point cloud conversion in spatial databases. *IOP Conference Series: Earth and Environmental Science*, 44(5):052011.

Cifuentes, R., Van der Zande, D., Farifteh, J., Salas, C., and Coppin, P. (2014). Effects of voxel size and sampling setup on the estimation of forest canopy gap fraction from terrestrial laser scanning data. *Agricultural and Forest Meteorology*, 194:230–240.

Clarke, G. L., Ewing, G. C., and Lorenzen, C. J. (1970). Spectra of backscattered light from the sea obtained from aircraft as a measure of chlorophyll concentration. *Science (New York, N.Y.)*, 167(3921):1119–1121.

Cohen, J. (1960). A coefficient of agreement of nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Conway, J., Burgiel, H., and Goodman-Strauss, C. (2008). *The Symmetries of Things*. Ak Peters Series. Taylor and Francis.

Cormen, T., Leiserson, C., and Rivest, R. (2001). *Introduction to Algorithms*. MIT Press.

Cottone, N. and Ettl, G. (2001). Estimating populations of whitebark pine in Mount Rainier National Park, Washington, using aerial photography. *Northwest Science*, 75(4):397–406.

Crassidis, J. L. (2006). Sigma-point Kalman filtering for integrated GPS and inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 42(2):750–756.

Cura, R., Perret, J., and Paparoditis, N. (2015). Point Cloud Server (Pcs) : Point Clouds in-Base Management and Processing. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W5(August):531–539.

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

de la Puente, P., Rodríguez-Losada, D., López, R., and Matía, F. (2008). *Extraction of Geometrical Features in 3D Environments for Service Robotic Applications*, pages 441–450. Springer Berlin Heidelberg, Berlin, Heidelberg.

Delaunay, B. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2.

Devllers2002, O., PION, S., and TEILLAUD, M. (2002). Waling in a triangulation. *International Journal of Foundations of Computer Science*, 13(02):181–199.

Dey, T. K., Li, G., and Sun, J. (2005). Normal estimation for point clouds: a comparison study for a Voronoi based method. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 39–46.

Díaz-Vilarino, L., González-Jorge, H., Martínez-Sánchez, J., Bueno, M., and Arias, P. (2016). Determining the limits of unmanned aerial photogrammetry for the evaluation of road runoff. *Measurement*, 85:132 – 141.

Döllner, J. and Buchholz, H. (2005). Continuous level-of-detail modeling of buildings in 3D city models. *Proceedings of the 2005 international workshop on Geographic information systems GIS 05*, 33(5):173.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A. (2011). On the segmentation of 3D lidar point clouds. *Icra*, pages 2798–2805.

Duncanson, L., Cook, B., Hurtt, G., and Dubayah, R. (2014). An efficient, multi-layered crown delineation algorithm for mapping individual tree structure across multiple ecosystems. *Remote Sensing of Environment*, 154:378–386.

El-Halawany, S. and Lichti, D. (2011). Detection of road poles from mobile terrestrial laser scanner point cloud. In *Multi-Platform/Multi-Sensor Remote Sensing and Mapping (M2RSM), 2011 International Workshop on*, pages 1–6.

Elberink, S. and Khoshelham, K. (2015). Automatic Extraction of Railroad Centerlines from Mobile Laser Scanning Data. *Remote Sensing*, 7(5):5565–5583.

Ellum, C. and El-Sheimy, N. (2002). Land-based mobile mapping systems. *Photogrammetric Engineering and Remote Sensing*, 68(1):15–17.

Elseberg, J., Borrmann, D., and Nuchter, A. (2011). Efficient processing of large 3D point clouds. *2011 XXIII International Symposium on Information, Communication and Automation Technologies*, (October):1–7.

Elseberg, J., Borrmann, D., and Nüchter, A. (2013). One billion points in the cloud - An octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76:76–88.

Evans, J. S. and Hudak, A. T. (2007). A multiscale curvature algorithm for classifying discrete return LiDAR in forested environments. *IEEE Transactions on Geoscience and Remote Sensing*, 45(4):1029–1038.

Fan, H., Yao, W., and Tang, L. (2014). Identifying man-made objects along urban road corridors from mobile lidar data. *IEEE Geoscience and Remote Sensing Letters*, 11(5):950–954.

Fekete, S., Diederichs, M., and Lato, M. (2010). Geotechnical and operational applications for 3-dimensional laser scanning in drill and blast tunnels. *Tunnelling and Underground Space Technology*, 25(5):614–628.

Fischler, M. a. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Flood, M. and Guteliue, B. (1997). Commercial implications of topographic terrain mapping using scanning airborne laser radar. *photogrammetry engineering and remote sensing*, 63:363–366.

Floriani, L. D. and Magillo, P. (2009). *Triangulated Irregular Network*, pages 3178–3179. Springer US, Boston, MA.

Foy, S., Deegan, C., Mulvihill, F., Fitzgerald, C., Markham, C., and McLoughlin, S. (2007). Road sign safety identification through the use of a mobile survey system. In *In Proceedings of the International Symposium on Mobile Mapping Technology*, volume XXXVI-5/C55, Padua, Italy.

Frank, A. U. (1992). Spatial concepts, geometric data models, and geometric data structures. *Computers and Geosciences*, 18(4):409–417.

Fregonese, L., Barbieri, G., Biolzi, L., Bocciarelli, M., Frigeri, A., and Taffurelli, L. (2013). Surveying and monitoring for vulnerability assessment of an ancient building. *Sensors (Basel, Switzerland)*, 13(8):9747–9773.

Fröhlich, C. and Mettenleiter, M. (2004). Terrestrial Laser Scanning - New Perspectives in 3D Surveying. *Proceedings of the ISPRS Working Group VIII/2: Laser-Scanners for Forest and Landscape Assessment*, pages 7–13.

Frueh, C. and Zakhor, A. (2003). Constructing 3D city models by merging ground-based and airborne views. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, 2:II–562–9.

Garvin, J., Bufton, J., Blair, J., Harding, D., Luthcke, S., Frawley, J., and Rowlands, D. (1998). Observations of the Earth's topography from the Shuttle Laser Altimeter (SLA): Laser-pulse Echo-recovery measurements of terrestrial surfaces. *Physics and Chemistry of the Earth*, 23(9-10):1053–1068.

Gikas, V. (2012). Three-dimensional laser scanning for geometry documentation and construction management of highway tunnels during excavation. *Sensors (Switzerland)*, 12(8):11249–11270.

Gikas, V. and Daskalakis, S. (2008). DETERMINING RAIL TRACK AXIS GEOMETRY USING SATELLITE AND TERRESTRIAL GEODETIC DATA. *SURVEY REVIEW*, 40(310):392–405.

Gikas, V. and Stratakos, J. (2012). A novel geodetic engineering method for accurate and automated road/railway centerline geometry extraction based on the bearing diagram and fractal behavior. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):115–126.

Girardeau-Montaut, D. (2003). Cloudcompare - a 3d point cloud processing software. `http://www.danielgm.net/cc/`. Accessed: 2016.12.09.

Golovinskiy, A., Kim, V., and Funkhouser, T. (2009). Shape-based recognition of 3D point clouds in urban environments. *International Conference on Computer Vision (ICCV)*, pages 2154–2161.

