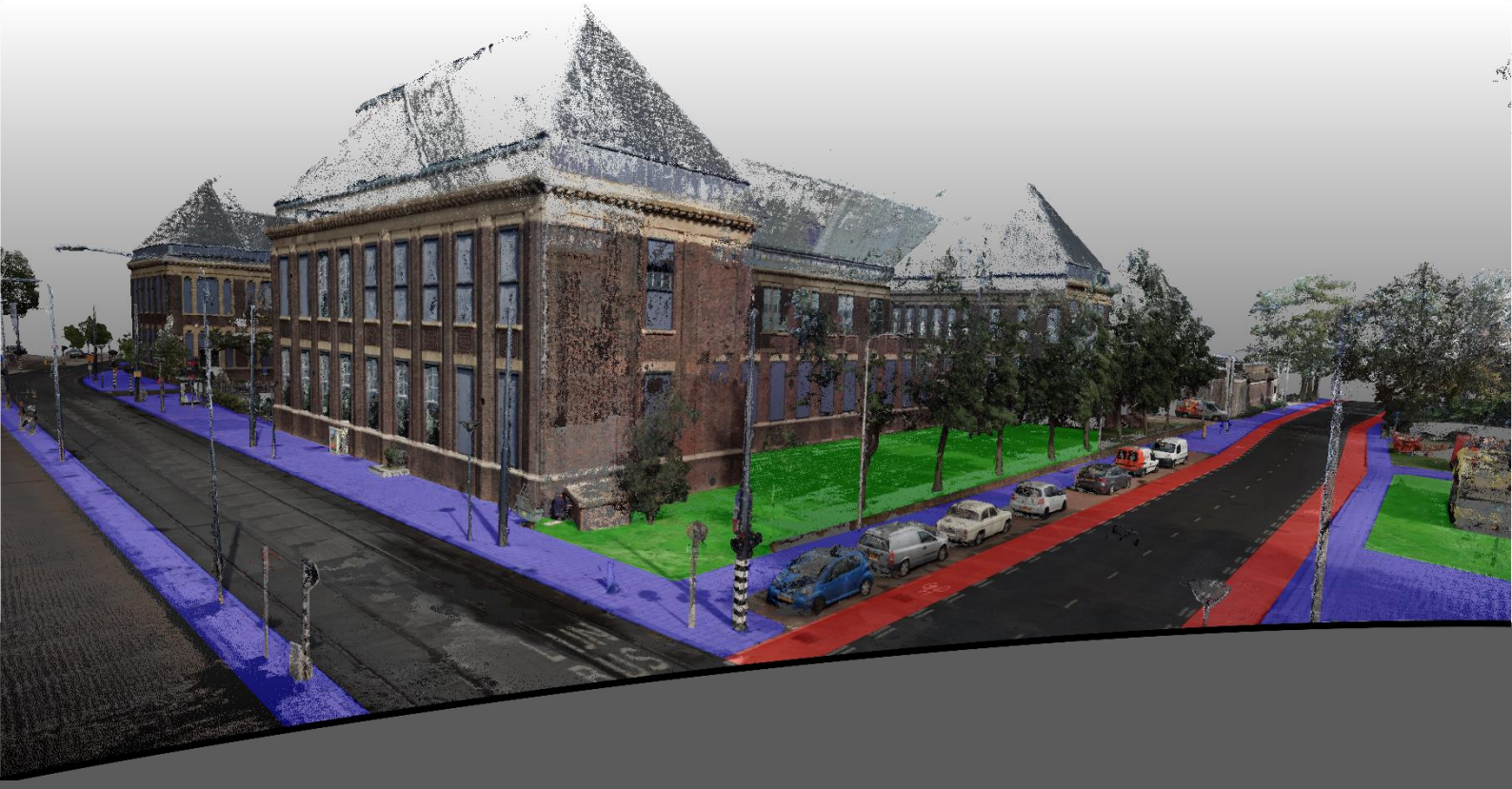


Technical Report

SEMANTICALLY ENRICHING POINT CLOUDS
THE CASE OF STREET LEVEL

Team Members:

Adrie Rovers
Irene de Vreede
Merwin Rook
Stella Psomadaki
Tim Nagelkerke

Coordinator: Stefan van der Spek
Coach: Wilko Quak
Domain expert: Bart Beers
Project Management: Robert Voûte

cyclomedia
Smart Imagery Solutions

TU Delft

Preface

The Geomatics Synthesis Project (GSP) 2015 is a nine-week long project undertaken by second year Master students of Geomatics. The project gives the students the opportunity to take part in a real world project with strict time constraints.
This year's GSP general topic focuses on Explorative Point Clouds meaning exploring ways to use the point cloud data directly. Three teams were divided and three approaches were developed. Each team is assigned to a company, which provides use cases to the group. This team is assigned to research new possibilities in the use of terrestrial point clouds. This terrestrial point cloud is offered by Cyclomedia, that offers its services as high resolution georeferenced cyclorama images.
Point clouds hold a lot of information, that is, in a lot of cases, not optimally used. These point clouds are now mostly used to make 3d models, while the points, and all their valuable information they hold, are wasted. In this research, we research and suggest an approach to get more information out of the point cloud data, creating new use cases and more opportunities for re-use.
In this approach, we aim at enriching the data with extra information that is of value for engineers, policy makers and can help in the optimization of our city designs that are getting more affected by changing and more extreme weather conditions. By adding this extra information, we aim at prioritizing the use of these valuable point clouds, as this hard information with a very high resolution, and offers us new opportunities to understand, interpret and design the human environment in a more effective, cost efficient and innovative way.

Table of Contents

List of figures

List of Tables

List of abbreviations

2D	Two Dimensional
3D	Three Dimensional
AMN	Associative Markov Networks
ASCII	American Standard Code for Information Interchange
Avg	Average
BIM	Building Information Model
BK-City	Faculty of Architecture Technical University Delft
CGP	Ground Control Points
CIE	Commission Internationale de l'Eclairage
CRF	Conditional Random Fields
DEM	Digital Elevation Model
GNSS	Global Navigational Satellite System
GPS	Global Positioning System
GSP	Geomatics Synthesis Project
HD	High Definition
HSI	Hue Saturation Intensity
HSV	Hue Saturation Value
INS	Internal Navigation System
kD-Tree	k Dimensional tree
k-NN	k Nearest Neighbours
LAS	Log ASCII Standard
LAZ	Log ASCII Zip
LiDAR	Light Detection and Ranging
MCC	Multiscale Curvature classification
NE	North-East
NW	North-West
PCA	Principal Component analysis
PCL	Point Cloud Library
PDAL	Point Data Abstraction Model
RANSAC	RANdom SAmple Consensus
RGB	Red Green Blue
SE	South-East
SIC	Surface Infiltration Model
Std	Standard Deviation
SVM	Support Vector Machine
SW	South-West
TIN	Triangular Irregular Network
WKT	Well Known Text

Abstract

Growing possibilities in the use and cheaper techniques and methods in the acquisition of point clouds, rapidly developed the use and production of point clouds. These point clouds consist out of millions of points, and can therefore be qualified as big data. Difficulties in the processing of these point clouds lead to a transformation of the points into a model, resulting that the raw information is not used anymore. The aim of this research is to develop and investigate techniques that create added value to these point clouds, by adding extra class information, also described as a label, to these points. We therefore formulated the following research question: How can a point cloud be semantically enriched by providing a labelling process using geometrical properties and colour values? The focus while answering this question lies with labelling a coloured point cloud, created by a mobile mapper of Cyclomedia, whereby only the ground points are labelled with: Grass, Road, Cycle Path or Tiles (Footpath).
While investigating such a research topic several methodologies were tested for their value. However, not all of them were able to provide the required output. In addition to that, working makes it necessary to test different indexing techniques. To create classes or in other words, labels, different classification methods, point based and region based, are described and tested.
This research shows that a region growing algorithm, based on curvature, the point normals, and colometrical properties, in combination with training samples gives the best results, with a mapping accuracy of 92% for road, 77% for cyclepath, 80% for tiles and 99% accuracy for grass. These result can be further improved by our developed smoothing algorithm.

Acknowledgements

The team would like to take the opportunity to thank the people that played a substantial role during the development of this project.

We would like to thank the coordinator of the Synthesis Project, Stefan van der Spek for organising the content of this year's Synthesis Project. We would also like to thank Wilko Quak for his invaluable input and tutorship during the development of the methodology during those 9 weeks.

A special acknowledgement is to be given to Bart Beers for giving us this opportunity to work with Cyclomedia's products and for this willingness to answer every question concerning the company and the processes used by them.

We are also grateful to Danbi Lee, a former Geomatics student, for providing us with ideas and advice for the applications investigated in this project.

Last but not least, we would like to thank the coaches of the other teams Edward Verbree, Martijn Meijers and Theo Tijssen for their inputs and ideas during the reviews, as well as Robert Voûte for his insight about project management.

Chapter 1: Executive Summary

1. Introduction

In today's technologically driven era, the terms position and location are highly affecting our lives. In addition to that, the introduction of virtual earths and street view applications have well shifted our perception of geo-information from 2D towards 3D. This shift was also affected by new technologies like LiDAR that automated the generation of 3D city models. LiDAR technologies within a decade matured leading to the use of point clouds into everyday applications. But despite the immense amount of information hidden inside a point cloud, its information in today's city modelling paradigm is often not exploited to its best potential. In the majority of today's projects, point clouds get thrown away after 3d models are created. These point clouds are for a lot of users too big to handle, and therefore not easily replaced by traditional data sources like surveying. However, it is foreseen that with the increasing power of mobile devices like smartphones or tablets and their browsers, those days of insufficient computing power are coming to an end. It is therefore relevant to explore ways to utilise the point cloud data directly.

Cyclomedia, a specialist in capturing, processing and visualising the urban environment by using cyclorama images, is the project client of this research. For Cyclomedia, the road is the central object for its data collection and, therefore, investigating the ways to extract road information from point clouds is central for this research. In addition to that, the road structure in a city can be very useful in extracting other features, like: buildings, traffic signs, vegetation or cars. Besides road extraction, the aim is to label point clouds with their function or composite material. These labels can help in order to create certainty about surface conditions, such as the surface infiltration capacity (SIC). A SIC map of an urban area could be very useful input for evidence-based urban design of water-sensitive/ low-impact neighbourhoods and water management schemes for reducing flood vulnerability.

In this research, the aim is to create a workflow that results in a dataset that will help urban planners, urbanists and road- and city maintenance organisations in doing their jobs better and more efficient. In other words, the aim is at enriching the data with class labels, giving it added value and new user possibilities. The central question in this research is therefore: *How can a point cloud be semantically enriched by providing a labelling process using geometrical properties and colour values?*

The executive summary is organised as following. Section 2 describes the Data collection equipment and process. Section 3 describes the pre-processing workflow that together with the workflow presented in Section 4 was proven to give the best results for street level extraction. The summary ends with section 5 where general conclusions and recommendations for this project are given.

2. Data collection

The data collection for this project is performed by Cyclomedia's mobile mapper system. The system used was their newly developed prototype system with LiDAR, five cameras and a combination of GNSS receiver and INS system to establish the position of the car. The LiDAR scanner used was the Velodyne HDL 32 that has 32 channels and catches 700,000 points per second with ± 2 cm accuracy. The horizontal field of view of the scanner is 360°. The five cameras, on the other hand, together produce 360° panorama images. These images are used in the process of stereo-imaging to create the point cloud, and to provide the LiDAR point cloud with colour. In this project, only the stereo imaging point cloud is used which contains points with a three-dimensional position calculated through the observation of a point by multiple images.

3. Pre-processing

3.1. Non-ground filtering

Point clouds contain rich information about the scene that they were obtained from. However, not all information is relevant for all applications. In the case examined in this project, the focus area lies on the ground surface. This means that the non-ground points (buildings, trees, cars) should be removed from the dataset. After investigating several algorithms, the non-ground filtering process offered in LAStools was proven to produce the best results.

3.2. Relevant classes

Designing a workflow that would extract the road information and its surroundings first requires the definition of the application relevant classes. In this case road extraction and materials focused on street level are examined. The most relevant subcategories are: *Grass, Road, Cycle path* and *Footpath*. Those categories represent how humans interpret the street level. For more specific applications, (e.g. the surface infiltration capacity), where the composition of materials is needed, a specialist would benefit from a classified point cloud with the classes *asphalt*, *grass* and *tiles*.

3.3. Point cloud indexing

Since point clouds from their nature are big data, quick access to the points can be incorporated by applying a spatial index. After comparing octrees and kD-trees, the kD-tree data structure turned out to provide the most balanced structure and the quickest access to the neighbouring points of each point. The reason for this is that the kD-tree makes use of a 1D distance comparison for indexing whereas in the case of an octree 3D distance calculations are needed.

3.4. Geometrical properties of the neighbourhood

The nearest neighbours are used to calculate 3D features for every point, because geometrical information of one specific point does not provide much information on its own. However, if a neighbourhood of the point is analysed, much can be understood with the geometrical patterns.

These geometrical properties of a point's neighbourhood can be calculated with the principal component analysis (PCA). PCA can be used to estimate the eigenvalues which can be used to infer different measures, like linearity and planarity, that describe the regions' properties. The only measures that are eventually used in our algorithm are the normal and the curvature. Other researched geometrical properties of a point's neighbourhood, like planarity, scattering or the sum of the eigenvalues, are not sufficient to make a distinction between already horizontal planes.

3.5. Colour spaces transformation

Applications for point clouds have been greatly extended since the introduction of colour in them. This is because humans can correlate colours to semantics and thus infer a great amount of relevant information. The colour information of the point cloud is therefore used to cluster points together that possibly belong to the same class. However, the RGB colour space that is easy interpreted by a human, possesses a lot of difficulties in computer vision since colour differences are not uniform. Therefore, more uniform colour spaces namely, HSV, HSI and CIE were investigated. Those spaces transform the highly correlated RGB values to less correlated components. The results of those colour transformations were tested for their applicability in distinguishing between the different classes in later stages.

4. Point cloud labelling

Labelling a point cloud in this research means getting those parts out of the point cloud necessary for assessing a certain task. Training data is used to make it possible to label, classify, parts of a point cloud. This training data, samples, are manually selected clusters of points with a known class. In choosing the right samples, it is important to pick various regions and clusters of points that show inconsistencies in colour, compared to the overall class. Experience showed that these variations in training data increased the classification results.
In order to identify the method that produces the best outcome, gets the necessary parts out of the point cloud most accurate, two methodologies were investigated: a point based and a region based classification method.

4.1. Point based classification

The point based method investigates whether it is possible to label a point cloud by only utilising the colour properties of each individual point. Point based labelling is only possible if samples for each class are available. From these samples, the range of the colour value in HSV or CIELab is determined, by calculating the

minimum and the maximum value for each class. Next, it is checked if the colour properties for every individual point are either between the minimum and maximum of each component per class or between the mean minus or plus 2 sigmas. Machine learning algorithms were also tested on point properties. Although some promising results came from these classifications, point based methods were proven to be error prone since a lot of colour inconsistencies are present in the point cloud. CIELab, in general, gave the best distinction but the accuracy assessment for point based methods proved to be not sufficient.

4.2. Region based classification

Region based classification schemes take the neighbourhood properties of the points into consideration. These properties of the individual point are used to determine if neighbouring points hold the same properties, and belong to the same region or class, or not. The neighbourhood properties that are used are curvature, which gives an indication of the smoothness of the underlying surface, the normal, which is the vector that is perpendicular to the points region and the colour and colour distance.

To grow a region, a seed point is chosen, and all neighbouring points are tested if they hold similar values on certain properties, if they do, and therefore belong to the same class, the points are added to the region. This process is then repeated for all new points in that region, until no newer points are found that belong to that region.

In the applied region growing algorithm, a low curvature threshold is used, because the majority of the classes are found inside the same plane, therefore raising the curvature threshold leads to under-segmentation. Under-segmentation means that two different classes are labelled as one. While, when a surface is divided into really small surfaces, you can call this over-segmentation. Under-segmentation happened with curvature thresholds over 0.001. The decision was made that over-segmentation is more acceptable than under-segmentation. Intentionally causing over-segmentation is, at least in theory, much simpler to correct than under-segmentation, since human intervention cannot be avoided in the second case.

4.2.1. Unsupervised classification

The previously caused over-segmentation of the region growing algorithm is tackled by a merging algorithm that is based on colour similarity. The main idea is that the small regions (with points less than 5000) that are formed in the region growing algorithm, are merged with their neighbouring regions that have the most similar colour. The colour similarity is compared by calculating the euclidean distance of the components of the CIELab space, excluding the L component that represents lightness. The most similar region is the one with the lowest value. The result shows more relevant regions. However, there are still cases of over-segmentation and even under-segmentation, where two classes are labelled as one.

4.2.2. Supervised classification

The supervised approach to classifying a point cloud requires a set of training sample data that have labels associated with them. This sample data comes from a scene and is selected with prior knowledge about the scene. This means that, for the area of interest of this project, samples are collected that belong to the road, cycle path, tiles and grass.

The different classification methods that are tested can roughly be subdivided in two categories: First, Gaussian processes, which assume that the class properties are Gaussian distributed, and a class gets assigned to a point or a region with the highest probability. Second, machine learning algorithms, that are implemented by making use of the python module sklearn, that allows classification through simple vectorised training samples and data. From these training samples, a decision space is created and used to classify the regions or points. The output of the Gaussian process is most satisfactory, with a mapping accuracy of 92% for road, 77% for cyclepath, 80% for tiles and 99% accuracy for grass. The sklearn module didn't give satisfactory results, as a lot of points, that don't belong to one of the predefined classes, are assigned a class. This is caused by the functioning of the algorithm, that is written in such a way that each point or every region gets assigned to the class with the highest score, even if it is very unlikely that this class should be assigned.

4.2.3. Smoothing

To tackle the errors of omission and commision, a smoothing algorithm is developed. This method is based on image smoothing filters that remove noise. The algorithm traverses all points in the classified point cloud. For every point it checks its k nearest neighbours and calculates what is the most frequent class among these neighbours. Next, the point gets reclassified if the original classification differs from the outcome of the smoothing algorithm. This process offers both a visual and an accuracy advantage to the final result.

5. Conclusions & recommendations

5.1. Conclusions

From the outcomes of this research, it becomes clear that a region-based supervised classification using training samples is the best way to distinguish between the contextually relevant classes. This is because the colour information of the point cloud is often times less accurate than the positional information. The colours of our point cloud show some differences in relation to reality and therefore unsupervised schemes present over-segmented results. Furthermore, over-segmentation with unsupervised classification can be the result of shadow and water on surfaces, which then appear darker, worn out colours of the

road/cycle path markings, different materials of tiles and other noise in general. To work around those inaccuracies, training samples can be provided. This way a supervised classification captures all appearances of a class whereas an unsupervised classification would create different regions. However, during the project, the researchers realised that one training sample per class is not sufficient. Therefore, the user has to provide the best combination of training samples per scene that will not be excessive in number and amount, but representative enough of the colours present. Finally, extending the method to other areas can be succeeded, but only if the provided training samples are representative.

5.2. Recommendations

The following recommendations are provided as future work for the project. Recommendations one and two are based on difficulties and inefficiencies in the manually picking of training samples. The third recommendation is a solution for the difficulties we faced in the region growing on colour values.

1. Explore the possibilities to match the regions with land use maps.
2. Provide the positional information of the mapping vehicle/device with (each part of) the point cloud.
3. Radiometrically enhance the cyclorama images before the point cloud creation.

Future work can be found in relating the point clouds, in particular the semantically labelled parts, to over simplified models of reality mostly utilised today by scientists.

Chapter 2: Introduction

Data acquisition of point clouds can be done by both laser scanners and high accurate georeferenced images. It can be said that scanning is embattled with the much older technique of creating point clouds with photogrammetry. Scanners provide a product in just one step, creating the point cloud by photos is much more complex but also much cheaper (Geodatapoint, 2013).
The point clouds that are generated with these techniques, consist out of a big set of points. These point collections give the possibility to create geometric correct representations of objects and landscapes (Hmida et al., 2013). Furthermore, the possibility of processing dense point clouds in an efficient and cost-effective way, creates a lot of possibilities in different fields such as topographic, industrial or cultural heritage (Vosselman and Maas, 2010). Point clouds therefore are datasets which become more popular every day with more and more uses in the urban environment, such as collecting information about building facades, street furniture etc.
However, despite their popularity, point clouds are usually not used to their full potential. Usually, these datasets are transformed and translated into models, such as 3D BIM models, while the original point cloud dataset is deleted (van der Spek and Verbree, 2015). Not using the raw point cloud, means throwing away a lot of valuable information. The scope of this research is to carry out operations on a raw dataset, whereby the richness of the data is optimally used and the use of the point cloud optimized.
This report is organized as follows: Chapters Introduction focuses on the problem description, the application and objectives. Chapter Literature Overview contains a relevant literature overview. Then, chapter Methodology will focus on the methodology used in this research. Chapter Implementation will describe the implementation of the literature and the used methods. Next, we conclude with our findings, followed by recommendations for future work. As an addition the project management and process are recorded in Appendix A, the developed python code is recorded in Appendix B. The report ends with the Media Outreach Strategy in Appendix C.

1. Problem description

The human brain is capable of interpreting point clouds by using and combining information about the scene, the point attributes, memory and by recognizing shapes and identifying objects. Computers, on the contrary, do not have this broad and intelligent spectrum of additional information and interpretation as humans do. Therefore, the recognition and interpretation of these point clouds by computers is considered hard. Different authors recognize these problems and aim at semantically labelling and recognizing objects in point clouds by using and combining different rules (Xiong et al., 2011, Shapovalov et al., 2010, Vosselman and Maas, 2010, Hmida et al., 2013).
This research aims at creating a methodology to create a dataset with semantically labelled points. Together, these labelled points create a scene that is better

interpretable by computers and can be used in analysis of the human environment. Therefore, the research question is:
How can a point cloud be semantically enriched by providing a labelling process using geometrical properties and colour values?
In order to answer this research question, the following sub questions have to be answered:

- How can a point cloud be indexed?
- How can the non-ground points be filtered out?
- What subcategories of ground are important?
- How can subcategories be classified?
- What is the best way to label a point cloud?

2. Application and objectives

Point clouds hold a lot of information, information that is often not used. In this research, we aim at creating a workflow that results in a dataset that will help urban planners, urbanists and road- and city maintenance organisations. The raw data that is used comes from Cyclomedia. Cyclomedia aims at delivering a product to support decision makers and urban planners, as they state on their website: "CycloMedia Technology Inc. utilizes analytic measurement software technology in conjunction with 360-degree panoramic imagery (HD-Cycloramas) to help planning departments better analyse and understand their cities" (Cyclomedia.com, 2015). The end product, meaning the semantically enriched data, will help other parties and organisations to do their work more efficient and in a more precise way.
This chapter describes the analyses of the customer demand for the enriched data. For this, two different use cases are described. The first use case covers the extraction of pre-specified features from a point cloud, whereby the road is the main subject. Other classes could be: vegetation (grass), tiles (footpath) and cycle path. Secondly, enriching the point cloud with data for further analysis is described. This use case focuses on providing data for water retention calculations, surface infiltration capacity analysis (SIC), and flood risk analysis, specifically flooding caused by heavy rainfall, as described by Lee (2013).

2.1. Road extraction from point clouds

Niemeyer et al. (2014) recognize the problem in object extraction from point clouds: "There are different types of objects such as buildings, low vegetation, trees, fences, and cars that can be found in a small local neighbourhood, which makes it difficult to extract them reliably. In order to handle this problem, research often focuses on the extraction of a single object type, i.e. buildings, roads, and trees". The road can be used as guiding object in the search of other objects, in other words: identifying the road as main object in the urban environment gives the possibility to identify other materials and objects around the road.
The extraction of the road surface has been a subject of research for the past years. Different approaches have been tested and described. Smadja et al. (2010) use curve fitting through parts, or through one scan line, to find the road surface. (Boyko and Funkhouser, 2011) use additional data, like topographic maps. Other

research aims at recognizing objects near the road, like signs, lights (Landa and Prochazka, 2014) or road markings (Yang et al., 2012). Only few use raw point data to find the road surface. Geosignum, a Dutch start up, managed to automatically extract roads from point cloud data, as they state on their website: "changes in the existing road alignment geometry, road curvature and cross-slopes can be automatically calculated and mapped. In addition to this, road surfaces and road edges-lines can be automatically extracted in a very short time for further analysis or a new engineering design". However, Geosignum does not publish about the techniques they use (Geosignum, 2015).
Roads are one of the main objects that are to be recognized by our algorithm, as It represents one of the most present and important structures in urban or manmade landscapes. Thereby, roads can form a basis for further feature extraction, as a lot of (urban) objects are situated near roads. Thereby, road maintenance companies can use the point cloud, because the extraction of weak spots, holes and bumps can be done more effective, efficient, and precise and with lower error than when it is done manually.

2.2. The use for surface infiltration capacity analysis

Lee (2013) states that: "Effective urban water management requires certainty about surface conditions such as the surface infiltration capacity (SIC). A SIC map of an urban catchment area could be a useful input for evidence-based urban design of water-sensitive/low-impact neighbourhoods and multi tiered water management schemes for reducing flood vulnerability". Thereby she states that: "methods for mapping SIC are underdeveloped". Lee used multispectral images for the classification of the earth surface. This approach has some scale and accuracy problems, as the roughness, caused by mixed pixels, of images does not allow small scale object or material detection. Other analysis mainly focuses on the geological properties and water infiltration capacity of certain types of soil (Bean et al., 2004).
Our way of analysis allows for analysis on the now smallest scale, while upgrading to a higher scale should be possible with the use of Cyclomedia's mobile mappers. Lee (2013) analysed and described the properties that heavily affect the water retention capacity of urban soils. She identified six parameters that affect the water retention capabilities:

1. Surface cover void ratio (water-penetrable surface area reduced by siltation over time).
2. Surface material (affecting adsorption).
3. Surface roughness (including soil crusting, affecting flow resistance and overall SIC).
4. Subsurface hydraulic conductivity, where the sub-grade layer with the lowest conductivity is limiting.
5. Subsurface compaction (measured by bulk density or void ratio and indicated by use and age relationships).
6. Subsurface antecedent soil moisture, or how wet is the soil already.

Thereby, the fluid viscosity, the hydraulic gradient and temperature also play a major role in how the water interacts with the earth’s surface.

The current used technique is inaccurate as the information is derived from topographical maps. Thereby, the classes that are currently used are mostly not usable for SIC analysis, as they represent the use of a certain area, and not the material. The parameters that we expect can be calculated by LIDAR data are: the surface material including the surface cover void ratio and the surface roughness. Even with LiDAR, not all the needed information can be extracted for accurate SIC analysis, but it is an improvement as the former techniques were less precise and less sufficient.
The scale of the analysis will, in first place, be kept small. Extending the technique to higher scales should be possible. Thereby, the research will in the first place be limited to surface material detection. Although vegetation, like trees and bushes, plays a major role in SIC analysis, non-ground vegetation will not be mapped in this research.

Chapter 3: Literature overview

1. Point Cloud Data Acquisition

Laser scanning is an emerging technology in the field of surveying that provides "rapid and detailed acquisition of a surface in terms of a set of points on that surface, the so called point cloud" (Shekhar and Xiong, 2008). The systems that are used to acquire the point clouds use non-contact techniques. Point clouds can be obtained to describe surfaces both on land and offshore.

1.1. Laser scanning techniques

The two ways for acquiring a point cloud is light transit time estimation and triangulation (Vosselman and Maas, 2010). A light wave travels through a specific medium in a given velocity. For this reason, the time difference of the light travelling from the source sensor to the reflective surface and back is good estimator of the distance measurement. The systems that utilize this concept are known as time-of-flight systems. It is also possible to use phase measurement techniques as an indirect measure of the time-of-flight. The second way is triangulation. This method utilises the cosine law by emitting a laser beam on a reflective surface and observing the position of this beam from an observation point (usually a camera) at a known distance. In the figure provided by (Vosselman and Maas, 2010) the previous techniques are schematically presented in Figure 1. Each one of these techniques have their own trade-offs, advantages and costs.

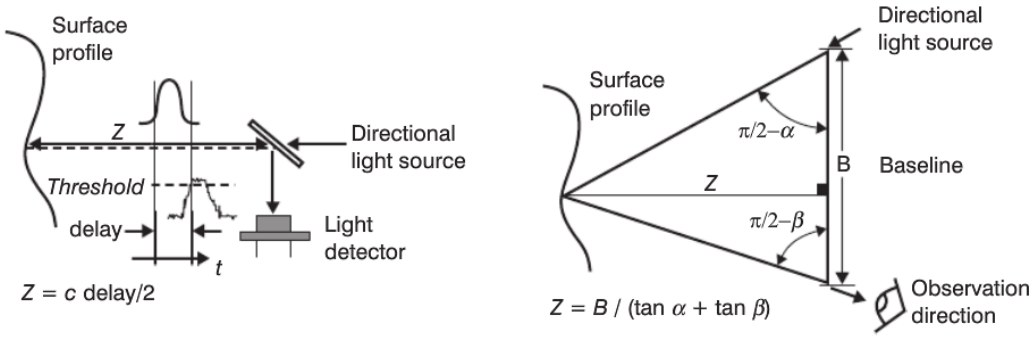


Figure 1 Scanning Techniques Left: Time-of-flight, Right: Triangulation
Source: Vosselman and Maas, 2010

1.2. Image based techniques

Another point cloud acquisition technique emerging in the last decade is generating dense point clouds out of images. This image-based point cloud generation is based on stereo photogrammetry which compares to the human vision and the principle of parallax. If a point is observed by different viewpoints in different images, the three-dimensional position of the point can be then computed. This gave emerge to "Structure-from-motion" (Westoby et al., 2012) a low-cost photogrammetric method that generates 3D point clouds by without any a-priori knowledge of the camera calibration and position. All the parameters are calculated

by image matching techniques. The object can finally be positioned in space by applying a 3D similarity transformation using Ground Control Points (GCPs). This is achieved by acquiring multiple overlapping images (Figure 2) of the feature of interest. This process has some advantages compared to the LiDAR technique in that it captures the colour and thus the point cloud is provided with context. Several projects have utilised this technique including Photo-tourism (Snavely et al., 2006), documentation of monuments etc. Several open source implementations exist like Bundler (Snavely et al., 2006), VisualSFM (Wu, 2013), but also several commercial software have been developed e.g. Agisoft PhotoScan, Pix4D, 123D Catch etc.

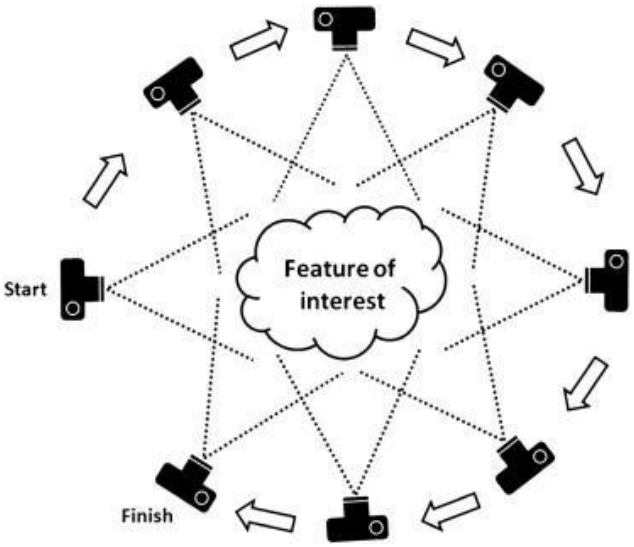


Figure 2 Structure from motion; multiple overlapping images
Source: Westoby et al., 2012

1.3. Acquisition methods

The acquisition of points can be performed with the following methods:

- Tachymetry: This method does not provide point clouds in the sense defined before but with this technique points of interest are measured. The accuracy provided is high.
- Terrestrial Laser Scanning: If a dense and accurate point cloud has to be obtained for a small area, this method is the most appropriate. In order to cover the whole area defined, multiple scans might need to take place.
- Mobile Mapping: This technique is mostly used for larger areas. Laser scanners and cameras with the combination of GPS and INS systems are mounted on cars which obtain data from highways and roads. The data from the mobile mapping can be acquired in two modes (Vosselman and Maas, 2010, p. 293): the stop-and-go mode where the vehicle makes short stops to scan the area, and the on-the-fly mode where the vehicle obtains the data without stopping.
- Airborne Laser Scanning: This technique is done from airplanes or helicopters which are mounted with a laser scanner system that measures the distance between the scanner and the ground and a GPS and INS system which acquires the position and orientation of the aircraft. This method is used to acquire Digital Elevation Models (DEM).

2. Non-ground filtering

The different data that is captured by the laser scanning or image based method include buildings, vegetation, city furniture, bare ground etc. Especially bare ground identification and thus non- ground points filtering has been a quite difficult task (Zhang et al., 2003). Different concepts for extracting the bare ground points have been investigated through the years. In Sithole and Vosselman (2004), four different filtering concepts for airborne methods have been compared: Slope based, block minimum, surface based filters and segmentation algorithms. Since many algorithms have been developed, choosing the one that suits our needs can be rather challenging. According to (Korzeniowska et al., 2014) the decision for choosing the right algorithm is based on processing time, the complexity of adjusting the different parameters, and the verification of the accuracy of the final result. In the next section, the different filtering methods available in commercial and open source software will be evaluated in relation to the Synthesis Project requirements.

2.1. Point Data Abstraction Library (PDAL)

The PDAL is a library suitable for reading, manipulating and writing point cloud data (Butler and Gerlek). PDAL in its capabilities provides a bridge to the Point Cloud Library (PCL) by providing processing pipelines. To use the PCL the user must have installed both PDAL and PCL. So, taking advantage of the command-line capabilities of PDAL, the progressive morphological filter by Zhang et al. (2003) is tested. The algorithm detects non-ground measurements from LIDAR data by gradually increasing the window sizes of morphological filters and by using an elevation difference threshold that depends on the elevation variation of the specific terrain being filtered.