Gong, J., Zhu, Q., Zhong, R. F., Zhang, Y. T., and Xie, X. (2012). An Efficient Point Cloud Management Method Based on a 3D R-Tree. *Photogrammetric Engineering and Remote Sensing*, 78:373–381 ST – An Efficient Point Cloud Management.

Gorte, B. (2002). Segmentation of tin-structured surface models. In *Proceedings Joint International Symposium on Geospatial Theory, Processing and Applications, on CDROM*, page 5.

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

Gräfe, G. (2008). Kinematic 3D Laser Scanning for Road or Railway Construction Surveys. *1st International Conference on Machine Control & Guidance 2008*, pages 1–10.

Guan, H., Li, J., Yu, Y., Chapman, M., and Wang, C. (2015). Automated road information extraction from mobile laser scanning data. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):194–205.

Guan, H., Li, J., Yu, Y., Wang, C., Chapman, M., and Yang, B. (2014). Using mobile laser scanning data for automated extraction of road markings. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87:93–107.

Guo, L., Chehata, N., Mallet, C., and Boukir, S. (2011). Relevance of airborne lidar and multispectral image data for urban scene classification using Random Forests. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(1):56–66.

Haala, N. and Brenner, C. (1999). Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):130–137.

Haala, N., Peter, M., Kremer, J., and Hunter, G. (2008). Mobile Lidar Mapping for 3D Point Cloud Collection in Urban Areas: a Performance Test. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XXXVII-B5*, pages 1119–1124.

Haddad, N. A. (2011). From ground surveying to 3D laser scanner: A review of techniques used for spatial documentation of historic sites. *Journal of King Saud University - Engineering Sciences*, 23(2):109–118.

Halfawy, M. (2008). Integration of Municipal Infrastructure Asset Management Processes: Challenges and Solutions. *Journal of Computing in Civil Engineering*, 22(3):216–229.

Hatger, C. and Brenner, C. (2003). Extraction of Road Geometry Parameters From Laser Scanning and Existing Databases. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(Part 3/W13):225–230.

Henning, J. G. and Radtke, P. J. (2006). Detailed stem measurements of standing trees from ground-based scanning lidar. *Forest Science*, 52(1):67–80.

Hohenthal, J., Alho, P., Hyyppa, J., and Hyyppa, H. (2011). Laser scanning applications in fluvial studies. *Progress in Physical Geography*, 35:782–809.

Holmgren, J. and Persson, A. (2004). Identifying species of individual trees using airborne laser scanner. *Remote Sensing of Environment*, 90(4):415–423.

Holzer, S., Rusu, R. B., Dixon, M., Gedikli, S., and Navab, N. (2012). Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *IEEE International Conference on Intelligent Robots and Systems*, pages 2684–2689.

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

Horn, B. (1984). Extended Gaussian Images. *Proceedings of the IEEE*, 72(12):1671–1686.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

Hu, H., Ding, Y., Zhu, Q., Wu, B., Lin, H., Du, Z., Zhang, Y., and Zhang, Y. (2014). An adaptive surface filter for airborne laser scanning point clouds by means of regularization and bending energy. *ISPRS Journal of Photogrammetry and Remote Sensing*, 92:98–111.

Hung, R., King, B., and Chen, W. (2015). Conceptual Issues Regarding the Development of Underground Railway Laser Scanning Systems. *ISPRS International Journal of Geo-Information*, 4(1):185–198.

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

Hyyppä, J., Kelle, O., Lehikoinen, M., and Inkinen, M. (2001). A segmentation-based method to retrieve stem volume estimates from 3-D tree height models produced by laser scanners. *IEEE Transactions on Geoscience and Remote Sensing*, 39(5):969–975.

Idrees, M. O. and Pradhan, B. (2016). A decade of modern cave surveying with terrestrial laser scanning: A review of sensors, method and application development. *International Journal of Speleology*, 45:71–88.

Ivan, I., Benenson, I., Jiang, B., Horák, J., Haworth, J., and Inspektor, T. (2015). Geoinformatics for intelligent transportation. In *Lecture Notes in Geoinformation and Cartography*, volume 214, page 272.

Jaakkola, A., Hyyppä, J., Hyyppä, H., and Kukko, A. (2008). Retrieval Algorithms for Road Surface Modelling Using Laser-Based Mobile Mapping. *Sensors*, 8(9):5238–5249.

Jeong, J., Park, J., Kim, B., and Jeong, D. (2007). Automatic identification of road sign in mobile mapping system. In *roceedings of the International Symposium on Mobile Mapping Technology, Padua*, volume XXXVI-5/C55, Padua, Italy.

Jung, D. and Gupta, K. K. (1997). Octree-based hierarchical distance maps for collision detection. *Journal of Robotic Systems*, 14(11):789–806.

Kang, X., Liu, J., and Lin, X. (2014). Streaming progressive TIN densification filter for airborne LiDAR point clouds using multi-core architectures. *Remote Sensing*, 6(8):7212–7232.

Karolina, D., Ian, J., James, M., Robert, J., Jeffrey, P., and Jorg, M. (2013). Analysis of full-waveform lidar data for classification of an orange orchard scene. *{ISPRS} Journal of Photogrammetry and Remote Sensing*, 82:63 – 82.

Kazuhiro, A., John, S., and S., M. E. (2005). Forest road design with soil sediment evaluation using a high-resolution DEM. *Journal of Forest Research*, 10(6):471–479.

Kim, J.-S., Lee, J.-C., Kang, I.-J., Cha, S.-Y., Choi, H., and Lee, T.-G. (2008). Extraction of geometric information on highway using terrestrial laser scanning technology. In *ISPRS Congress*, pages 539–544.

King, A. D. (1998). Inertial navigation - Forty years of evolution. *Gec Review*, 13(3):140–+.

Kirkpatrick, D. and Seidel, R. (1983). On the Shape of a Set of Points in the Plane. *IEEE Transactions on Information Theory*, 29(4):551–559.

Klasing, K., Althoff, D., Wollherr, D., and Buss, M. (2009). Comparison of surface normal estimation methods for range sensing applications. *IEEE International Conference on Robotics and Automation*, pages 3206–3211.

Kraus, K. and Pfeifer, N. (1998). Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(4):193–203.

Kraus, K. and Pfeifer, N. (2001). Advanced dtm generation from lidar data. *International Archives Of Photogrammetry Remote Sensing And Spatial Information Sciences*, 34(3/W4):23–30.

Kukko, A., Jaakkola, A., Lehtomäki, M., Kaartinen, H., and Yuwei, C. (2009). Mobile mapping system and computing methods for modelling of road environment. In *2009 Joint Urban Remote Sensing Event*.

Kukko, A., Kaartinen, H., Hyyppä, J., and Chen, Y. (2012). Multiplatform mobile laser scanning: Usability and performance.

Kumar, P., McElhinney, C. P., Lewis, P., and McCarthy, T. (2014). Automated road markings extraction from mobile laser scanning data. *International Journal of Applied Earth Observation and Geoinformation*, 32:125–137.

Kwon, S. W., Bosche, F., Kim, C., Haas, C. T., and Liapi, K. A. (2004). Fitting range data to primitives for rapid local 3D modeling using sparse range point clouds. In *Automation in Construction*, volume 13, pages 67–81.

Lahivaara, T., Seppanen, A., Kaipio, J., Vauhkonen, J., Korhonen, L., Tokola, T., and Maltamo, M. (2014). Bayesian approach to tree detection based on airborne laser scanning data. *IEEE Transactions on Geoscience and Remote Sensing*, 52(5):2690–2699.