2.2. LAStools

LAStools by Martin Isenburg is "a collection of highly-efficient, scriptable tools with multi-core batching that process LAS, compressed LAZ, Terrasolid BIN, ESRI Shapefiles, and ASCII" (LASTools, Accessed on 18/09/2015). In the collection of LAStools, lasground is contained. Lasground is a tool for bare earth extraction. It belongs to the method of progressive densification (Pfeifer and Mandlburger, 2009) and specifically is a variation of the algorithm developed by Axelsson (2000). According to the algorithm, the bare ground is identified by progressively densifying a sparse TIN with the laser point cloud. For a point to be classified as bare ground, it must be within a certain threshold of a minimum distance from the triangle it is contained in and the angle relative to the triangle (Figure 3).

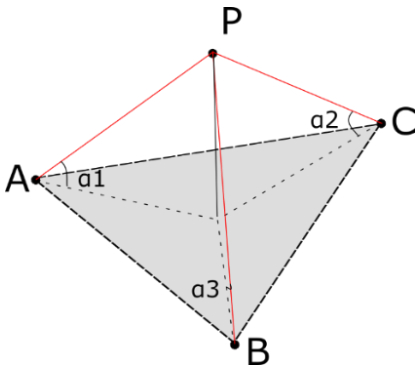


Figure 3 Schematic overview of the parameters taken into consideration by the algorithm of Axelsson

2.3. Multiscale Curvature Algorithm

The Multiscale Curvature Algorithm for Classifying Discrete Return LiDAR in Forested Environments (Evans and Hudak, 2007). According to the paper, this algorithm was developed, and is therefore best fit for forested areas.

3. Point cloud indexing

Spatial indexing has always been an issue for data accessing and querying, even within the 2D space. A spatial index is used to efficiently perform a spatial selection. Not having an index means a sequential traversal of all the data within a file or database in order to check a spatial criterion (van Oosterom, 1999, p. 385). When working with large amounts of data, as in the case of point clouds, this is inefficient and requires a lot of processing power and time.
The kD-tree, a main memory data structure, is a first option for the indexing of point clouds (van Oosterom, 1999, p. 386). A kD-tree can store points of any dimension K. A 2-dimensional example is shown in Figure 4. The kD-tree recursively divides the space using a root-leaf structure. The root corresponds to the complete spatial area and the leafs represent the resulting areas of the split. If an area is not split any further it becomes an end-leaf. In turns, the area is split on the X- and Y- axis. For the 3D case the Z-axis is also included.
Two main variants of the kD-tree exist. In the first case a point from the dataset is used to split the area, resulting in a tree where every internal node represents a point, see Figure 4. On the left. There are different ways of constructing this type of kD-tree because we can use any node and any axis for splitting. Another option is to split the space, not on a point, but where the division results in two point sets with the same amount of points (approximately). This is also known as an adaptive kD-tree (Figure 4, right). The advantage is that the tree stays balanced. A balanced tree can be build faster and is more efficient, although it is hard to keep the tree balanced while inserting points (van Oosterom, 1999, p. 386). A third possibility is to split the area on points within the dataset, but in addition calculate the median of the points for splitting. This will also result in a balanced tree, but it will take more time to build than when random points are chosen.

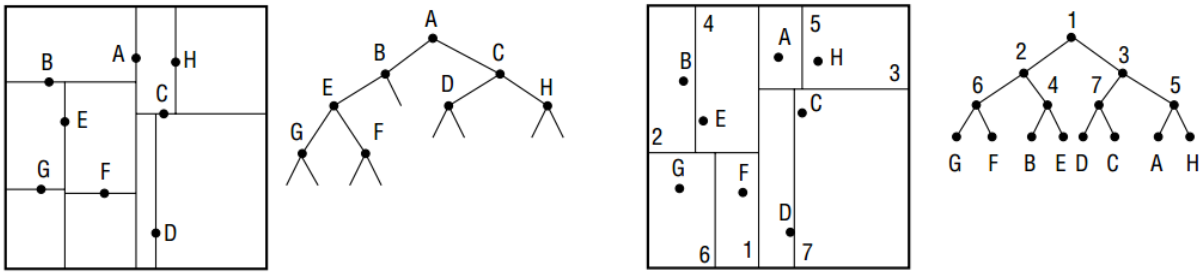


Figure 4 The kD-tree (left) and the adaptive kD-tree (right)
Source: van Oosterom 1999, (p. 386)

In the 2-dimensional space quadtrees and R-trees are mostly implemented and used (van Oosterom, 1999, pp. 391, 394). An R-tree groups features hierarchically according to their location. This is done using bounding boxes. A Quadtree on the other hand, recursively divides the space in four quadrants. The area's in a quadtree are then enumerated in a fixed order (NW, NE, SE, SW for example), and then represented in the tree. If all of these areas uniformly represent the same value, or satisfy a certain condition like a maximum amount of points, the area is not subdivided again and the leaf becomes a node.

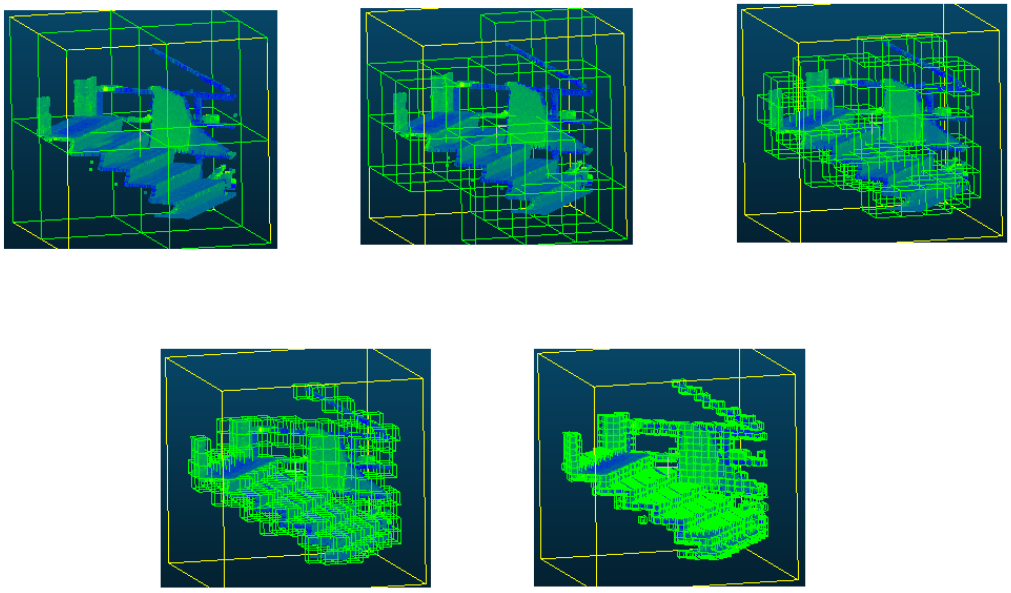


Figure 5 Example of an octree for the indexing of a point cloud (Own illustration)

The 3d version of the quadtree is called the octree. It adds an extra dimension z, which is the height. It is a commonly used spatial index for storing data of 3-dimensions in point cloud processing. An octree subdivides a 3d volume (R^3), the bounding box of the point cloud, into eight disjoint "cubes" (octants) filled with points, over and over again until each "cube" consists a defined number, or less, of points. The root of the octee is the "cubic" which holds all points of the point cloud. The leaves, or endnotes, of the tree hold a maximum predefined number of points. See Figure 5.

Just like the KD-tree, the octree and the quadtree are based on the root and leaf structure. See Figure 6. Using an octree is efficient for resolving multiple issues. An octree can store a raw point cloud, it ought to be fast in performing operations and is memory efficient (Elseberg et al., 2011).

Performing spatial operations is very similar for the quadtree, octree and kD-tree: Starting at the root, it is recursively checked if a node (subspace) intersects with the search range. If there is no intersection the leaf is no longer traversed. For the kD-tree this means that every intersecting point is reported. For the adaptive kD-tree, quadtree and octree this means that only in the intersecting end-leafs it has to be checked if points are included in the search range (van Oosterom, 1999, p. 392).

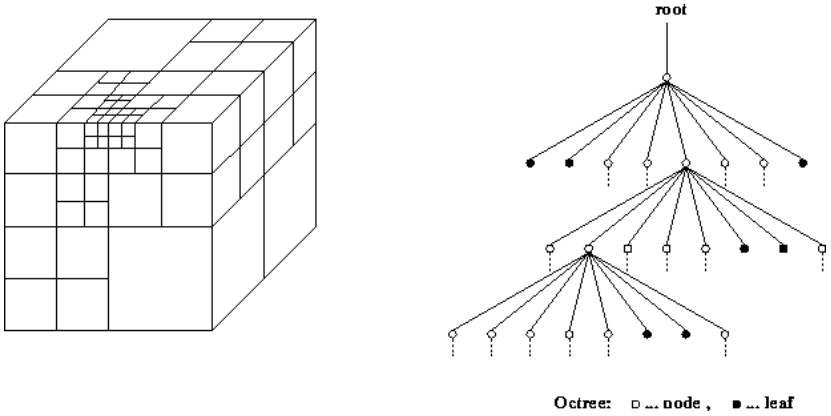


Figure 6 The octree
Source: Mäntylä, 1987

4. Nearest Neighbour Search

A point cloud is just a collection of points. However, a lot of information can be derived from a point cloud by applying certain algorithms. Most of these operations and algorithms that are performed on a point cloud, like: pattern recognition, object recognition, data-clustering, function approximation and vector quantization, are based on nearest neighbours information (Lai et al., 2007). As explained before, point clouds are huge datasets and therefore often decomposed in subdivisions (indexing). Nearest neighbour searches are therefore often based on these decompositions of the underlying space (Sankaranarayanan et al., 2007). Different methods for the decomposition of space and several options in calculating nearest neighbours lead to several possible combinations.
Three variants of nearest neighbour search can be distinguished, whereby a different number of neighbour-points are selected: The K-nearest neighbours search, the range based search and the K-nearest search within a certain radius (Elseberg et al., 2012, Sankaranarayanan et al., 2007).

4.1. K -nearest search

Searching for K-nearest neighbours within a point cloud means that for every point the k-nearest neighbours are found. K, the amount of neighbours that has to be

found depends on the application. Finding the k-nearest neighbours is useful when a certain amount of neighbours is required for each point of the point cloud (Elseberg et al., 2012, Sankaranarayanan et al., 2007).

4.2. Range based search

The range based search means that all neighbours within a certain radius $r \in \mathbb{R}$ for a query point $q \in \mathbb{R}^3$ are retrieved: $\mathcal{N}(q, r) = \{p \in \mathcal{P} \mid \|p - q\| < r\}$.

4.3. kNN and radius search combination

A combination of the two techniques can result in a retrieval of the k Nearest Neighbours within a certain radius (Elseberg et al., 2012).
Most of the libraries employ the kD-tree for nearest neighbour searches (Elseberg et al., 2012). kD-trees avoid unnecessary computations by making use of a tree structure, whereby, if traversing the tree, a direction is given as a Boolean until the subdivision of space intersects with the predefined sphere. (Behley et al., Kammerl and Muja, 2011). An example of a kD-tree radius search is depicted in Figure 8.

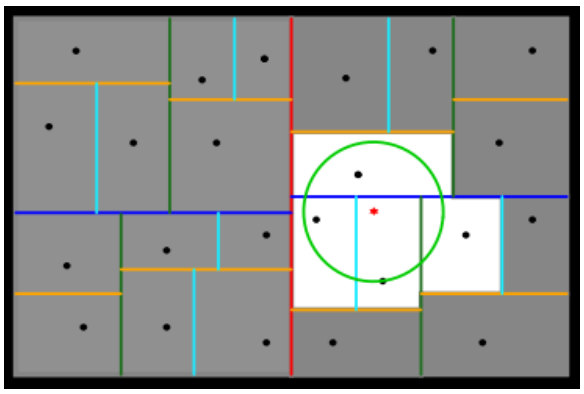


Figure 7 KD-tree radius search
Source: Kammerl and Muja, 2011

A nearest neighbour search can also be done by an octree. It allows a fast indexing and therefore it is easy to traverse quickly to the deepest nodes (Sankaranarayanan et al., 2007). Starting at the root, the octree is recursively traversed to investigate if octants overlap a search radius/sphere. Only when intersection occurs, the leaf is further traversed. For each intersecting end node, all points need to be tested against the search range (sphere) (Behley et al., Kammerl and Muja, 2011). Figure 9 depicts a quadtree search but the same principle applies.

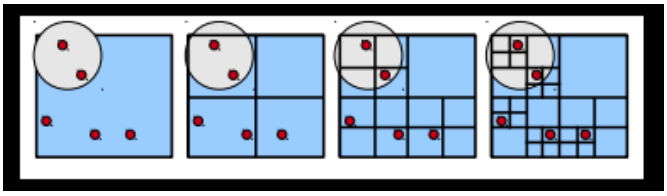


Figure 8 Principle of octree radius search
Source: Kammerl and Muja, 2011

5. 3D features characteristics

Point clouds consist of millions of points distributed on the scene. Information about a specific point thus does not provide much information on its own. However, if a neighbourhood of points is investigated much can be said about the geometrical patterns. The space that is taken around a point is called a neighbourhood and the points around it, its k-neighbours (Figure 9).

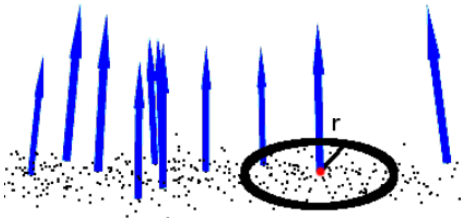


Figure 9 A neighbourhood around a point in a point cloud
Source: Rusu, 2010

5.1. Principal Component Analysis

Dealing with data that have more than 2D can be rather difficult for the human brain to interpret. This is also the case for 3D data, like point clouds. In order to solve this deficiency of the human brain, an alternative should be seeked which is for example to represent the 3D data to 2D, and thus drop a dimension. Principal Component Analysis is a classical technique for finding low-dimensional representations which are linear projections of the original interrelated data.

The transformation to this lower dimension is performed by transforming the original correlated data to a different set of variables, called the principal components, which are uncorrelated and in which the first variable retain most of the variation present at the original data. The computation of those principal components reduces the whole problem to an eigenvalue-eigenvector problem for a symmetric matrix (Jolliffe, 2002). For points in 2D space, PCA finds the best approximating line (Figure 10, left) that minimises the square distances. For points in 3D space PCA finds the best approximating space (Figure 10, right).

5.2. Estimating normals

The orientation of a local neighbourhood in the whole scene is a very important estimate in many applications like light shading and visual effects (Rusu R. B., 2010). This orientation is found by determining the normal of the plane that best describes the neighbourhood.
In three-dimensional geometry a normal is a vector that is perpendicular to the tangent plane of a surface given a point P at that surface. Calculating the normals of a surface is most of the times a trivial calculation. In contrast to this case, in point clouds, because of the fact that they are an approximation of a surface, the calculation of the normals is not a straightforward method. Two ways that this information can be acquired are (Point Cloud Library (PCL)):

- Create a surface from the acquired point cloud using meshes and then obtain the surface normal from the mesh.
- Use a neighbourhood to infer the surface normals, which is a least squares plane fitting estimation problem (Figure 10).

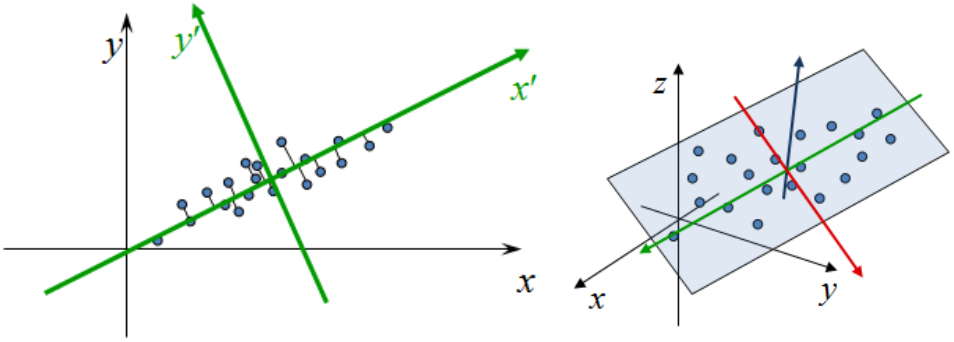


Figure 10 PCA finds an orthogonal basis that best represents a given data set
Source: Olga Sorkine, 2013

Since explorative point clouds investigate methods to work with point cloud without making a model as an intermediate step, the second option is explored for the rest of the project.

An insight into the local characteristics of the neighbourhoods are given from the Covariance matrices, which mathematically describe how neighbourhoods dispersed around their centroid. Given $\boldsymbol{p}_i \in \mathcal{P}^k$ a point in a certain neighbourhood in the point cloud, the centroid of of the nearest neighbors is computed as $\overline{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i$ .The calculation of the normal is then based on the analysis of the eigenvalues and the eigenvectors of the covariance matrix $\mathcal{C} = \frac{1}{k} \sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T,\ \mathcal{C} \cdot \vec{\mathrm{v}_j} = \lambda_j \cdot \vec{\mathrm{v}_j},\ j \in \{0, 1, 2\}$ of the neighbourhood $\boldsymbol{p}_i \in \mathcal{P}^k$, which is described as: $\mathcal{C} = \frac{1}{k} \sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T,\ \mathcal{C} \cdot \vec{\mathrm{v}_j} = \lambda_j \cdot \vec{\mathrm{v}_j},\ j \in \{0, 1, 2\}$, $\mathcal{C} = \frac{1}{k} \sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T,\ \mathcal{C} \cdot \vec{\mathrm{v}_j} = \lambda_j \cdot \vec{\mathrm{v}_j},\ j \in \{0, 1, 2\}$, Where $k$is the number of neighbouring points of p_i, λ_j is the j-th eigenvalue of the covariance matrix, and $\vec{\mathrm{v}_j}$ the j-th eigenvector (Rusu R. B., 2010). Finally, the plane normal n is the eigenvector of $\vec{\mathrm{v}_j}$ with the smallest eigenvalue λ_j.

5.3. General direction of the normals

The importance of the normals was mentioned in the previous section. However, the PCA method has no mathematical method for solving the sign of them. Therefore, the signs assigned are not consistent over a point cloud (Rusu, 2010). Rusu (2010) in his work mentions that if the viewpoint of the point cloud is known

this problem becomes trivial, as we consistency can be checked by the checking if the calculated normal satisfies the following:

$$\vec{n}_i \cdot (v_p - p_i) > 0$$

If this is not the case, then just changing the sign to the opposite solves the problem. This is depicted in Figure 11.

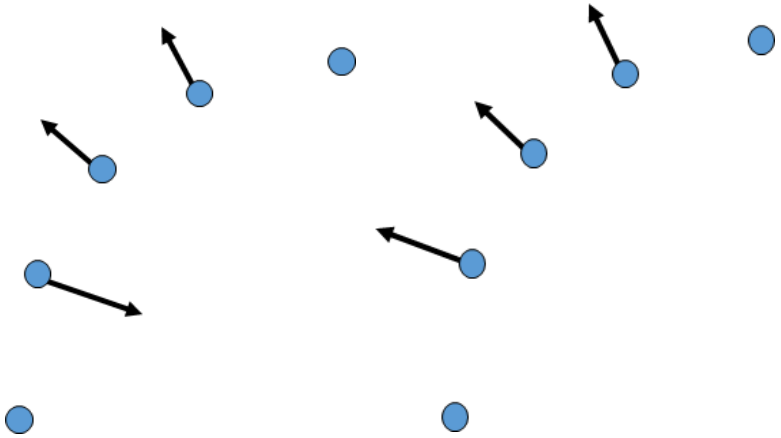


Figure 11 Left, before changing the general direction. Right: after changing the general direction. Source: Olga Sorkine, 2013

The methodology of these sections is summarised in the following pseudocode.

```
For each point in the point cloud
       Find k-nearest neighbours
       Find the centroid of the neighbourhood
       Calculate 3x3 the covariance matrix  of the neighbourhood
       Calculate the eigenvalues from the covariance
       Find the eigenvector with the smallest eigenvalue (normal)
       Calculate the vector from the point to the view point and call it dir
       If the dot product of the normal with the dir is <0:
              Change the sign of the normal
```

6. Geometrical local surface properties

The eigenvalues calculated with the PCA method are used to infer different measures that describe a surface. Eigenvalues are thus used for feature extraction. The eigenvalues, defined as $\lambda_{1,i}$, $\lambda_{2,i}$, $\lambda_{3,i}$ whereby $\lambda_{1,I} \geq \lambda_{2,I} \geq \lambda_{3,i} \geq 0$, are described as the two recent approaches in selecting individual neighbourhoods. Within these approaches 3 other main dimensionality features are described which can be used within the classification: linearity ($L_{\lambda,i}$), planarity ($P_{\lambda,i}$), and scattering ($S_{\lambda,i}$) (Weinmanna et al., 2015). The other 3d features that are described in Weinmanna et al. (2015) are omnivariance ($O_{\lambda,i}$), anisotropy ($A_{\lambda,i}$), eigenentropy ($E_{\lambda,i}$), the sum of the eigenvalues ($\Sigma_{\lambda,i}$), and change of curvature($C_{\lambda,i}$). These features can be found by using the normalized eigenvalues: $e_i = {}^{\lambda_i}/_{\sum \lambda}$ (Weinmann et al., 2014). The description of each property is given in the following lines.

- Curvature: "For a plane, the curvature is the absolute value of the rate of change of the angle of inclination of the tangent line with respect to distance along the curve" (James & James, 1992). The curvature using PCA can be

calculated by the ratio between the minimum eigenvalue and the sum of the eigenvalues. This approximates the change of curvature in the neighborhood of a point p (Rusu, 2010).

- Linearity: The feature of linearity is used to investigate whether a set of points can be modeled by a 3D line. An example where this characteristic could be used is for the extraction of powerlines (Waldhauser, et. al., 2014).
- Planarity: The planarity feature is used to describe the smoothness of a surface (Waldhauser, et. al., 2014).
- Scattering: The feature of scattering investigates the sphericity of a neighbourhood.
- Omnivariance: The feature of omnivariance describes how a neighbourhood of points spread inhomogeneously across a 3D volume (Waldhauser, et. al., 2014).
- Anisotropy: Anisotropy is a measure which is higher if the eigenvectors differ a lot. Anisotropic operations have the potential to discriminate between orientated and non-orientated objects, as oriented objects have a higher anisotropy (Oude Elberink and Maas, 2000).
- Eigenentropy: The feature of eigenentropy a provides a measure of the order or disorder of 3D points within the covariance ellipsoid (Weinmann, et. al., 2014)
- Sum of eigenvalues: The sum of eigenvalues describes the total variation which is the sum of the squared distances of the points of a neighbourhood from their centroid (Jolliffe, 1986).

The mathematical expression of the previous properties in terms of the three eigenvalues from PCA is given in Figure 12.

Linearity:	$L_\lambda = \frac{e_1 - e_2}{e_1}$
Planarity:	$P_\lambda = \frac{e_2 - e_3}{e_1}$
Scattering:	$S_\lambda = \frac{e_3}{e_1}$
Omnivariance:	$O_\lambda = \sqrt[3]{e_1\, e_2\, e_3}$
Anisotropy:	$A_\lambda = \frac{e_1 - e_3}{e_1}$
Eigenentropy:	$E_\lambda = -\sum_{i=1}^{3} e_i \ln(e_i)$
Sum of eigenvalues:	$\Sigma_\lambda = e_1 + e_2 + e_3$
Change of curvature:	$C_\lambda = \frac{e_3}{e_1 + e_2 + e_3}$

Figure 12 Local Surface Properties calculated from the eigenvalues
Source: Weinmann et al., (2014)

7. Colour information

Colour is an important property in recognizing and distinguishing different objects. The human eye is able to sense and distinguish colour by a group of cells, called

cones. Those cones are located at the back of the retina and each one of them is sensitive to a different wavelength: red, green and blue. This kind of vision is called trichromatic vision. Since colour description is subjective for every human, scientists have developed mathematical systems for describing colour, called colour spaces. Some of those systems are based on the human colour perception capabilities and others on the physical properties of colour for example the intensity values.

Acquiring point clouds can be done with or without colour. Points within a coloured point cloud hold their colour in the three dimensions red, green and blue (RGB) colour model. Therefore, exploring the different colour spaces is an important part of this research.

7.1. Additive and subtractive colour space

Additive colour scheme is based on the fact that colour is created by mixing (thus adding) different light of a particular wavelength emitted from differently coloured light sources. This is opposed to the subtractive colours where light is removed from the various ranges of the visible spectrum (Choudhury, 2014). The additive and subtractive colours are explained in Figure 13.

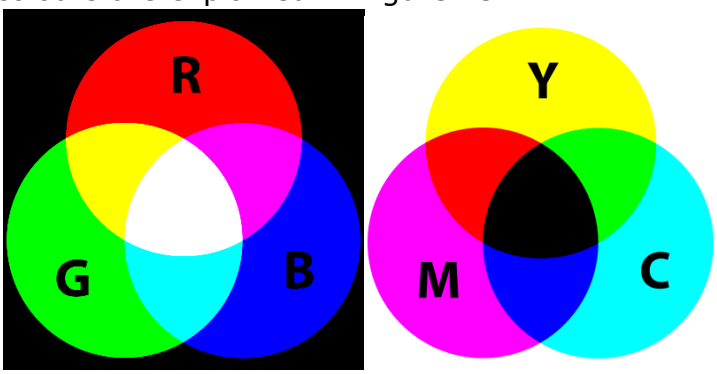


Figure 13 Left: Additive colour space, Right: Subtractive colour space

The additive colours or the RGB colour space is dependent on the intensity of colour and can be represented in a 3d cube (Cheng et al, 2001; Sapkota, 2008; Sareen et al, 2010; (Figure 14). Each corner point represents a value of red, green and blue components. All colours can be created by a unique combination of these three "main" colours.

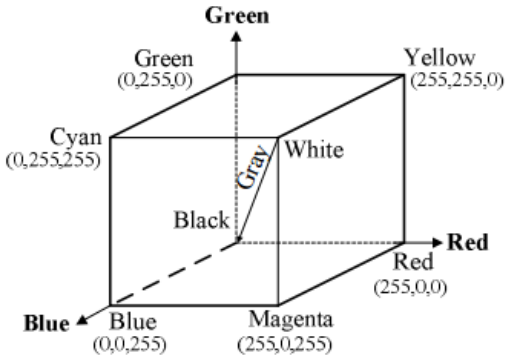


Figure 14 RGB colour space in 3d cube

Source: Sareen et al, 2010

RGB colours are mainly used in display devices, it is therefore suitable for colour display. RGB though is according to Cheng et al (2001), not suitable for colour segmentation and analysis because there is a very high correlation among the red, green and blue components therefore Cheng et al (2001) mentioned that it is impossible to see if two or more colours are similar based on their distance in RGB space.

Since the RGB colour values are not profound suitable for segmentation based on colour another colour space which facilitates easier automated comparison and matching algorithm's, an alternative for the RGB model, is necessary.

7.2. HSV colour space

An alternative to the RGB model is the HSV data model. HSV stands for: hue, saturation and value. HSV is a cylindrical model (Figure 15), whereby "hue" corresponds to the angle around central vertical axis, the distance from the axis to the side corresponds to "saturation" and the distance along the axis corresponds to lightness, or "value". This colour space facilitates the easier comparison of colour, as individual colours can be distinguished by combining 2 value ranges.

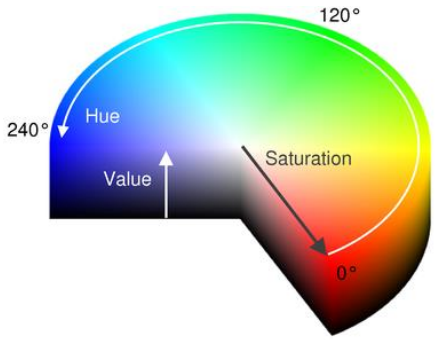


Figure 15 The HSV colour space in a cylindrical model
Source: http://bit.ly/1PeK7jx

The hue represents basic colours, and is determined by the dominant wavelength in the spectral distribution of light wavelengths. The saturation is a measure of the purity of the colour, and signifies the amount of white light mixed with the hue. The value represents the brightness of a certain colour. The colour table of the HSV colour model is show in Table 1. This table shows that, for example, black can be recognized by the "value" and red, green and blue by combining "hue" and "saturation".
In order to create a HSV value for every point from an RGB value, the following formula is used:

$$H = \begin{cases} \left(0 + \frac{G-B}{MAX-MIN}\right) \times 60, & \text{if } R = MAX \\ \left(2 + \frac{B-R}{MAX-MIN}\right) \times 60, & \text{if } G = MAX \\ \left(4 + \frac{R-G}{MAX-MIN}\right) \times 60, & \text{if } B = MAX \end{cases}$$

$$S = \frac{MAX - MIN}{MAX}$$

$$V = MAX$$

Table 1 The colour table of the HSV colour model source: mehrarodgers.files.wordpress.com

Color	Hue	Saturation	Value
Black	0° < H < 360°	0 < S < 1	V < 0.1
White	0° < H < 360°	S < 0.15	V > 0.65
Gray	0° < H < 360°	S < 0.15	0.1 < V < 0.65
Red	H < 11°, H > 351°	S > 0.7	V > 0.1
Pink	H < 11°, H > 351°	S < 0.7	V > 0.1
	310° < H < 351°	S > 0.15	V > 0.1
Orange	11° < H < 45°	S > 0.15	V > 0.75
Brown	11° < H < 45°	S > 0.15	0.1 < V < 0.75
Yellow	45° < H < 64°	S > 0.15	V > 0.1
Green	64° < H < 150°	S > 0.15	V > 0.1
Blue-green	150° < H < 180°	S > 0.15	V > 0.1
Blue	180° < H < 255°	S > 0.15	V > 0.1
Purple	255° < H < 310°	S > 0.5	V > 0.1
Light Purple	255° < H < 310°	0.15 < S< 0.5	V > 0.1

7.3. HSI colour space

The HSI is a color space that stands for hue (H), saturation (S), and intensity (I). It is a system that is more intuitive to the way the human eye perceives colour (Cheng et al, 2001). This makes this colour space very good for image processing techniques. The way the HSI colour space represents colour is presented in Figure 16 Cheng et al (2001) argues that in a segmentation algorithm the hue component can be the only parameter applied. This is especially useful when the scene has non-uniform illumination levels such as shades. Besides this, they argue, that it is computationally less intensive than the 3D RGB space.

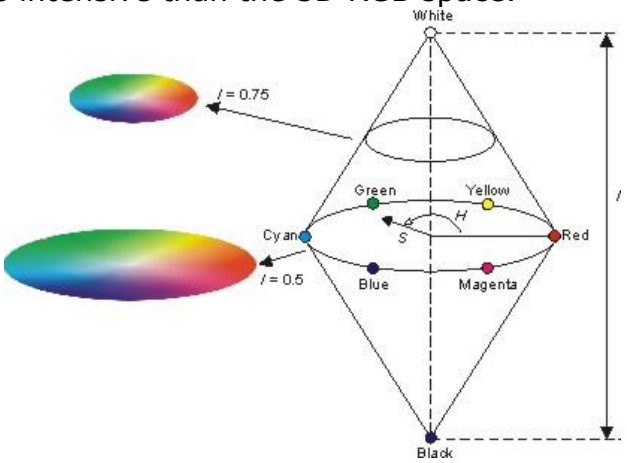


Figure 16 The HSI colour space

Source: http://bit.ly/1PeLej0

In order to create a HSI value for every point from an RGB value, the following formula is used:

$$H = \arctan\left(\frac{\sqrt{3}(G-B)}{(2R-G-B)}\right)$$

$$I = \frac{R+G+B}{3} \text{ and } S = 1 - \frac{\min(R,G,B)}{I}$$

7.4. CIE spaces

The CIE is a system that represents colour according to the human visible system and is thus based on additive colours. The colours in this space are specified by performing measurements with a spectrophotometer. Any colour in this colour space is specified by three values (X,Y,Z), where Y represents luminance. There are two colour spaces based on CIE, the CIELuv and CIELab. The conversion from RGB to CIEXYZ is a linear transformation:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

where M is a 3x3 matrix depending on the RGB colour space that is currently used. The most common RGB matrices are listed in Rucelindbloom (s.a.). From this equation the CIELuv and CIELab can be then be calculated.

For CIELab this is done with the following equation (Ford & Roberts, 1998), where X_0,Y_0 and Z_0 are the colours of standard white or illuminant.

$$L^* = \begin{cases} 116(\frac{Y}{Y_n})^{\frac{1}{3}} - 16 & \text{if } \frac{Y}{Yn} > 0.008856 \\ 903.3(\frac{Y}{Y_n}) & \text{if } \frac{Y}{Yn} \leq 0.008856 \end{cases}$$

$$a^* = 500 * (f(X/X_n) - f(Y/Y_n))$$
$$b^* = 200 * (f(Y/Y_n) - f(Z/Z_n))$$

$$\text{where} f(t) = \begin{cases} t^{\frac{1}{3}} & \text{if } t > 0.008856 \\ 7.787 * t + 16/116 & \text{if } t \leq 0.008856 \end{cases}$$

In this case the white is considered as (255,255,255) which corresponds to the d65 illuminant and the white of the average daylight. This illuminant is mostly used in the sRGB color space.

For CIELuv the transformation is done with the following equation (Ford & Roberts, 1998), where X_0,u_0 and v_0 are the colours of standard white.

$$u = \frac{2x}{(6y - x + 1.5)}$$

$$v = \frac{3y}{(6y - x + 1.5)}$$

$$u' = \quad u = \quad \frac{2x}{(6y - x + 1.5)}$$

$$v' = \quad 1.5v = \quad \frac{4.5y}{(6y - x + 1.5)}$$

$$L^* = \begin{cases} 116(\frac{Y}{Y_n})^{\frac{1}{3}} - 16 & \text{if } \frac{Y}{Yn} > 0.008856 \\ 903.3(\frac{Y}{Y_n}) & \text{if } \frac{Y}{Yn} \leq 0.008856 \end{cases}$$

$$u^* = 13(L^*)(u' - u'_n)$$

$$v^* = 13(L^*)(v' - v'_n)$$

For segmentation algorithms the CIE spaces have the advantage that the differences between colours can be assessed as easily as calculating the euclidean distance (Luccheseyz & Mitray, 2001).