Lalonde, J.-F., Vandapel, N., and Hebert , M. (2007). Data structures for efficient dynamic processing in 3-d. *The International Journal of Robotics Research*, 26(8):777–796.

Lastra, M. and Revelles, J. (2000). An Efficient Parametric Algorithm for Octree Traversal. *J. of WSCG (2000)*, pages 212–219.

Lee, H. S. and Younan, N. H. (2003). DTM extraction of lidar returns via adaptive processing. *IEEE Transactions on Geoscience and Remote Sensing*, 41(9 PART I):2063–2069.

Lee H., Y. H. S. (2009). Marching-cube-and-octree-based level-of-detail modelling of 3d objects. *International Journal of Modelling and Simulation*, 29(2):121–126.

Lehtomäki, M., Jaakkola, A., Hyypp??, J., Kukko, A., and Kaartinen, H. (2010a). Detection of vertical pole-like objects in a road environment using vehicle-based laser scanning data. *Remote Sensing*, 2(3):641–664.

Lehtomäki, M., Jaakkola, A., Hyyppä, J., Kukko, A., and Kaartinen, H. (2010b). Detection of vertical pole-like objects in a road environment using vehicle-based laser scanning data. *Remote Sensing*, 2(3):641–664.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., and Thrun, S. (2011). Towards fully autonomous driving: Systems and algorithms. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 163–168.

Li, B., Schnabel, R., Klein, R., Cheng, Z., Dang, G., and Jin, S. (2010a). Robust normal estimation for point clouds with sharp features. *Computers and Graphics (Pergamon)*, 34(2):94–106.

Li, F., Tang, R., Liu, C., and Yu, H. (2010b). A method for object reconstruction based on point-cloud data via 3D scanning. *2010 International Conference on Audio, Language and Image Processing*, pages 302–306.

Li, Q., Zheng, N., and Cheng, H. (2004). Springrobot: a prototype autonomous vehicle and its algorithms for lane detection. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):300–308.

Li, D.and Oude Elberink, S. (2013). Optimizing detection of road furniture (pole-like object) in mobile laser scanner data. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume II-5/W2, pages 55–60, Antalya, Turkey. ISPRS Workshop Laser Scanning 2013.

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

Li, Y., Wu, H., Xu, H., An, R., Xu, J., and He, Q. (2013). A gradient-constrained morphological filtering algorithm for airborne LiDAR. *Optics and Laser Technology*, 54:288–296.

Liang, X. and Hyyppä, J. (2013). Automatic stem mapping by merging several terrestrial laser scans at the feature and decision levels. *Sensors*, 13:1614–1634.

Liang, X., Kukko, A., Kaartinen, H., Hyyppä, J., Yu, X., Jaakkola, A., and Wang, Y. (2014). Possibilities of a personal laser scanning system for forest mapping and ecosystem services. *Sensors (Switzerland)*, 14(1):1228–1248.

Liebowitz, J. (2002). A look at NASA Goddard Space Flight Center's knowledge management initiatives. *IEEE Software*, 19(3):40 –42.

Lin, C., Chen, J., Su, P., and Chen, C. (2014). Eigen-feature analysis of weighted covariance matrices for lidar point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 94:70 – 79.

Lindberg, E., Eysn, L., Hollaus, M., Holmgren, J., and Pfeifer, N. (2014). Delineation of tree crowns and tree species classification from full-waveform airborne laser scanning data using 3-d ellipsoidal clustering. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(7):3174–3181.

Lindenbergh, R., Berthold, D., Sirmacek, B., Herrero-Huerta, M., Wang, J., and Ebersbach, D. (2015). Automated large scale parameter extraction of road-side trees sampled by a laser mobile mapping system. In *Remote Sensing and Spatial Information Sciences*, number XL-3 in International Archives of the Photogrammetry, pages 589–594, La Grande Motte, France. ISPRS.

Lindstrom, P. (2000). Out-of-core Simplification of Large polygonal models. In *Proceedings of SIGGRAPH 2000*, volume 34, pages 259–262.

Liu, H., Wang, L., and Jezek, K. C. (2006). Automated delineation of dry and melt snow zones in Antarctica using active and passive microwave observations from space. *IEEE Transactions on Geoscience and Remote Sensing*, 44(8):2152–2163.

Losasso, F., Gibou, F., and Fedkiw, R. (2004). Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics*, 23(3):457.

Lu, X., Guo, Q., Li, W., and Flanagan, J. (2014). A bottom-up approach to segment individual deciduous trees using leaf-off lidar point cloud data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 94:1–12.

Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B., and Huebner, R. (2003). *Level of Detail for 3D Graphics*.

Luo, H., Wang, C., Wen, C., Cai, Z., Chen, Z., Wang, H., Yu, Y., and Li, J. (2016). Patch-Based Semantic Labeling of Road Scene Using Colorized Mobile LiDAR Point Clouds. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1286–1297.

Ma, R. (2005). DEM Generation and Building Detection from Lidar Data. *Photogrammetric Engineering & Remote Sensing*, 71(7):847–854.

Maas, H. G. and Vosselman, G. (1999). Two algorithms for extracting building models from raw laser altimetry data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):153–163.

Major, F., Malenfant, J., and Stewart, N. F. (1989). Distance between objects represented by octtrees defined in different coordinate systems. *Computers and Graphics*, 13(4):497–503.

Mallet, C., Bretar, F., Roux, M., Soergel, U., and Heipke, C. (2011). Relevance assessment of full-waveform lidar data for urban area classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6 SUPPL.).

Mancini, A., Frontoni, E., and Zingaretti, P. (2012). Automatic road object extraction from Mobile Mapping Systems. In *Proceedings of 2012 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, MESA 2012*, pages 281–286.

May, S., Droeschel, D., Holz, D., Augustin, S., and Fuchs, S. (2008). 3D Pose Estimation and Mapping with Time-of-Flight Cameras. *Current*, I(September 2007):2008–2008.

Mc Elhinney, C., Kumar, P., Cahalane, C., and McCarthy, T. (2010). Initial results from European Road Safety Inspection (EURSI) mobile mapping project. In *ISPRS Commission V Technical Symposium*, volume 2007, pages 440–445.

McDaniel, M., Nishihata, T., Brooks, C., Salesses, P., and Iagnemma, K. (2012). Terrain classification and identification of tree stems using ground-based LiDAR. *Journal of Field Robotics*, 29(6):891–910.

Mcguire, M. and Mara, M. (2014). Efficient GPU Screen-Space Ray Tracing. *Journal of Computer Graphics Techniques*, 3(4):73–85.

McNeff, J. G. (2002). The global positioning system. *IEEE Transactions on Microwave Theory and Techniques*, 50(3):645–652.

Meagher, D. (1982). Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147.

Menenti, M. and Ritchie, J. C. (1994). Estimation of effective aerodynamic roughness of Walnut Gulch watershed with laser altimeter measurements. *Water Resources Research*, 30(5):1329–1337.

Meng, X., Currit, N., and Zhao, K. (2010). Ground filtering algorithms for airborne LiDAR data: A review of critical issues. *Remote Sensing*, 2(3):833–860.

Miraliakbari, A., Hahn, M., and Sok, S. (2015). Automatic Extraction of Road Surface and Curbstone Edges From Mobile Laser Scanning Data. In *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XL-4/W5, pages 119–124.

Mitra, N. J., Nguyen, A., and Guibas, L. (2004). Estimating Surface Normals in Noisy Point Cloud Data. *International Journal of Computational Geometry and Applications*, 14(4-5):261–276.

Mongus, D., Luka??, N., and ??alik, B. (2014). Ground and building extraction from LiDAR data based on differential morphological profiles and locally fitted surfaces. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:145–156.

Mongus, D. and Zalik, B. (2012). Parameter-free ground filtering of LiDAR data for automatic DTM generation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67(1):1–12.

Monserrat, O. and Crosetto, M. (2008). Deformation measurement using terrestrial laser scanning data and least squares 3D surface matching. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):142–154.

Morsdorf, F., Kötz, B., Meier, E., Itten, K. I., and Allgöwer, B. (2006). Estimation of LAI and fractional cover from small footprint airborne laser scanning data based on gap fraction. *Remote Sensing of Environment*, 104(1):50–61.

Moskal, L. and Zheng, G. (2012). Retrieving forest inventory variables with terrestrial laser scanning (TLS) in urban heterogeneous forest. *Remote Sensing*, 4(1):1–20.

Mount, D. and Arya, S. (2010). ANN - a library for approximate nearest neighbor searching. `http://www.cs.umd.edu/~mount/ANN/`. Accessed: 2016.10.10.

Muja, M. (2011). FLANN - fast library for approximate nearest neighbors. `http://www.cs.ubc.ca/research/flann/`. Accessed: 2016.10.10.

Muja, M. and Lowe, D. (2014). nanoflann - a C++ header-only library for nearest neighbor (nn) search with kd-trees. `https://github.com/efernandez/nanoflann`. Accessed: 2016.10.10.

Nahangi, M., Czerniawski, T., Rausch, C., and Haas, C. (2016). Arbitrary 3d object extraction from cluttered laser scans using local features. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 33, page 1. Vilnius Gediminas Technical University, Department of Construction Economics & Property.

Nebiker, S., Bleisch, S., and Christen, M. (2010). Rich point clouds in virtual globes - A new paradigm in city modeling? *Computers, Environment and Urban Systems*, 34(6):508–517.

Nelson, R., Krabill, W., and MacLean, G. (1984). Determining forest canopy characteristics using airborne laser data. *Remote Sensing of Environment*, 15(3):201–212.

Nuechter, A., Lingemann, K., and Borrmann, D. (2013). Point cloud library documentation: Down sampling a point cloud using a voxel grid filter. `http://pointclouds.org/documentation/tutorials/voxel_grid.php#voxelgrid`. Accessed: 2013.05.06.

Nuechter, A., Lingemann, K., and Borrmann, D. (2016). 3DTK - the 3d toolkit. `http://slam6d.sourceforge.net/index.html`. Accessed: 2016.10.10.

O'Callaghan, J. F. and Mark, D. M. (1984). The extraction of drainage networks from digital elevation data. *Computer Vision, Graphics, and Image Processing*, 28(3):323 – 344.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N. (2009). *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons.

Otepka, J., Ghuffar, S., Waldhauser, C., Hochreiter, R., and Pfeifer, N. (2013). Georeferenced Point Clouds: A Survey of Features and Point Cloud Management. *ISPRS International Journal of Geo-Information*, 2:1038–1065.

Over, M., Schilling, A., Neubauer, S., and Zipf, A. (2010). Generating web-based 3D City Models from OpenStreetMap: The current situation in Germany. *Computers, Environment and Urban Systems*, 34(6):496–507.

Palha, A., Murtiyoso, A., Michelin, J.-C., Alby, E., and Grussenmeyer, P. (2017). *Open Source First Person View 3D Point Cloud Visualizer for Large Data Sets*, pages 27–39. Springer International Publishing, Cham.

Papadias, D. and Theodoridis, Y. (1997). Spatial Relations, Minimum Bounding Rectangles, and Spatial Data Structures. *International Journal of Geographic Information Science*, 11:111–138.

Papon, J., Abramov, A., Schoeler, M., and Worgotter, F. (2013). Voxel cloud connectivity segmentation - Supervoxels for point clouds. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2027–2034.

Paul Chew, L. (1989). Constrained delaunay triangulations. *Algorithmica*, 4(1-4):97–108.

Pauling, F., Bosse, M., and Zlot, R. (2009). Automatic segmentation of 3d laser point clouds by ellipsoidal region growing. In Scheding, S., editor, *Australasian Conference on Robotics and Automation 2009 (ACRA 09). Proceedings*, pages 11–20, Sydney, NSW, Australia. Curran Associates.

Payeur, P. (2006). A computational technique for free space localization in 3-D multiresolution probabilistic environment models. *IEEE Transactions on Instrumentation and Measurement*, 55:1734–1746.

Persson, A., Holmgren, J., and Söderman, U. (2002). Detecting and measuring individual trees using an airborne laser scanner. *Photogrammetric Engineering & Remote Sensing*, 68(9):925–932.

Petitjean, S. (2002). A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 34(2):211–262.

Peucker, T. K., Fowler, R. J., Little, J. J., and Mark, D. M. (1978). The triangulated irregular network. In *Amer. Soc. Photogrammetry Proc. Digital Terrain Models Symposium*, volume 516, page 532.

Pfeifer, N. (2001). Derivation Of Digital Terrain Models In The Scop++ Environment. *OEEPE Workshop on Airborne Laserscanning and Interferometric SAR for Digital Elevation Models*, ,, page 13.

Poppenga, S. K., Worstell, B. B., Stoker, J. M., and Greenlee, S. K. (2010). Using selective drainage methods to extract continuous surface flow from 1-meter lidar-derived digital elevation data. Technical report, U.S. Geological Survey Scientific Investigations Report.

Preparata, F. and Shamos, M. (1985). *Computational geometry: an introduction*. Springer-Verlag New York.

Pu, S., Rutzinger, M., Vosselman, G., and Oude Elberink, S. (2011). Recognizing basic structures from mobile laser scanning data for road inventory studies. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6 SUPPL.).

Pu, S. and Vosselman, G. (2006). Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry*, 36:25–27.

Pu, S. and Vosselman, G. (2009). Knowledge based reconstruction of building models from terrestrial laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6):575–584.

Pu, S. and Zhan, Q. (2009). Classification of mobile terrestrial laser point clouds using semantic constraints. In *SPIE Optical Engineering and Applications*, pages 74470D1–74470D9.

Puente, I., González-Jorge, H., Martínez-Sánchez, J., and Arias, P. (2013a). Review of mobile mapping and surveying technologies.

Puente, I., González-Jorge, H., Riveiro, B., and Arias, P. (2013b). Accuracy verification of the Lynx Mobile Mapper system. *Optics and Laser Technology*, 45(1):578–586.

Puttonen, E., Jaakkola, A., Litkey, P., and Hyyppä, J. (2011). Tree classification with fused mobile laser scanning and hyperspectral data. *Sensors*, 11(5):5158–5182.

Qin, R. and Gruen, A. (2014). 3D change detection at street level using mobile laser scanning point clouds and terrestrial images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 90:23–35.

Rabbani, T. and Heuvel, F. V. D. (2005). Efficient Hough Transform for Automatic Detection of Cylinders in Point Clouds. *ISPRS Workshop on Laser Scanning*, 3:60–65.