8. Segmentation methods

A point cloud is a rich source of information. However, extracting information from it can be rather challenging. Therefore, in one of the first steps a point cloud segmentation process needs to take place. The idea behind point cloud segmentation is to cluster points that share common characteristics into homogeneous regions. The segmentation of point clouds originates from the segmentation of 2D images, which has been studied and understood for many decades. Segmenting point clouds has been rather difficult. Therefore, many algorithms have been developed the last years. In the next sections, some of the most commonly used algorithms are presented.

A taxonomy of 3D point cloud segmentation techniques is described in Nguyen & Le (2013): 1. Edge based methods, 2. Region based methods, 3. Attributes based methods, 4. Model based methods and 5. Graph based methods. Edge based methods are used to detect boundaries in point clouds and from those boundaries to segment regions. Region based methods take into advantage of the neighbourhood of the points, in order to divide the point cloud into regions of similar properties. Attribute based methods segment the point cloud by taking into account the attributes of the point cloud. Model based techniques use geometrical shapes like cones, cylinders, planes, spheres and they group points according to these. The points that belong to one particular shape are part of one segment. Finally, graph based methods where a point cloud is considered as a graph in which the vertices are the point cloud points and the edges connect pairs of vertices. In the following sections the most well-known segmentation methods will be shortly described.

8.1. 3D Hough transform

The 3D Hough transform (Vosselman & Dijkman, 2001) belongs to the attribute based segmentation methods and detects planar surfaces in point clouds. In the 3D Hough transform every 3D point defines a surface. Coplanar points are then defined by checking whether the surfaces intersect at a point. The plane is then defined by the coordinates of that intersection point. The next step includes verifying that the points belong to the neighbourhood. This can be done by the k-NN algorithm or by using voxels. Finally, the plane parameters are optimized by performing least squares adjustment. Many variants of this algorithms have been developed through the years. An overview of those variants can be found in Borrmann, et al. (2011). The most apparent disadvantage of this algorithm is the fact that not all objects are planar.

8.2. Random sample consensus (RANSAC)

The RANSAC algorithm is a general algorithm developed for computer vision by Bolles & Fischler (1981) that fits models to the data, even when there are gross errors present. One of the use of the RANSAC algorithm is on point cloud segmentation. The algorithm works by using minimal sets which is the minimum number of points that are needed to to approximate a shape. This shape is then tested with the rest of the points to determine how many of them are described well by this shape. The algorithm is then repeated until the shape describes an

acceptable amount of points and then the rest of points are tested from the beginning (Schnabel, et al., 2007). The advantages of this algorithm lie in its simplicity, extensibility and that it can be performed to a dataset containing a lot of noise.

8.3. Region Growing

Region growing belongs to the surface based segmentation techniques. This method tries to group points that satisfy a spatial proximity characteristic and a similarity criterion. The region growing algorithm is divided into two steps: first, detection of a seed and second, the actual growing.

A seed is a point that it is known that it belongs to a segment. The seed detection step tries to look for nearby points that belong to the same plane. Whether two points belong to the same plane can be realised by comparing the smoothness, i.e. the curvature. This is the method presented in Rabbani, et al., (2006). The authors used the surface normals and the approximation of the curvature by the residuals. The pseudocode of the method, which is also used by the point cloud library is as follows:

```
Sort the point cloud according to the curvature value and initialise an Available points
list.
Until this available point list is empty:
        initialise a region and a seeds set
        Take the point P with the minimum curvature
        Append the point P at the seeds set and the region set
        Remove point P from the available points
        For every seed, find its neighbours:
                remove the current seed form the seed list
                For each of the neighbours
                        if the angle between the normal of the seed and that point is less than
                        a threshold value.
                                add the point to the current region
                                if the neighbours curvature value is less than a threshold,
                                        add the point to the list of seeds
If the seeds set has no more seeds, the region has grown and the algorithm is repeated from
the beginning.
```

The algorithm presented controls the degree of segmentation by imposing two different thresholds: the curvature threshold and the angle threshold. It is up to the user to define the thresholds according to the structure of the scene. This, actually, is the disadvantage of the algorithm. It needs to be fine-tuned each time. Automatic estimation of the parameters is yet not possible.

Apart from purely geometrical information to be used as a similarity measure, like the normals, also, non-geometrical characteristics can be used. Nowadays, point clouds come with RGB colours and, therefore, colour similarity could be used to as a decision parameter to whether a point belongs to a region or not. Zhana et al., (2009) adjusted the algorithm of Rabbani, et al., (2006) to use colour similarity instead of normals and curvature. The roughly segmented regions are then merged into bigger ones by, again, using colour similarity as a measure. The colour space that was used to calculate the colourimetrical distance was RGB. As with all the

region growing algorithms, the thresholds need to be adjusted to the scene and, for this reason, one fits all threshold has yet to be defined.

9. Classification methods

Once the segmentation is done, features have been calculated, the next step is the classification.

Apart from the quality of the segments from the segmentation, the classification algorithm used for labelling the segments, is important for the accuracy of the different surfaces detection. The performance of a classifier is data and site specific, it is therefore critical to identify beforehand the appropriate classifier for the task (Zang et al. 2013).

Different workflows in the process of feature extraction and machine learning have extensively been researched. Although, most of these works focus on the different, separate steps in the feature extraction of point clouds, and not on the whole workflow (Weinmanna et al., 2015). Thereby, a lot of research focusses on object recognition and feature extraction, and less on the classification, semantic labelling of individual points or regions (Xiong et al., 2011, Shapovalov et al., 2010, Hmida et al., 2013, Vosselman and Maas, 2010). Even though point cloud classification is to be an interesting topic in the field of data processing, because it is a precondition of many applications (Zhang et al., 2013).

There exist two basic alternatives for classifying a point cloud (Rusu and Cousins; Ramiya et al., 2014):

1. Define different classes of surface or object types and classify each point separately (**point-based**).
2. Define different regions through segmentation and region growing based on geometric properties and then assign all points within a region to one class (**region-based**).

Both approaches have their advantages and disadvantages. But even though the point-based classification might outperform the region-based classification (Rusu and Cousins), the technique most used is the region-based classification because it gives more realistic results, has better quality through incorporating context information from neighbours, it needs less manual labor and it is less time consuming (Ramiya et al., 2014).

In Weinmann et al. (2015) and Weinmann et al. (2014) it is argued that point cloud classification approaches in literature mainly consider the different components independently from each other and that it would be most desirable to use the components of neighbourhood search, feature extraction and classification in conjunction for an improved result.

9.1. 3d feature extraction and selection

Geometric features are mainly used for assigning a semantic label, a class, to individual points or points within a region. Besides geometric features also colour values are used for classification. Chapter 5 and 6 explain the mainly used 3d geometric features for classification, the features used in this project and the chapters give examples on how to calculate those values, geometrical and colour.

9.2. Classification approaches

Once the usable geometrical and colour values are calculated a classification of points within the point cloud can be done through several approaches (Weinmann et al., 2014).

9.2.1. Point-based classification methods

Point-based classification methods are done through analyzing features of one single point of a point cloud inherent to the point cloud such as distance from the ground or planarity.
Algorithms belong to the point-based classifications are the labelling algorithm and the multi-scale curvature classification (Zhang et al., 2013). These classification methods are in general and also in this case mostly done on elevation, are time consuming and need a lot of human interaction (Ramiya et al., 2014). **The labelling algorithm -** Labels each point based on the elevation and slope. the elevation and slope of each class needs to be mathematically defined from training data (Sahn and Sampath, 2005). **The multi-scale curvature classification (MCC) -** Is an algorithm that operates through assigning classes to points based on their curvature value. Points belong to different classes when one point exceeds a certain threshold and another point does not (Tinkham et al., 2011).

9.2.2. Region-based classification methods

Region-based classification algorithms are to classify pre-defined regions, segments, or are to classify points while instantly growing a region with the points with the same geometrical properties. A region-based classification is especially effective on urban area's where many edges and thus different segments are found (Zhang et al., 2013). Several region-based classification algorithms which classify points and create regions are the Associative Markov Networks and the Conditional Random Fields. Region-based classification methods which classify pre-defined segments are the K-Nearest Neighbour Classifier, Naive Bayes and the Support Vector Machine. Also the assignment of segments to classes can be done through a decision tree or bayesian probability.

The Associative Markov Networks (AMN) - is a graphical model for point cloud classification which models the spatial interactions within the point cloud. The associative markov network is an undirected graph in which the nodes are represented by Nrandom variables $y_1, \ldots, y_N$. An edge defines the interaction between variables of surrounding nodes. These variables are complementary to the predefined classes, labels. Each node and each edge has a potential, these potentials reveal that for a given point/segment some classes, some labels, are more likely to be assigned that others (numerical score). The conditional probability is expressed as:

$$P(y|x) = \frac{1}{Z}\prod_{i=1}^{N}\varphi(x_i, y_i)\prod_{(i,j)\in E}\psi(x_{ij}, y_i, y_j)$$

(Wang et al., 2012, Kabali et al.,). An disadvantage of the AMN is that it is restricted through the assumption that features of nodes are independent from conditions of other nodes.

The Conditional Random Fields (CRF) - Provides a framework for classification represented by a graph of nodes and edges. Each point within a point cloud can be considered a node, edges link adjacent nodes. in the case of labelling points of a point cloud each 3d point will be assigned a label based on the training data. The posterior of the CRF is modelled by:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}\exp\left(\sum_{i\in\mathbf{n}} A_i(\mathbf{x}, y_i) + \sum_{i\in\mathbf{n}}\sum_{j\in\mathbf{N}_i} I_{ij}(\mathbf{x}, y_i, y_j)\right)$$

Where Ni, which is the neighbourhood of node ni, is represented by the edges linked to this node. The potential is computed for each node and each edge by: $Ai\,(X, yi)$and $Iij\,(X, yi, yj)$. The potential of each node links the data to the class labels where the potential of each edge represents the dependencies of a node on its adjacent nodes. $Z\,(x)$ turns the potentials into probability values. (Niemeyer et al., 2012).

The K-Nearest Neighbour classifier (KNNC) - Means finding the nearest neighbour of a known point in instance space and labelling the unknown instance with the same class label as the known point. This algorithm does not make assumptions on the distribution of the data. Training data for a KNNC needs to consist of a set of vectors, points, which have certain geometrical properties and colour values and class labels associated with these vectors, points. The testing data is unlabelled data. A simple scenario: $x$is the point to be labelled. the closest point to $x$is point $y$which has the label of *Grass*. If point $x$has the same geometrical properties as point $y$it will get the same label als point y; *Grass* (Sutton, 2012).

The Naive Bayes - Is a simple technique based on Bayes theorem. It is a classification with strong autonomous assumptions between features. It does not consider possible correlations between features. The Naive Bayes is a probabilistic classifier that finds for every point the probability to be classified in a certain class, based on the prior and the likelihood. It only requires a small amount of training data to estimate parameters necessary for the classification.

To predict the class $c \in C = \{c_1, \ldots, c_m\}$ of an input sample (a point cloud segment) x with certain attributes(features) $(a_1, \ldots, a_n)$ the largest posterior probability is calculated:

$$\arg\max_{c_i \in C} \Pr(c_i \mid x).$$

Then the independent condition is written as:

$$\Pr(c_i \mid x) \propto \Pr(c_i)\Pr(x \mid c_i) = \Pr(c_i)\prod_{j=1}^{n}\Pr(a_j \mid c_i).$$

These formulas estimate the probability of a class and attributes for a certain input sample. The predicted score of a certain class is derived from:

$$\varphi_i = \ln \Pr(c_i) + \sum_{j=1}^{n} N(a_j, c_i) \ln \Pr(a_j \mid c_i),$$

The result is determined by the class with the highest predicted score, a point cloud segment is labelled as the class with the highest predicted score for this segment (Wu and Lin, 2011).

The Support Vector Machine (SVM) - Is a widely used classification method within the remote sensing community which considers training samples $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ where one sample $x_i \in R^n$ belongs to two classes. The training samples are non-linearly mapped to a high dimension feature space. The SVM finds a hyper-plane, a linear decision surface, which divides the set of training data in a way where all the points with the same label are on the same side of the hyper-plane. The idea is that the SVM finds the most optimal hyper-plane $w^T z + b = 0$ in a high dimensional feature space (Figure 17)(Cortes and Vapnik, 1995; Samadzadegan et al.).

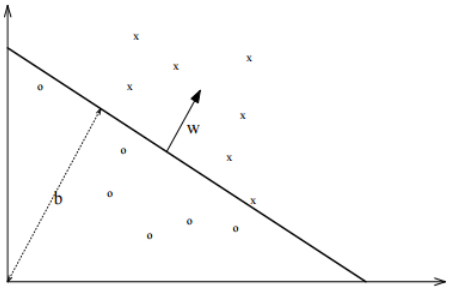


Figure 17 Hyper-plane
(Cristiani, 2001)

The SVM is in general a linear classifier, but in order to classify non-linear data the solution can be to map the data into a feature space where they are linearly separable (Figure 18).

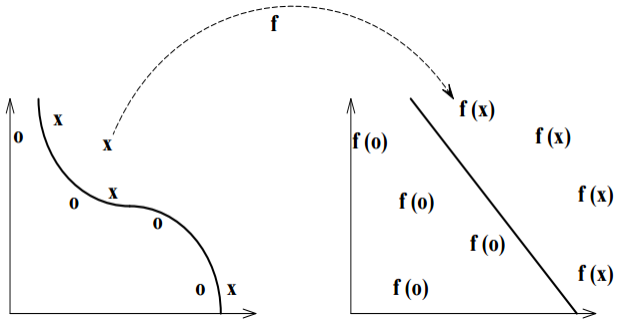


Figure 18 Non-linear data in linear feature space
(Critstiani, 2001)

Around each hyper-plane a mistake bound, a margin, exist between the hyper-plane and the nearest point (Figure 19)(Cristiani, 2001). This is used when more than one hyper-plane can fit between the different classes. The hyper-plane which leaves the maximum margin from both classes is set to be the optimal hyper-plane (Figure SX)(Cortes and Vapnik, 1995).

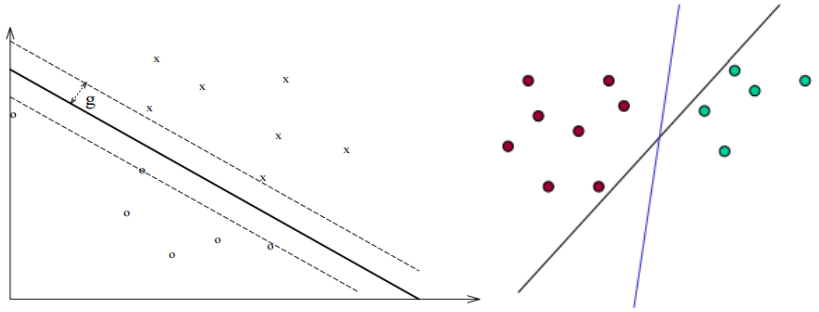


Figure 19 Left: Hyper-plane and margin, Right: finding the optimal Hyper plane (Cristiani, 2001)

Furthermore, SVM's operate in a kernel induced feature space. Important in a SVM is the kernel parameter, the effectiveness of the SVM depends on this selection of kernel, a soft margin parameter and the kernels parameter. This last parameter defines the structure of the high dimensional feature space. Usually these parameters are selected by trying a finite number of values and select those which provide the least error. A common choice is a gaussian kernel which has just one parameter γ. Each combination of a kernel parameter and a soft margin parameter is checked using cross validation. The classification then is done using the selected parameters (Cortes and vapnik, 1995; Samadzadegan et al.).

Multiclass SVM: Since the SVM is a binary classification method but for remote sensing applications several classes are usually of interest. To solve this problem this multiclass problem is to be reduced to multiple binary classification problems. It is possible to build binary classifiers which divide the data between one label and the rest, as it is possible to build binary classifiers which divide the data between every pair of classes (one-versus-all and one-versus-one) (Figure 20). The one-versus-all method with an X amount of classes composes X SVM models which can categorize the samples into samples from one class and samples of the remaining classes (Samadzadegan et al.).

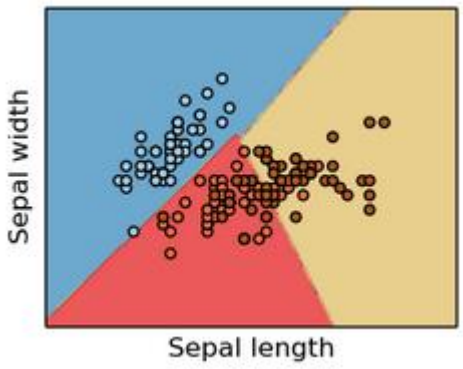


Figure 20 Multiclass SVM with linear kernel

The decision tree - Uses a tree as a predictive model, whereby observations of the geometrical values of a point cloud segment lead to the conclusion about this segment. It uses a classification scheme to do this, a hierarchical structure that is accompanied by descriptive information. Algorithms to create a decision tree work top-down, where each geometrical or colour value separately and one by one

guides the point cloud segment through the scheme, eventually classifying it (Figure 21). The classification scheme gives the segment and its class a meaning. (Rokach and Maimom).