Rahman, M., Gorte, B., and Bucksch, A. (2009). A new method for individual tree delineation and undergrowth removal from high resolution airborne lidar. In Gorge, V., editor, *Proceedings ISPRS Workshop Laserscanning 2009*, volume XXXVIII (3/W8) of *Laser scanning 2009*, Paris, France. ISPRS.

Razak, K., Abu Bakar, R., Wah, Q., and Wan Mohd Akib, W. (2011). Geodetic laser scanning technique for characterizing landslides along high-risk road zone: Applications and limitations. In *In Proceedings of the FIG Working Week*, Marrakech, Morocco.

Remondino, F. (2003). From point cloud to surface: the modeling and visualization problem. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV:24–28.

Rhayma, N., Bressolette, P., Breul, P., Fogli, M., and Saussine, G. (2013). Reliability analysis of maintenance operations for railway tracks. *Reliability Engineering and System Safety*, 114(1):12–25.

Richter, R., Discher, S., and Döllner, J. (2015). Out-of-Core Visualization of Classified 3D Point Clouds. *3D Geoinformation Science: The Selected Papers of the 3D GeoInfo 2014*, pages 227–242.

Richter, R. and Döllner, J. (2014). Concepts and techniques for integration, analysis and visualization of massive 3D point clouds. *Computers, Environment and Urban Systems*, 45:114–124.

Rodríguez-Cuenca, B., García-Cortés, S., Ordóñez, C., and Alonso, M. (2015). Automatic Detection and Classification of Pole-Like Objects in Urban Point Cloud Data Using an Anomaly DetectionAlgorithm. *Remote Sensing*, 7(10):12680–12703.

Rusu, R., Blodow, N., and Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217.

Rusu, R. B. (2010). Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. *KI - K??nstliche Intelligenz*, pages 1–4.

Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. (2008). Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941.

Rutzinger, M., Höfle, B., Hollaus, M., and Pfeifer, N. (2008). Object-based point cloud analysis of full-waveform airborne laser scanning data for urban vegetation classification. *Sensors*, 8(8):4505–4528.

Rutzinger, M., Pratihast, A. K., Elberink, O., and Vosselman, G. (2010). Detection and Modelling of 3D Trees From Mobile Laser Scanning Data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(5):520–525.

Rutzinger, M., Pratihast, A. K., Oude Elberink, S. J., and Vosselman, G. (2011). Tree modelling from mobile laser scanning data-sets. *Photogrammetric Record*, 26(135):361–372.

Rutzinger, M., Rottensteiner, F., and Pfeifer, N. (2009). A Comparison of Evaluation Techniques for Building Extraction From Airborne Laser Scanning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2(1):11–20.

Ryding, J., Williams, E., Smith, M. J., and Eichhorn, M. P. (2015). Assessing handheld mobile laser scanners for forest surveys. *Remote Sensing*, 7(1):1095–1111.

Salvini, R., Francioni, M., Riccucci, S., Bonciani, F., and Callegari, I. (2013). Photogrammetry and laser scanning for analyzing slope stability and rock fall runout along the Domodossola-Iselle railway, the Italian Alps. *Geomorphology*, 185:110–122.

Samet, H. (1995). Spatial data structures. *Modern Database Systems: The Object Model, Interoperability, and Beyond*, pages 361–385.

Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Sampath, R. S. and Biros, G. (2010). A Parallel Geometric Multigrid Method for Finite Elements on Octree Meshes. *SIAM Journal on Scientific Computing*, 32(3):1361–1392.

Sankaranarayanan, J., Samet, H., and Varshney, A. (2007). A fast all nearest neighbor algorithm for applications involving large point-clouds. *Computers and Graphics (Pergamon)*, 31(2):157–174.

Schawlow, a. L. and Townes, C. H. (1958). Infrared and Optical Masers. *Physical Review*, 112(6):1940–1949.

Scheiblauer, C. and Wimmer, M. (2011). Out-of-core selection and editing of huge point clouds. *Computers and Graphics (Pergamon)*, 35(2):342–351.

Scheier, H., Lougheed, J., Tuchker, C., and Leckire, D. (1985). Automated measurements of terrain reflection and height variations using an airborne infrared laser system. *International Journal of Remote Sensing*, 6(1):101–113.

Schnabel, R. and Klein, R. (2006). Octree-based point-cloud compression. In Botsch, M. and Chen, B., editors, *Symposium on Point-Based Graphics 2006*. Eurographics.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226.

Schreiber, M., Knöpel, C., and Franke, U. (2013). Laneloc: Lane marking based localization using highly accurate maps. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 449–454.

Sch"utz, M. (2016). Potree: Rendering large point clouds in web browsers. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

Sérgio, R., Madeira, L., Bastos, A., Sousa, J., and Luís, P. (2005). Automatic traffic signs inventory using a mobile mapping system. In *In Proceedings of the International Conference and Exhibition on Geographic Information.*

Serna, A. and Marcotegui, B. (2013). Urban accessibility diagnosis from mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 84:23–32.

Shaffer, C. A. (1998). *A Practical Introduction to Data Structures and Algorithms Analysis, Java Edition.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Sirmacek, B. and Lindenbergh, R. (2015). Automatic classification of trees from laser scanning point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W5:137–144.

Sithole, G. (2001). FILTERING OF LASER ALTIMETRY DATA USING A SLOPE ADAPTIVE FILTER. *International Archives of Photogrammetry and Remote Sensing*, 34(3(WG4)):203–210.

Sithole, G. and Vosselman, G. (2004). Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59:85–101.

Slattery, K. T., Slattery, D. K., and Peterson, J. P. (2012). Road Construction Earthwork Volume Calculation Using Three-Dimensional Laser Scanning. *Journal of Surveying Engineering*, 138(2):96–99.

Solberg, S., Naesset, E., and Bollandsas, O. (2006). Single tree segmentation using airborne laser scanner data in a structurally heterogeneous spruce forest. *Photogrammetric Engineering & Remote Sensing*, 72(12):1369–1378.

Soudarissanane, S., Lindenbergh, R., Menenti, M., and Teunissen, P. (2011). Scanning geometry: Influencing factor on the quality of terrestrial laser scanning points. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(4):389–399.

Strom, J., Richardson, A., and Olson, E. (2010). Graph-based segmentation for colored 3D laser point clouds. In *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 2131–2136.

Su, Y.-T., Bethel, J., and Hu, S. (2016). Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 113:59–74.

Tang, M., Manocha, D., and Tong, R. (2010). MCCD: Multi-core collision detection between deformable models using front-based decomposition. *Graphical Models*, 72(2):7–23.

Tang, S., Dong, P., and Buckles, B. (2013). Three-dimensional surface reconstruction of tree canopy from lidar point clouds using a region-based level set method. *International Journal of Remote Sensing*, 34(4):1373–1385.

Tao, C. V. (2000). Mobile mapping technology for road network data acquisition. *Journal of Geospatial Engineering*, 2(2):1–14.

Tarolli, P., Calligaro, S., Cazorzi, F., and Dalla Fontana, G. (2013). Recognition of surface flow processes influenced by roads and trails in mountain areas using high-resolution topography. *European Journal of Remote Sensing*, 46(1):176–197.

Tarsha-Kurdi, F., Landes, T., and Grussenmeyer, P. (2007). Hough-Transform and Extended Ransac Algorithms for Automatic Detection of 3D Building Roof Planes From Lidar Data. *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, XXXVI(1):407–412.

Teo, T. A. and Chiu, C. M. (2015). Pole-Like Road Object Detection from Mobile Lidar System Using a Coarse-to-Fine Approach. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(10):4805–4818.

Teo, T.-A., Rau, J.-Y., Chen, L.-C., Liu, J.-K., and Hsu, W.-C. (2006). Reconstruction of Complex Buildings using LIDAR and 2D Maps. *Innovations in 3D Geo Information Systems*, pages 345–354.

Thrun, S., Burgard, W., and Fox, D. (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 1(April):321–328 vol.1.

Thürmer, G. and Wüthrich, C. a. (1997). Normal Computation for Discrete Surfaces in 3D Space. *Computer Graphics Forum*, 16(3).

Tovari, D. and Pfeifer, N. (2005). Segmentation based robust interpolation – a new approach to laser data filtering. In *Laserscanning 2005*, volume IAPRS Vol, page 6.

Tse, R., Gold, C., and Kidner, D. (2007). Using the delaunay triangulation/ voronoi diagram to extract building information from raw lidar data. In *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, pages 222–229.

Turner, A. (2007). From axial to road-centre lines: A new representation for space syntax and a new model of route choice for transport network analysis. *Environment and Planning B: Planning and Design*, 34(3):539–555.

Ummenhofer, B. and Brox, T. (2013). Point-based 3D reconstruction of thin objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 969–976.

Vaaja, M., Hyyppä, J., Kukko, A., Kaartinen, H., Hyyppä, H., and Alho, P. (2011). Mapping topography changes and elevation accuracies using a mobile laser scanner. *Remote Sensing*, 3(3):587–600.

van der Sande, C., Soudarissanane, S., and Khoshelham, K. (2010). Assessment of relative accuracy of ahn-2 laser scanning data using planar features. *Sensors*, 10(9):8198.

Van Deusen, P. (2010). Carbon sequestration potential of forest land: Management for products and bioenergy versus preservation. *Biomass and Bioenergy*, 34(12):1689–1694.

Van Gosliga, R., Lindenbergh, R., and Pfeifer, N. (2006). Deformation analysis of a bored tunnel by means of terrestrial laser scanning. *Proceedings of the ISPRS Commission V Symposium on Image Engineering and Vision Metrology*, XXXVI, Par:167–172.

Vanier, D. (2006). Towards sustainable municipal infrastructure asset management. *Handbook on Urban Sustainability*, 12(1):283–314.

Vaughn, N., Moskal, L., and Turnblom, E. (2012). Tree species detection accuracies using discrete point lidar and airborne waveform lidar. *Remote Sensing*, 4(2):377–403.

Vega, C., Hamrouni, A., El Mokhtari, S., Morel, J., Bock, J., Renaud, J., Bouvier, M., and Durrieu, S. (2014). PTrees: A point-based approach to forest tree extraction from lidar data. *International Journal of Applied Earth Observation and Geoinformation*, 33:98–108.

Velizhev, A., Shapovalov, R., and Schindler, K. (2012). Implicit Shape Models for Object Detection in 3D Point Clouds. In *XXII ISPRS Congress, Technical Commission III*, volume I-3, pages 179–184.

Vieira, M. and Shimada, K. (2005). Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Design*, 22(8):771–792.

Vo, A. V., Truong-Hong, L., Laefer, D. F., and Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:88–100.

Vosselman, G. (2000). Slope based filtering of laser altimetry data. In *IAPRS 2000*, pages 1–8.

Vosselman, G. and Dijkman, S. (2001). 3D building model reconstruction from point clouds and ground plans. *Int. Arch. of Photogrammetry and Remote Sensing*, XXXIV:37–43.

Vosselman, G., Gorte, B., Sithole, G., and Rabbani, T. (2004). Recognising Structure in Laser Scanner Point Clouds. *Information Sciences*, 46(April 2016):1–6.

Vosselman, G. and Maas, H.-G. (2010). *Airborne and Terrestrial Laser Scanning*. Whittles publishing.

Wald, I. and Havran, V. (2007). On building fast kd-trees for ray tracing, and on doing that in O(N log N). In *RT'06: IEEE Symposium on Interactive Ray Tracing 2006, Proceedings*, pages 61–69.

Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., and Schilling, A. (2007). Interactive Editing of Large Point Clouds. *Eurographics Symposium on Point-Based Graphics*, pages 37–45.

Wand, M., Berner, A., Bokeloh, M., Jenke, P., Fleck, A., Hoffmann, M., Maier, B., Staneker, D., Schilling, A., and Seidel, H. P. (2008). Processing and interactive editing of huge point clouds from 3D scanners. *Computers and Graphics (Pergamon)*, 32(2):204–220.

Wang, J., González-Jorge, H., Lindenbergh, R., Arias-Sánchez, P., and Menenti, M. (2013). Automatic Estimation of Excavation Volume from Laser Mobile Mapping Data for Mountain Road Widening. *Remote Sensing*, 5(9):4629–4651.

Wang, J., González-Jorge, H., Lindenbergh, R., Arias-Sánchez, P., and Menenti, M. (2014). Geometric road runoff estimation from laser mobile mapping data. In *ISPRS Annals of the Photogrammetry*, volume II-5, pages 385–391, Riva del Garda, Italy.

Wang, J., Lindenbergh, R., Shen, Y., and Menenti, M. (2016). Coarse point cloud registration by egi matching of voxel clusters. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume III-5, pages 98–103, Prague, Czech Republic. XXIII ISPRS Congress.

Wang, M. and Tseng, Y. (2004). Lidar data segmentation and classification based on octree structure. *Parameters*, 2(1):1–6.

Wang, W., Wang, J., and Sun, G. (2009). Noise reduction and modeling methods of tls point cloud based on r-tree. In *2009 Joint Urban Remote Sensing Event*, pages 1–5.

Wang, Y., Weinacker, H., and Koch, B. (2008). A Lidar point cloud based procedure for vertical canopy structure analysis and 3D single tree modelling in forest. *Sensors*, 8:3938–3951.

Wang, Z., Zhang, L., Fang, T., Mathiopoulos, P. T., Tong, X., Qu, H., Xiao, Z., Li, F., and Chen, D. (2015). A multiscale and hierarchical feature extraction method for terrestrial laser scanning point cloud classification. *IEEE Transactions on Geoscience and Remote Sensing*, 53(5):2409–2425.

Weber, J. and Penn, J. (1995). Creation and rendering of realistic trees. *SIGGRAPH '95 - Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 119–128.

Wehr, A. and Lohr, U. (1999). Airborne laser scanning—an introduction and overview. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):68–82.

White, R. A., Dietterick, B. C., Mastin, T., and Strohman, R. (2010). Forest roads mapped using lidar in steep forested terrain. *Remote Sensing*, 2(4):1120–1141.

Winker, D. M., Couch, R. H., and Mccormick, M. P. (1996). An overview of LITE: NASA's lidar in-space technology experiment. *Proceedings of the IEEE*, 84(2):164–180.

Woo, H., Kang, E., Wang, S., and Lee, K. H. (2002). A new segmentation method for point cloud data. *International Journal of Machine Tools and Manufacture*, 42(2):167–178.