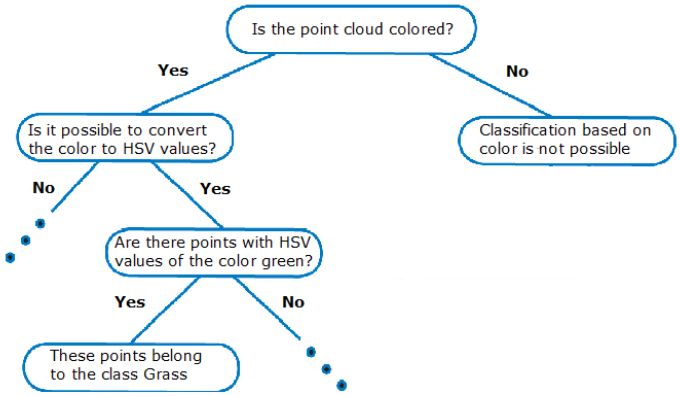


Figure 21 Decision Tree for point cloud classification

The Bayesian probability - Is used to evaluate the probability of a hypothesis. The Bayesian probability specifies some prior probability, which is then updated in the light of new, relevant data (evidence), in our case geometrical properties of points and their surroundings. The Bayesian interpretation provides a standard, pre-defined set of procedures and formulae to perform this calculation (Wikipedia (3), 2015). In Bayesian probability, a weight is given to each variable (or attribute) and added up or multiplied by a pre-defined factor. The end score assigns the actual class to the pre-defined region, segment.

Chapter 4: Methodology

1. Methodology

This chapter will define the general research strategy that outlines the way in which this research is undertaken, also it will give requirements and parameters necessary for a successful end product. The type of processing that is needed is based on the process of how to get to the end product. This process is based on literature, the literature and the process is explained in the literature overview chapter.

The processing of the point cloud begins with filtering and continues with indexing, segmentation, classification and labelling. All processing is as defined by the requirements of the project performed on the raw point cloud data. This means that no other geometries apart the points was used. For each step in the process of processing the raw point cloud requirements are set up which will be made clear in the next paragraphs.

1.1. Point cloud acquisition and generation

A point cloud of the faculty of architecture and its surroundings was planned to be acquired (Figure 22). This acquired point cloud is a product of the company Cyclomedia. Cyclomedia's common products are cycloramas and a web-viewer to use. However, since recently Cyclomedia is exploring 3D geo-information and point clouds. The products they delivered to our team are two different kinds of point clouds, both acquired with the mobile device mounted on a car.

The first point cloud, is a point cloud based on 360° high definition panorama images. The point cloud is obtained by a two-step approach. "First and initial point cloud is created by dense stereo matching based on a variation of the semi-global matching (SGM) in combination with Census transform and colour information. SGM uses pixel-wise matching (Hirschmüller, 2008). The point cloud is then filtered and turned into a triangle mesh using a method based on Labatut et al, (2007). Vertices from the resulting mesh are the points in the LAZ files" (Beers, 2015). The accuracy of the obtained point cloud depends on the intersection angle and the accuracy of the panorama positioning. According to Beers (2015), Cyclomedia does not currently track accuracy information for each point in the pipeline. The processing time for the construction of this point cloud is approximately three hours.

The second point cloud is from a prototype they recently developed, this point cloud is obtained by a LIDAR laser scanner. This new generation of Cyclomedia's mobile mappers is equipped with the Velodyne HDL 32 lidar scanner, that has 32 channels and catches 700,000 points per second with ± 2 cm accuracy. The Lidar scanner has a 360° horizontal field of view (Velodyne, 2015). The output of the device can be described as polar coordinates in 3D; 2 angles and a distance. By using the location and orientation of the IMU and the position derived with GNSS

receivers, together with the relative position and orientation of the LiDAR device with respect to the IMU, the polar coordinates in the device frame can be transferred into a Cartesian world-frame.
The Lidar point cloud does not contain colours in its raw format. In order to apply colour to the point cloud, the theoretical process is to "use the panorama closest to the LIDAR ray origin to colour the resulting point. There are a few additional rules to take the second or third best option if the point would be occluded by the recording vehicle itself. Otherwise, the algorithm currently makes no attempt to detect or mitigate the effects of occlusion. From this it is also clear that if the lidar points or origins are in the wrong location, the colouring can give the wrong results." (Beers, 2015).

Because 3D point clouds are still prototypes of Cyclomedia there is no cost indication of the acquisition. Both point clouds are delivered with colour, colour obtained from the high definition images. The only requirement that needs to be met for the acquisition of the point clouds is based on the weather. When raining the acquisition won't be reliable, inaccurate, since water drops can reflect the laser which will give "floating" points, point that are not there in reality, in the point cloud. The point cloud was obtained on a day with rainy weather conditions due to the time constraints of the company.

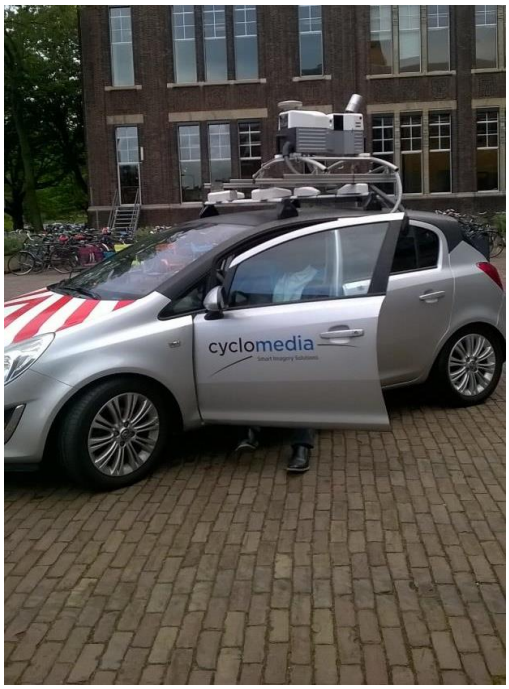


Figure 22 The data acquisition at the faculty of Architecture

1.2. Point cloud pre-processing

Both point clouds, in pieces delivered as different laz-files, contain points representing the whole faculty of architecture building and its surroundings. The research questions however only applies on a specific part of the point cloud; the road and its surrounding surfaces. Therefore the ground points must be filtered from the non-ground points. Filtering the ground and non-ground points must

answer to some requirements. These requirements are listed in the following subparagraph.

1.2.1. Filtering parameters and requirements

- The filtering must delete most non-ground points in order to cope with the data, at least the filtering process must contain all the ground points. So the majority of the non-ground points must be filtered out
- Ground points and non-ground points must be clearly defined.
- The filtering must be based on these clearly defined ground and non-ground points.
- The adjustment of these ground and non-ground parameters into the filtering must be straightforward.
- Since several techniques for filtering exist, the best way for filtering the point clouds in this research must be defined.

1.2.2. Predefine the relevant classes

To classify the points and give the correct label to each point first the relevant classes that needs to be labelled should be identified. Having a specific application in mind is very important when designing a process. In this case road extraction and materials focused on street level are examined.

- The defined classes must be distinguishable.
- The classes must cover important classes for the named applications.
- All possible ground surfaces (around the faculty of architecture) must be defined.

1.3. Point cloud processing

Now that the point cloud is filtered to non-ground points and ground points the ground points can be labeled according to the predefined classes. Labelling the ground points can be done by two methods, either point based labelling or either region based labelling. To test whether method is better for labelling ground classes both, point based labelling and region based labelling, will be executed and compared. Point based labelling will be done by using colour values to distinguish classes and samples will be made to classify each point to give the correct label to the points.

Region based labelling acquires a more detailed method. For this specific method the point cloud needs indexing to calculate the geometrical properties that will likely distinguish the different classes. Further region growing on geometrical properties and colour values that distinguish the classes will be used. To finally classify the obtained regions two choices are available:

- Supervised: The regions will be classified based on samples, taken manually
- Unsupervised: The regions will be classified according to the same geometrical properties and colour values

Both these choices will be tested and examined on the regions and these methods will be compared to see the advantages and disadvantages.

1.3.1. Indexing parameters and requirements

- The necessity of indexing the point cloud must be defined.
- The used indexing method must be clearly defined and underpinned.
- The indexing must be a fast process, even for point clouds containing millions of points.
- With the defined indexing method it must be possible to define nearest neighbours for every point since all further processing is based on these nearest neighbours.
- It must be possible, using the indexing structure, to calculate/define the geometrical properties of the point cloud to increase the ease of classification.

1.3.2. Segmentation parameters and requirements

- The segmentation method must be clearly defined using region growing.
- Segments must be obtained that only contain one class.
- Most segments must contain multiple points, otherwise those aren't segments.

1.3.3. Classification parameters and requirements

- The classification method(s) must be clearly defined.
- Samples, training data, of these defined ground surfaces must be taken from the point cloud.
- The classification must be executed according to the training data of the predefined classes.

1.3.4. Labelling parameters and requirements

- Points must be enriched with semantics.
- Spatial labels are derived from the context, in our case the built environment
- Labels must be clear for the users or for a specific use case.

1.4. Validation

An important phase of the method, the project process, is validation. Validation is a process to test and control the steps of the project process. Validation is in this research needed to evaluate the reliability of the used method. The classification of points is an important matter within this research. Therefore, the assessment is performed within this process. A common way of assessing classification processes is by creating a confusion matrix. The confusion matrix is a table that summarises the correctly assigned points to each class in combination with the points misassigned to different classes. The errors presented in the confusion matrix are the errors of omission and commission. The errors of omission represent the points that should be of a particular class but are classified wrongly. The errors of commission, on the other hand, represent points that belong to another class but are classified as belonging to another.

The semantic labelling and all its needed processing steps will in this research be done on just a small part of a much larger point cloud. Even if the used labelling method works on this small point cloud this does not guarantee that this method

will work on other parts of the larger point cloud of point clouds in general. Therefore, for the validation of the method it is necessary to test the used method also on other parts, on other point clouds.

1.5. End product

The end product of this research is to be a semantically enriched point cloud which holds information that can be used for road extraction and surface infiltration capacity analysis. There are two possible options for end products to be delivered to the end user. Either store all the points in one laz/xyz-file with the classification, or store each classification in separate laz/xyz-files. Both these methods have their advantages and disadvantages. If all the points are stored in one file the end-user has to download the whole file, even if the end-user only wants to see a specific category. Downloading the whole file will cost time and the interaction with massive point clouds is slow. On the other hand, if a user needs to download a lot of small files, one of those files can be easily lost. File management is therefore required. Because our application is made for specific use cases it is assumed to be better to store each class of the point cloud in a separate file.
For the visualisation of the end product an open source program like Cloud Compare is used. Optionally the files can be loaded on a server and the point cloud can be viewed within a web browser from any location. The potree library (http://potree.org/), making use of webgl, can be used for this.
Parameters and requirements:

- The end product must be a semantically enriched point cloud.
- The method used to get to the end product must be valid and applicable on other (terrestrial) point clouds.

Chapter 5: Implementation

1. The acquired point cloud

The Stereo Point Cloud of the faculty of Architecture and surroundings was delivered to the team on 16/9/2015. As can be seen from Figure 23, the point cloud is focused on the street level. The point cloud is however not without problems. It is possible that for specific areas, especially with high vegetation, the building facades are occluded. In addition to that, the roofs are in their majority missing. In the case study examined by this group, missing roof or façade information does not pose any issues as they are filtered out in the next stage.

Figure 23 The obtained point cloud

2. Non-ground filtering

Since working with big data is not an easy issue even with today's computer power, for testing purposes a small segment of the whole stereo point cloud is extracted at the beginning and will be further used to test the classification algorithms.

To obtain a subset of the whole stereo point cloud around the faculty of architecture, the lasclip tool from the LAStools suite is used to process all the separate files at once. Lasclip takes LAS or LAZ files and a polygon as input (See Figure 24, blue lines), the files intersecting with the polygon are outputted. In this way only the points that are inside this polygon are stored and later merged to one file. The used polygon to make a selection is stored as a shapefile, and looks as follows in Well Known Text (WKT):

POLYGON((594080.076999 5762420.870001, 594113.759998 5762465.878001, 594086.458999 5762487.190001, 594045.535999 5762449.965003, 594080.076999 5762420.870001))

The coordinates to make the polygon are obtained by selecting points in the total scene. The lasclip command line that is used is:

lasclip -lof [selection of the files] -merged -poly "Path\ClipSegment.shp" -odir "Output directory" -o "StereoMergedClip.laz"

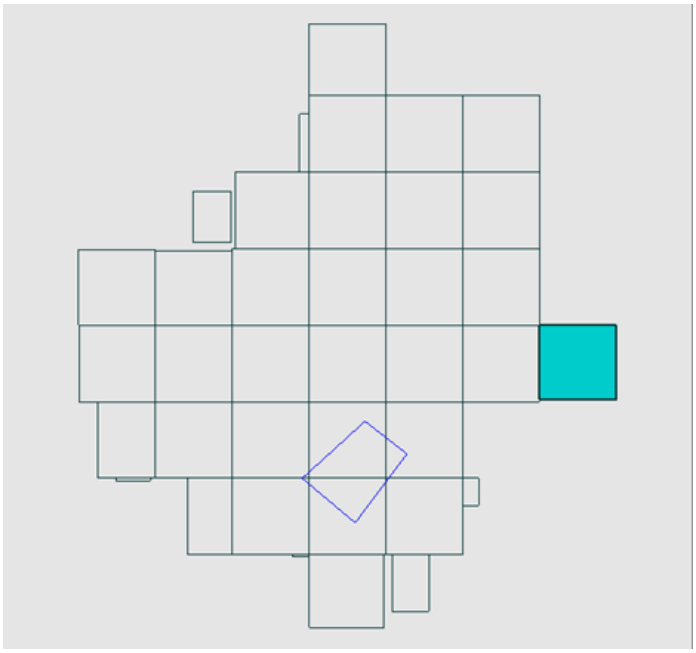


Figure 24 All the used las-files for input (boxes) and the polygon intersecting these files(blue lined polygon).

The result of this process is a file containing 3,413,935 points (Figure 25). This file contains the whole scene with the buildings, cars, trees, road etc. However, not all information is relevant for all applications. In the case examined in this project, the focus area lies on the ground. This means that the non-ground points (buildings, trees, cars) need to be removed from the dataset. To classify the previously clipped point cloud to ground points and non-ground points the lasground_new tool from the LAStools suite is used. This tool classifies the LAS or LAZ files imported into ground points and non-ground points by giving them a label (2 for the former and 1 for the latter) in each LAS file row.

Figure 25 Clipped stereo point cloud, 3,413,935 points.

The following command is used to make this classification:

lasground_new -i "Path\StereoMergedClip.laz" -offset 0.2 -odir " Output directory " -o "StereoGroundclassification.laz"

where, **offset** represent the maximum offset in meters up to which points above the current ground estimate get included in the filtered output, **odir** the output directory and **o** the name of the output file. The command line output is shown in Figure 26, where it is stated that 1855993 points are ground points. This as explained before just adds a label to the current point cloud and does not actually create separate files.

```
processing file 'C:\Users\Tim\Desktop\StereoMergedClip.laz'.
horizontal units are meter and vertical units are meter. nature mode.
reading 3413935 points. step is 25 m, sub is 5, spike is 1+1 m, and offset is 0.2 m ...
took 13.568 sec. finding initial ground points ...
took 0 sec. generating initial ground estimate ...
took 0.047 sec. refining ground ...
took 0.062 sec. integrating points 0.2 above ground ...
took 11.068 sec. outputting ...
took 4.896 sec. 1855993 points classified as ground.
done with 'C:\Users\Tim\Desktop\StereoGroundclassification.laz'. total time 29.641 sec.
```

Figure 26 The command line output for the filtering

Now that the classification is done the file can be split according to this classification using the lassplit tool from the LAStools suite. The lassplit splits the the inputted LAS and LAZ files according to some criteria defined. The command used to perform the splitting according to the ground and non ground classification is:

lassplit -i " Path \StereoGroundclassification.laz" -by_classification -odir " Output directory " -o "StereoGround.laz"

Which resulted in the division of the point cloud shown in Figure 27 & Figure 28.

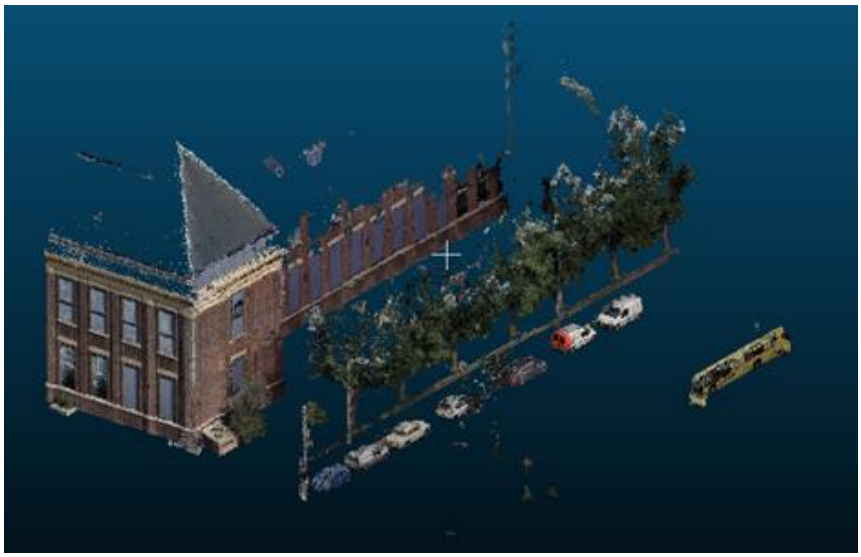


Figure 27 The processed non-ground points: 1,557,942 points

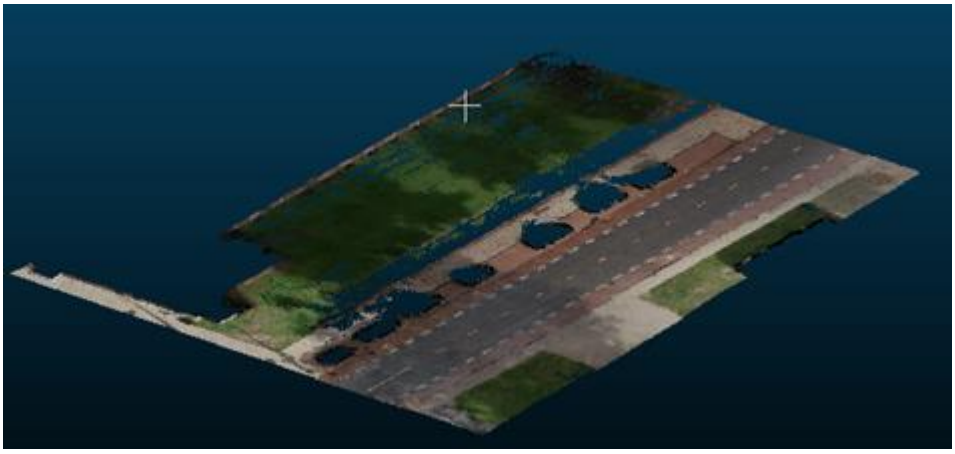


Figure 28 The outputted ground points: 1,855,993 points.

The same process can be used to filter the LiDAR point cloud. Exactly the same polygon is used to clip the whole LiDAR point cloud (Figure 29). With the following LAStools command:

lasclip -lof [selection of the files] -merged -poly "Path\ClipSegment.shp" -odir "Output directory" -o "LidarMergedClip.laz"

The output is shown in Figure 30. In this figure it is visible that the point clouds are not aligned, because the same wall is on two positions. Therefore, the focus will lie on the stereo point cloud and not on the LiDAR point cloud for this project.

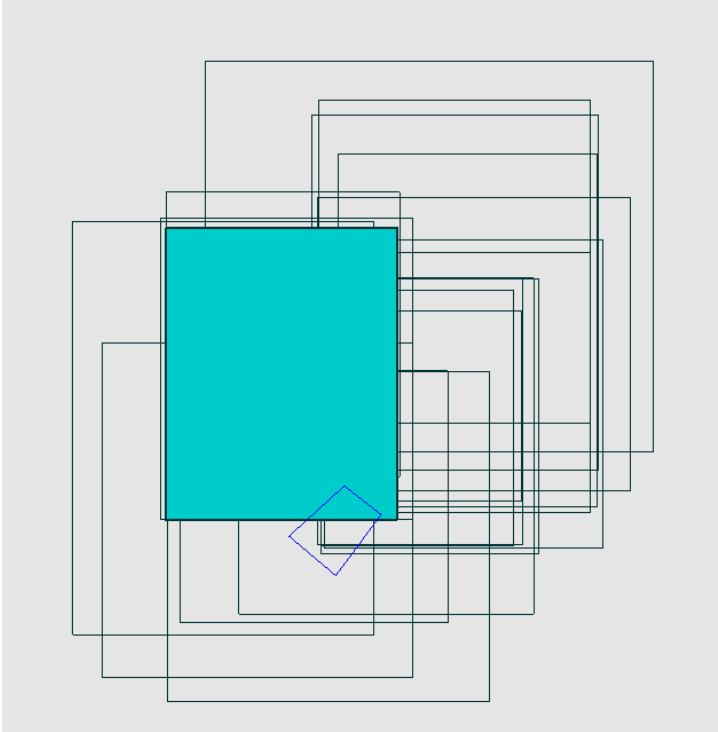


Figure 29 All the used LiDAR-files for input (boxes) and the polygon intersecting these files (blue lined polygon).

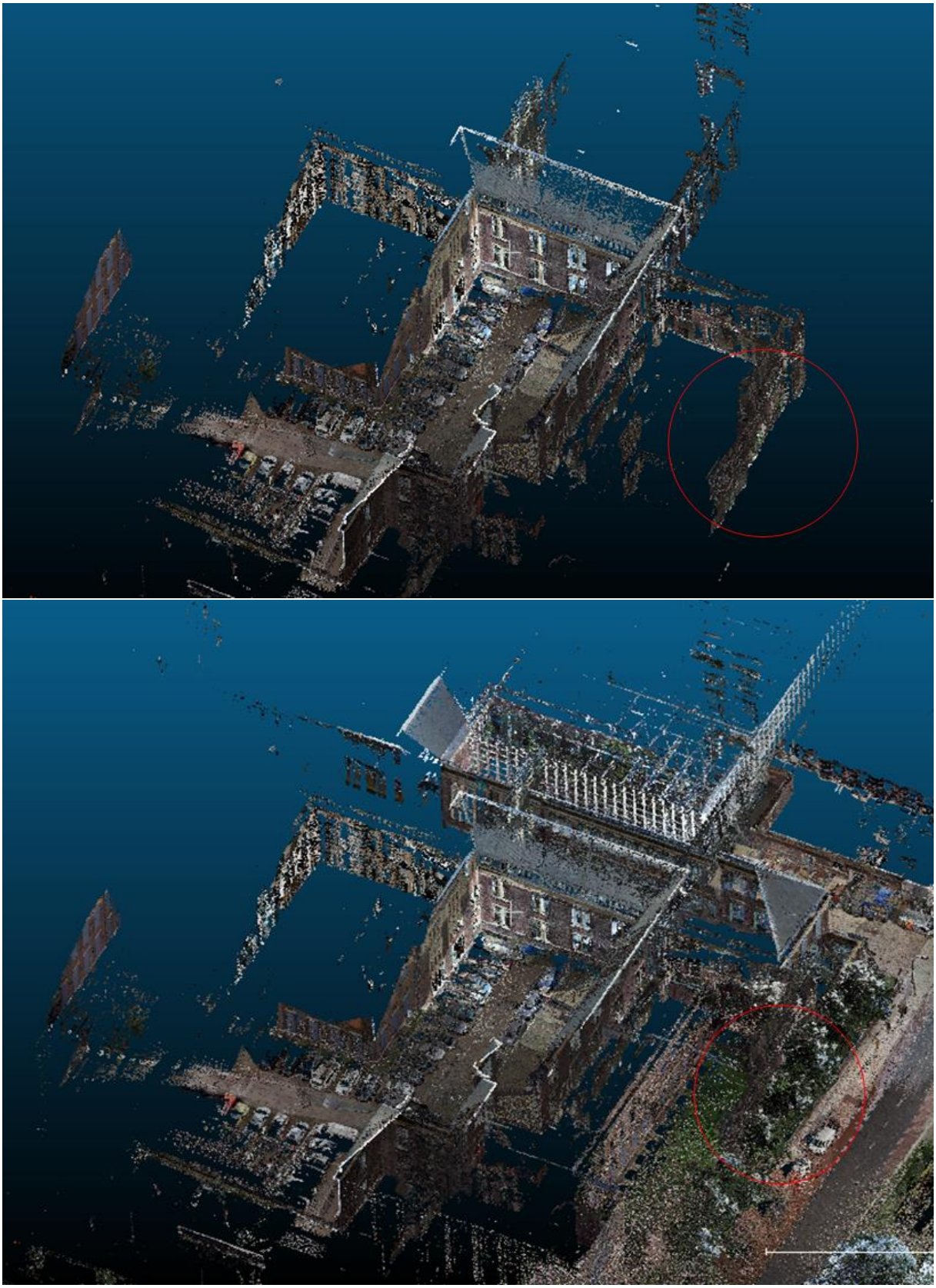


Figure 30 The clipped LiDAR point cloud.

3. Importing the data

The raw data provided by Cyclomedia are given in the .las or .laz format. The data provided are needed to be loaded inside Python. Two ways of doing this were investigated. Fist way includes transforming the las file to .xyz format and loading it as an ASCII file inside python. The second way includes using libLas, a library for reading and writing LAS format. In order to test the performance, the two readers were run using timers. The option used is the one that provides best performance.

3.1. xyz reader

The code presented in Table 2 the python implementation for the .xyz file reader. When running the file with a point cloud of 3,332,161 points the output in the correct list format is given in 11.7 seconds.

Table 2 The .xyz reader

```
def unpack_xyz(infile):
    points_xyz = []
    for each in infile:
        values = each.split(' ')
        xyz = (float(values[0]), float(values[1]), float(values[2]))
        points_xyz.append(xyz)
    return points_xyz
```

3.2. Liblas

The code presented in Table 3 shows the python implementation for the .las file reader. The file module of liblas is imported and running with the same point cloud of 3,332,161 points the output in the correct list format is given in 56.5 seconds.

Table 3 The .las reader

```
from liblas import file

def lasreader(fh):
    infile = file.File(fh, mode='r')
    coords = []
    for i in infile:
        coords.append((float(i.x), float(i.y), float(i.z)))
    return coords
```

It can be thus be understood that using the .xyz file significantly increases the efficiency of the code. However, this format is not the most widely used one and transformation tools from .las to .xyz are needed.

4. Data structures

The algorithms developed for the classification of the point cloud depend on two data structures. A data structure is needed in order to organise the information of the unstructured point cloud provided in a relevant way. The data structure is composed of two objects: a point object and a region object (Figure 31).

A point object is an object that has the following attributes: index, coordinates, CIELab values, curvature, normal, neighbours, region, and RGB colour. A point is related to another point by having neighbours. The number of neighbours of a point is fixed and is k. In addition to that, a point can be the neighbour of zero or more other points.

The region object is an object that has the following attributes: region number and list of points. Between a point and a region there is a composite relationship as a region cannot exist without a point. In addition to that, a point belongs to exactly one region and a region contains at least one point.

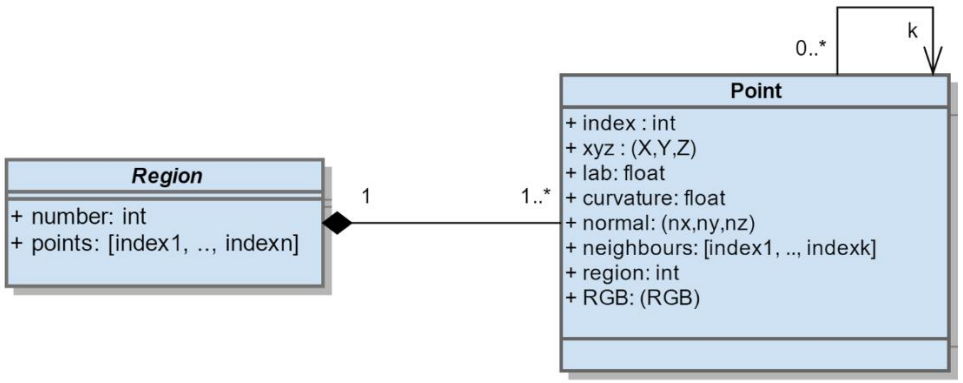


Figure 31 The UML diagram of the data structure used for the classification of the point cloud

5. Spatial index

For quick access of the point cloud points a spatial index is applied to the process. The kD-tree applied to this case of 3D point sets is a 3D tree. A kD-tree, in comparison to an octree, can provide a balanced data structure and it is optimised for the k-nearest neighbours search.

The kD-tree used for the process developed in this project is the cKDTree algorithm of the scipy.spatial module. This algorithm is an implementation of the algorithm of Maneewongvatana & Mount, (2001) in C and therefore provides faster nearest neighbours search. The same neighbourhood search without an index takes couple of hours since all the points (1.8 million) need to be traversed each time the k nearest neighbours for a point need to be found. With the k-D tree algorithm the construction and the neighbour search of all the point takes 20 seconds.

6. Colour spaces transformations

Colour information is an invaluable information for humans. Humans can interpret colours and through that assign semantics in milliseconds. Once a human looks at Figure 32, he/she immediately infers, the road, the grass, the tiles etc. Humans realise colour in the RGB colour space. For computer vision, however, the RGB space has a disadvantage, colour difference is not uniform. This means that two

very similar colours cannot be always be distinguished numerically as similar by comparing their RGB Euclidean distances. Therefore, different colour spaces needed to be evaluated for their value in segmenting the point cloud.

Figure 32 The original point cloud in the RGB space

6.1. RGB to HSV

The HSV colour space is implemented in the python environment by utilising the colorsys module. The resulted transformation in the point cloud, except the saturation value, is depicted in the Figure 33 and Figure 34. The saturation is hard to visualise as it should be represented by different colour according to the hue angle.

From the different hue values in Figure 33, there can be made a distinction between the classes grass, cycle path and, road and tiles. We, also, can observe a very strange situation on the road. This could be because of the existence of shade that makes the colours not homogeneous for this class.

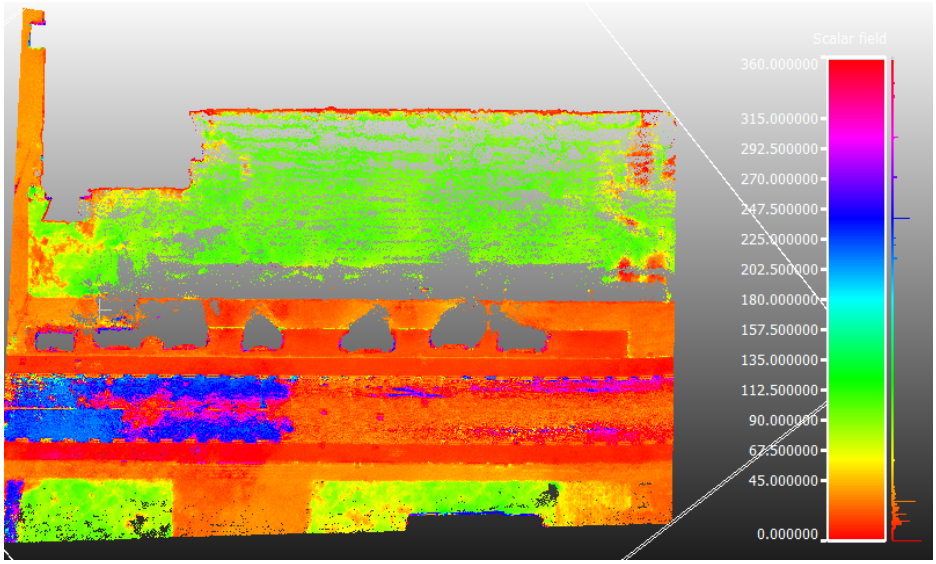


Figure 33 The H value of the HSV colour space coloured differently according to the angle

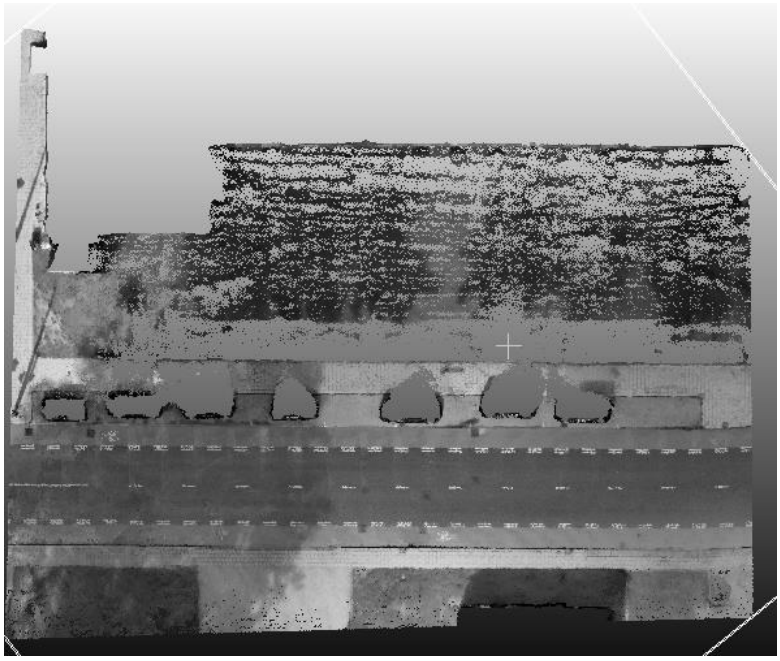


Figure 34 The V value of the HSV colour space coloured in greyscale

The V represents the brightness. In Figure 34, we can observe that the classes cannot be easily be distinguished as the road has the same grey value as parts of the vegetation. The same is observed for the cycle path and parts of the pavement mostly because of the existence of the shade.