Wu, B., Yu, B., Yue, W., Shu, S., W., T., Hu, C., Huang, Y., Wu, J., and Liu, H. (2013). A voxel-based method for automated identification and morphological parameters estimation of individual street trees from mobile laser scanning data. *Remote Sensing*, 5(2):584.

Wu, H., Guan, X., and Gong, J. (2011). ParaStream: A parallel streaming Delaunay triangulation algorithm for LiDAR points on multicore architectures. *Computers and Geosciences*, 37(9):1355–1363.

Wu, S., Jiang, D., Ooi, B., and Wu, K. (2010). Efficient b-tree based indexing for cloud data processing. *Proceedings of the VLDB Endowment*, pages 1207–1218.

Xiao, W., Vallet, B., Brédif, M., and Paparoditis, N. (2015). Street environment change detection from mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 107:38–49.

Xiao, W., Vallet, B., Schindler, K., and Paparoditis, N. (2016). Street-side vehicle detection, classification and change detection using mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:166 – 178.

Xiaoli, S., Abshire, J. B., McGarry, J. F., Neumann, G. A., Smith, J. C., Cavanaugh, J. F., Smith, D. J., Zwally, H. J., Smith, D. E., and Zuber, M. T. (2013). Space Lidar Developed at the NASA Goddard Space Flight Center—The First 20 Years. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 6(3):1660–1675.

Xiaoye Liu (2008). Airborne LiDAR for DEM generation: some critical issues. *Progress in Physical Geography*, 32(1):31–49.

Xiong, B., Oude Elberink, S., and Vosselman, G. (2014). A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:227–242.

Xu, H. and Barbič, J. (2014). Continuous Collision Detection Between Points and Signed Distance Fields. *Virtual Reality Interaction and Physical Simulation*.

Yan, L., Liu, H., Tan, J., Li, Z., Xie, H., and Chen, C. (2016). Scan line based road marking extraction from mobile LiDAR point clouds. *Sensors (Switzerland)*, 16(6).

Yang, B. and Dong, Z. (2013). A shape-based segmentation method for mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 81:19–30.

Yang, B., Dong, Z., Zhao, G., and Dai, W. (2015). Hierarchical extraction of urban objects from mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 99:45–57.

Yang, B. and Fang, L. (2014). Automated extraction of 3-D railway tracks from mobile laser scanning point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(12):4750–4761.

Yang, B., Fang, L., and Li, J. (2013). Semi-automated extraction and delineation of 3D roads of street scene from mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 79:80–93.

Yang, B., Fang, L., Li, Q., and Li, J. (2012a). Automated Extraction of Road Markings from Mobile Lidar Point Clouds. *Photogrammetric Engineering and Remote Sensing*, 78(4):331–338.

Yang, B., Wei, Z., Li, Q., and Li, J. (2012b). Automated extraction of street-scene objects from mobile lidar point clouds.

Yang, B. S., Fang, L. N., Li, Q. Q., and Li, J. (2012c). Automated Extraction of Road Markings from Mobile Lidar Point Clouds. *Photogrammetric Engineering and Remote Sensing*, 78(4):331–338.

Yang, J. and Huang, X. (2014). A Hybrid Spatial Index for Massive Point Cloud Data Management and Visualization. *Transactions in GIS*, 18(S1):97–108.

Yoon, J. S., Sagong, M., Lee, J. S., and sung Lee, K. (2009). Feature extraction of a concrete tunnel liner from 3D laser scanning data. *NDT and E International*, 42(2):97–105.

Yoshimura, R., Date, H., Kanai, S., Honma, R., Oda, K., and Ikeda, T. (2016). Automatic registration of mls point clouds and sfm meshes of urban area. *Geo-spatial Information Science*, 19(3):171–181.

Yu, W., He, F., and Xi, P. (2010). A rapid 3D seed-filling algorithm based on scan slice. *Computers and Graphics (Pergamon)*, 34(4):449–459.

Yu, X., Hyyppä, J., Kaartinen, H., and Maltamo, M. (2004). Automatic detection of harvested trees and determination of forest growth using airborne laser scanning. *Remote Sensing of Environment*, 90(4):451–462.

Yu, Y., Li, J., Guan, H., Wang, C., and Yu, J. (2015). Semiautomated extraction of street light poles from mobile LiDAR point-clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 53(3):1374–1386.

Zhang, H., Li, J., Cheng, M., and Wang, C. (2016). Rapid Inspection of Pavement Markings Using Mobile Lidar Point Clouds. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B1(June):717–723.

Zhang, J. and Lin, X. (2013). Filtering airborne LiDAR data by embedding smoothness-constrained segmentation in progressive TIN densification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 81:44–59.

Zhang, K., Chen, S. C., Whitman, D., Shyu, M. L., Yan, J., and Zhang, C. (2003). A progressive morphological filter for removing nonground measurements from airborne LIDAR data. *IEEE Transactions on Geoscience and Remote Sensing*, 41(4 PART I):872–882.

Zhao, H. and Shibasaki, R. (2003). Reconstructing a textured CAD model of an urban environment using vehicle-borne laser range scanners and line cameras. *Machine Vision and Applications*, 14(1):35–41.

Zheng, G., Chen, J., Tian, Q., Ju, W., and Xia, X. (2007). Combining remote sensing imagery and forest age inventory for biomass mapping. *Journal of Environmental Management*, 85(3):616–623.

Zhong, R., Wei, J., Su, W., and Chen, Y. (2013). A method for extracting trees from vehicle-borne laser scanning data. *Mathematical and Computer Modelling*, 58(3-4):727–736.

Zhou, L. and Vosselman, G. (2012). Mapping curbstones in airborne and mobile laser scanning data. *International Journal of Applied Earth Observation and Geoinformation*, 18(1):293–304.

Zhu, L. and Hyyppa, J. (2014). The use of airborne and mobile laser scanning for modeling railway environments in 3D. *Remote Sensing*, 6(4):3075–3100.

Zia, M. Z., Stark, M., and Schindler, K. (2013). Explicit occlusion modeling for 3D object class representations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3326–3333.

Ziegler, A. D. and Giambelluca, T. W. (1997). Importance of rural roads as source areas for runoff in mountainous areas of northern Thailand. *Journal of Hydrology*, 196:204–229.

Zogg, H.-M. and Ingensand, H. (2008). Terrestrial Laser Scanning for Deformation Monitoring–Load Tests on the Felsenau Viaduct (CH). *International Archives of Photogrammetry and Remote Sensing*, 37:555–562.

Zwally, H. J., Schutz, B., Abdalati, W., Abshire, J., Bentley, C., Brenner, A., Bufton, J., Dezio, J., Hancock, D., Harding, D., Herring, T., Minster, B., Quinn, K., Palm, S., Spinhirne, J., and Thomas, R. (2002). ICESat's laser measurements of polar ice, atmosphere, ocean, and land. *Journal of Geodynamics*, 34(3-4):405–445.

Acknowledgements

Reflecting on the whole procedure of my PhD research, I would regard it as an extremely unique experience and adventure. It has been a procedure consisting of both pains and gains that shaped me and will guide my future career as well. Many people contributed to the accomplishment of this dissertation in various ways, for which I am sincerely grateful.

First of all, I want to express my sincere gratitude to Roderik Lindenbergh. He became my official daily supervisor and co-promotor since we first met on Thursday, Sept. 20th, 2012, introduced by Massimo. In the followed years, his help and support were always there for both academic and personal problems. For me, he is the first person that I am able to discuss with whenever I have new ideas. Also, his critical comments helped me to focus on my ideas and his constant encouragement inspired me to advance in difficult moments. To train me into an independent researcher, his consecutive dedication and commitment are like the torches that guided me out of moor in darkness, and I am grateful for that forever. He is the best daily supervisor and co-promotor that a PhD candidate could ever have at TU Delft. To me, he is an exceptional icon to admire.

I would like to thank my promotor Prof. Massimo Menenti. It was the two-hour interview and discussion with him in room B205 at IRSA in Beijing encouraged me to make up my mind to choose TU Delft for my further education. During the followed years, his support and encouragement in our progress meetings helped me to build up my confidence gradually. I like the way that he structures a journal paper and dissertation very much.

My colleagues in the section of Optical and Laser Remote Sensing (OLRS) also deserve special thanks as we share knowledge and help to solve scientific problems in both group meetings and daily office time. I would especially like to thank Ben Gorte, Junchao, Yerong, Beril Sirmacek, Monica Herrero Huerta, Yueqian and Kaixuan. It has been a pleasure to work with all of you. My gratitude goes to the Chinese colleagues in the department. They are Jiangjun Ran, Yu Sun, Ling Chang, Yanqing Hou. It was a great pleasure that we not only shared information, but also helped each other to solve scientific and technical problems.

I would like to express my sincere thanks to Martin Kodde from Regional Innovation Center at Fugro B.V. We started our cooperation since 2013 with the point clouds sampled by Drive-Map system. Later on, he asked Sjoerd Staats to drive the Drive-Map system to collect the point cloud data on TU Delft campus for our research. Special thanks goes to Sjoerd, it was a unique experience for me to involve in the whole workflow of driving the system and pre-processing the data.

The other members of the department also deserve special thanks as they help me to deal with many administrative problems. They are Lidwien de Jong, Debbie Rietdijk and Irma Zomerdijk.

I would like to give my sincere gratitude to all the committee members for reviewing this dissertation, as well as giving comments and attending my PhD defense.

I would like to gratefully thank Prof. Chuanrong Li, Prof. Lingli Tang and Prof. Mei Zhou from Academy of Opto-Electronics, Chinese Academy of Sciences, for their concern of my PhD progresses and many supports of my research. I still remember the encouraging words you told me when we met. Thanks you so much.

With this opportunity, I would like to thank the China Scholarship Council for providing the four-year funding for my PhD research. I would also like to thank IQmulus project to allow me to attend international conferences and congress.

Being far away from my family was never easy, but many friends at Delft and other places provided another family. I owe many thanks to Jingyu Zhang, Ming Li, Xin Li, Zhen Yang, Lin Xiao, Feifei Wang, Chenxiao Liu, Xiaowei Ouyang, Xueming Li, Yuling Li, Tian Xin, Tao Lü and Xiangming Liu. It was the happy moments shared with you that eliminated the loneliness and alleviated homesick inside of me. I am also grateful to my housemates at You Ruo Po 54, whose friendship created a home-like environment. They are Changgong Zhang, Qian Yu, Shijie Li, Huarong Zheng and Dong Liu.

I would give my special thanks to Wen Yang for her understanding and love. Life is never easy but she opened an amazing window for me. Talking with her is just like the fresh air come through that window and finally nourish my mind, which I always appreciated and enjoyed.

Last but not least, I would like to gratefully thank my family, notably my parents and siblings, for their everlasting support and unconditional love in the past years. I owe too much for your support, especially my dear parents. 特别感谢父亲母亲这么多年来的艰辛付出和支持。是你们教会我做人做事的根本，也是你们教会我不论身在何处也都要努力保持积极上进的态度。谢谢你们！

Jinhu Wang
Delft, July 2017

About the author

Jinhu Wang was born in Yan'an, Shaanxi, China, on 9 Dec. 1987. He became a student major in Geomatrics in China University of Geosciences (Beijing) in 2005. He obtained bachelor degree of engineering in 2009 and won the outstanding undergraduate thesis on Kalman filtering in integrated navigation. In the same year, he was admitted as a master student in signal and information processing at Academy of Opto-Electronics (AOE), Chinese Academy of Sciences. The three years of study at AOE was mainly focused on airborne small-footprint full-waveform LiDAR point cloud segmentation and classification, supervised by Prof. Chuanrong Li and Prof Mei Zhou. He obtained master degree of science in 2012. In September of the same year, he started to work as a PhD candidate in section of Optical and Laser Remote Sensing in department of Geoscience and Remote Sensing at Delft University of Technology. He focused on scalable information extraction from point cloud obtained by mobile laser scanning systems, supervised by Dr. Roderik Lindenbergh and Prof. Massimo Menenti.

List of Publications

Peer-reviewed journals

J. Wang, H. González- Jorge, R. Lindenbergh, P. Arias-Sánchez, M. Menenti. Autopmatic estimation of excavation volume from laser mobile mapping data for mountain road widening. *Remote Sensing*, **2013**, 5, 4629-4651. (Chapter 4)

J. Wang, R. Lindenbergh, M. Menenti. SigVox- A 3D feature matching algorithm for automatic street object recognition in mobile laser scanning point cloud. *ISPRS Journal of Photogrammetry and Remote Sensing*, **2017**, 128, 111-129. (Chapter 6)

J. Wang, R. Lindenbergh, M. Menenti. VoxTree - A scalable method for tree delineation in 3D laser scanning point clouds. under review. (Chapter 5)

Y. Shen, R. Lindenbergh, J. Wang. Change Analysis in Structural Laser Scanning Point Clouds: The Baseline Method. *Sensors*, **2017**, 17(1), 26.

Peer-reviewed Conference proceedings

B. Sirmacek, R. Lindenbergh, J. Wang. Qaulity assessment and comparison of smartphone and Leica C10 laser scanner based point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **2016**, XLI-B5, 581-586.

J. Wang, R. Lindenbergh, Y. Shen, M. Menenti. Coarse point cloud registration by EGI matching of voxel clusters . *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **2016**, III-5, 97-103. [Best Poster Award]

J. Wang, R. Lindenbergh, M. Menenti. Evaluating voxel enabled scalable intersection of large point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **2015**, , ISPRS Geospatial Week 2015, 28 Sep – 03 Oct 2015, La Grande Motte, France, XL-3/W3, 25-31.

B. Gorte, S. Oude Elberink, B. Sirmacek, J. Wang. IQPC 2015 Track: Tree separation and classification in mobile mapping LiDAR data. *The International Archives of the Photogrammetry*, **2015**, , ISPRS Geospatial Week 2015, 28 Sep – 03 Oct 2015, La Grande Motte, France, XL-3/W3, 607-612.

R. Lindenbergh, D. Berthold, B. Sirmacek, M. Herrero-Huerta, J. Wang, D. Ebersbach. Automated large scale parameter extraction of road-side trees sampled by a laser mobile mapping system. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **2015**, , ISPRS Geospatial Week 2015, 28 Sep – 03 Oct 2015, La Grande Motte, France, XL-3/W3, 589-594.

J. Wang, H. González- Jorge, R. Lindenbergh, P. Arias-Sánchez, M. Menenti. Geometric road runoff estimation from laser mobile mapping data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **2014**, II-5, 385-391. (Chapter 4)