6.2. RGB to CIELab

The transformation from RGB to the CIELab is implemented in the python environment. The transformation is performed by applying the equations presented in section 6.4. with the value for the absolute white to be taken as (255,255,255) which corresponds to the d_{65} illuminant.

The resulted transformation is depicted in the following figures (Figure 35, Figure 36, Figure 37). In Figure 35 one can very easily distinguish the tiles that are represented in the lightest grey. Vegetation has on the other hand the darkest grey value.

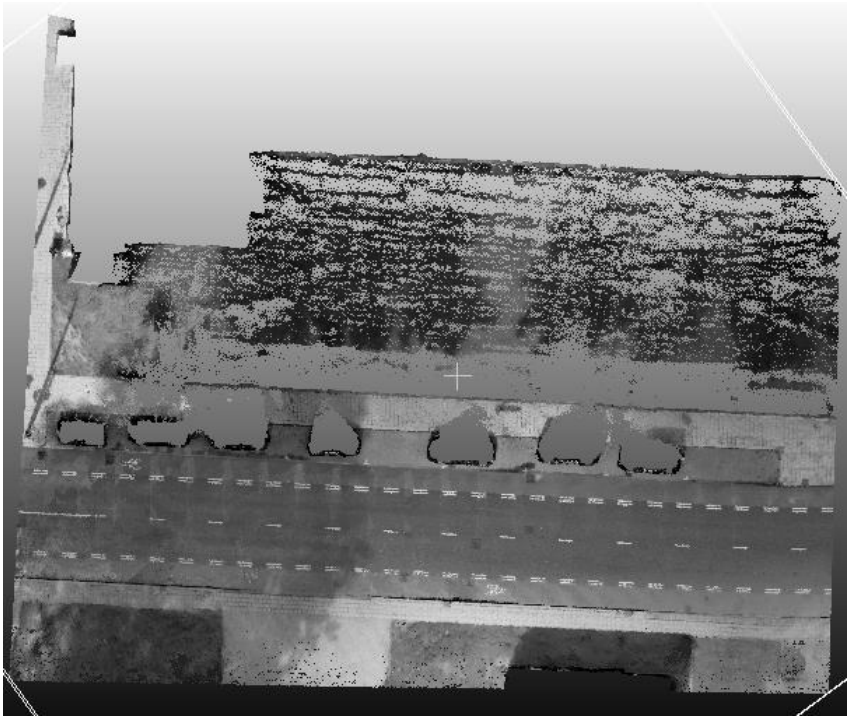


Figure 35 The L axis of the CIELab colour space represented as grayscale

In Figure 36, on the other hand, the differences become even more apparent. One can distinguish the grass, the bike lane, and the asphalt road and tiles. Distinction between asphalt road and tiles is however not possible in this case.

Finally, in Figure 37 the grass is again easily distinguishable. The other classes seem more or less a mixture of colours and therefore would be hard to distinguish with this value.

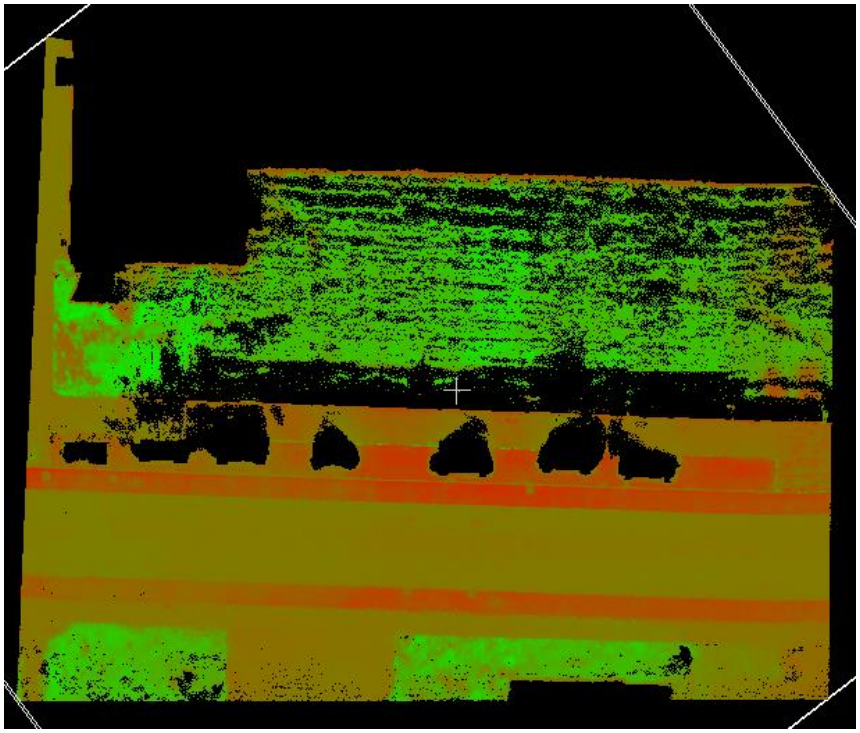


Figure 36 The a axis of the CIELab colour space represented from green to red

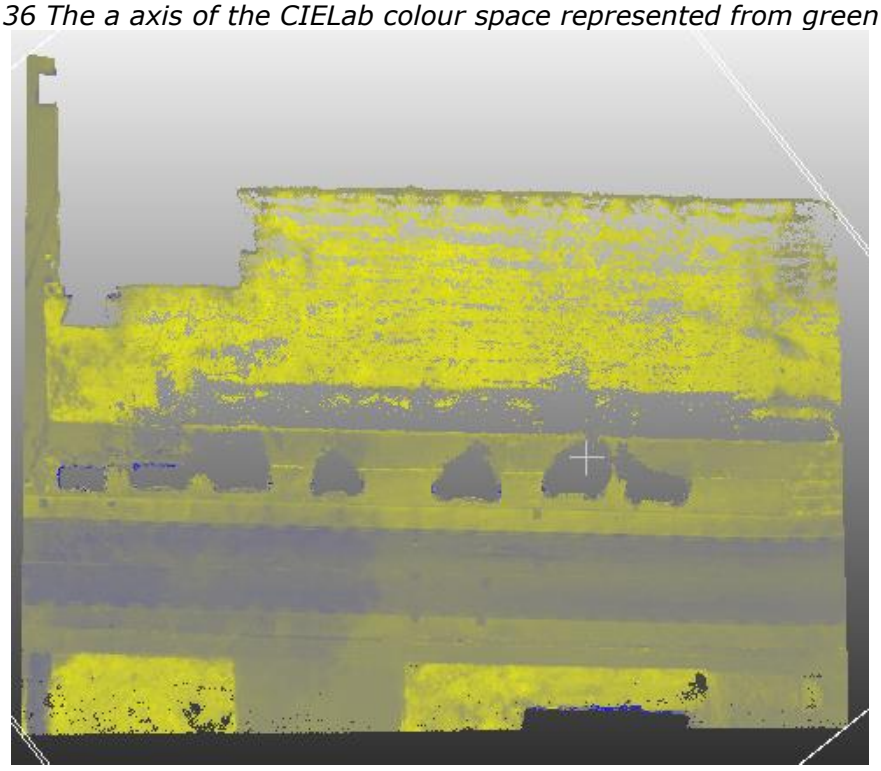


Figure 37 The b axis of the CIELab colour space represented from blue to yellow

6.3. RGB to CIELuv

For the transformation to the CIELuv space, the python-colormath module was imported into python. The three components (L,u,v) for the point cloud is depicted in Figure 38, Figure 39, Figure 40. From each component different objects can be distinguished. In Figure 38, as in the case of the CIELab, the vegetation and the tiles are the most easily distinguishable. In Figure 39 and Figure 40, the different

classes are no longer easily distinguishable. Therefore, the CIELuv will not be used for the classification.

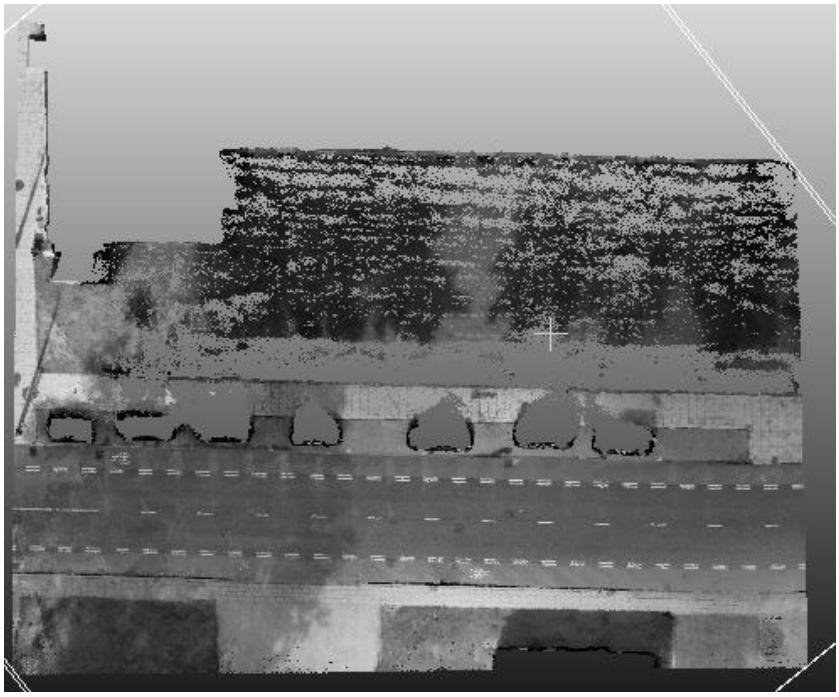


Figure 38 The L axis of the CIELuv colour space represented as grayscale

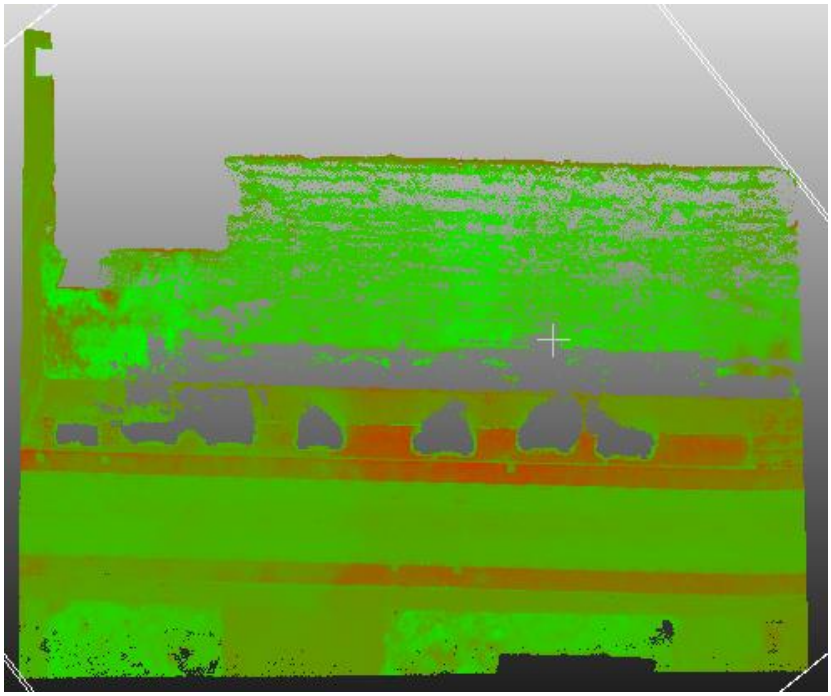


Figure 39 The u axis of the CIELuv colour space represented as grayscale

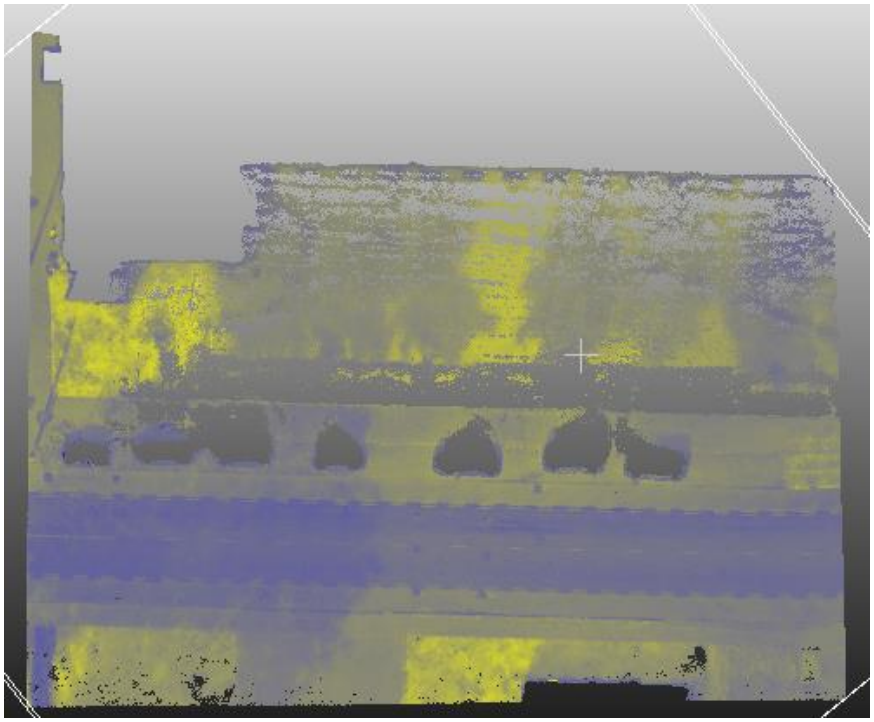


Figure 40 The v axis of the CIELuv colour space represented as grayscale

7. Training samples

The method we use to semantically label the point is best captured by the definition of supervised classification, which is an algorithm that uses a known dataset (called the training dataset) to make predictions. The aim of supervised, machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty (Mathworks, 2015). In other words, supervised learning is the machine learning task of inferring a function from labelled training data. The training data consist of a set of training examples (Wikipedia (A), 2015). In order to create our semantically labelled point cloud, we extracted our training data as separate different classes. The following classes are extracted from the point cloud: grass, tiles, cycle path (red asphalt) and road (black asphalt), as shown in Figure 41, with the number of points per sample class in Table 4. These sample sets are used to calculate all the different point classification attributes in our point cloud labelling algorithm, as described in earlier chapters.

Figure 41 The different sample classes

Table 4 Number of points per class sample

	Tiles	**Grass**	**Cycle path Red asphalt**	**Road Black asphalt**
Number of points	45438	66294	139148	310934

7.1. Definition of classes

In order to create classes for the two applications we aim for (discussed in chapter 4), it was decided to create classes for use (Figure 42Figure 42 Classes on use) and classes for material detection (Figure 43). The use represents mostly how a human would firstly interpret the scene. He/ She would identify the road, the cycle path and the footpath (tiles). For more specific applications, (e.g. the surface infiltration capacity) where the materials are needed because of their different behaviour to water, the specialist would be in need for the asphalt area, the grass and the tiles. Therefore, a semantically enriched point cloud with different types of labels for the specific application should be provided.

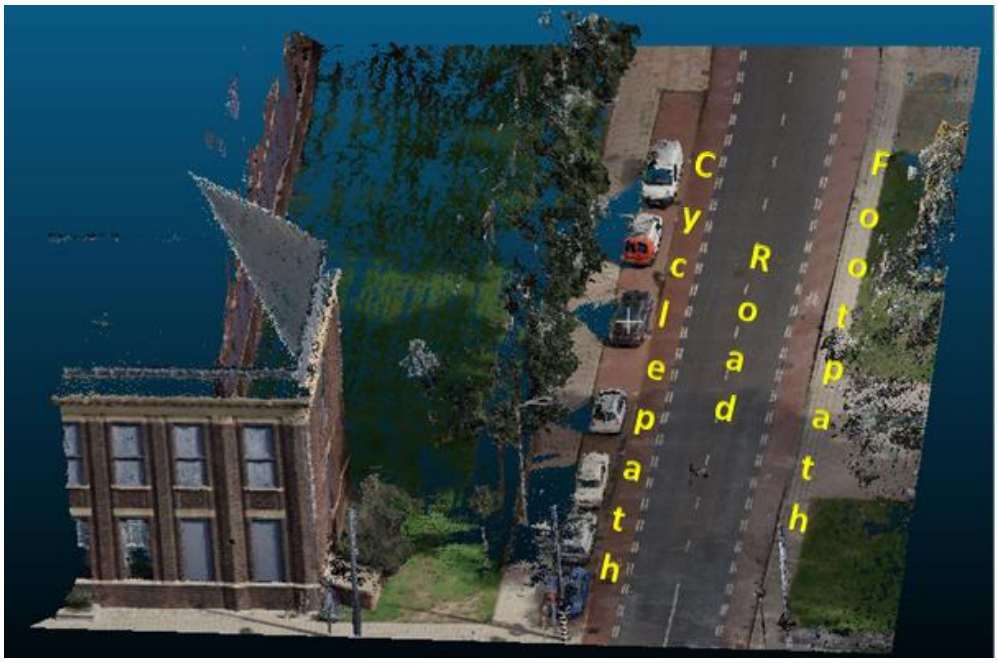


Figure 42 Classes on use

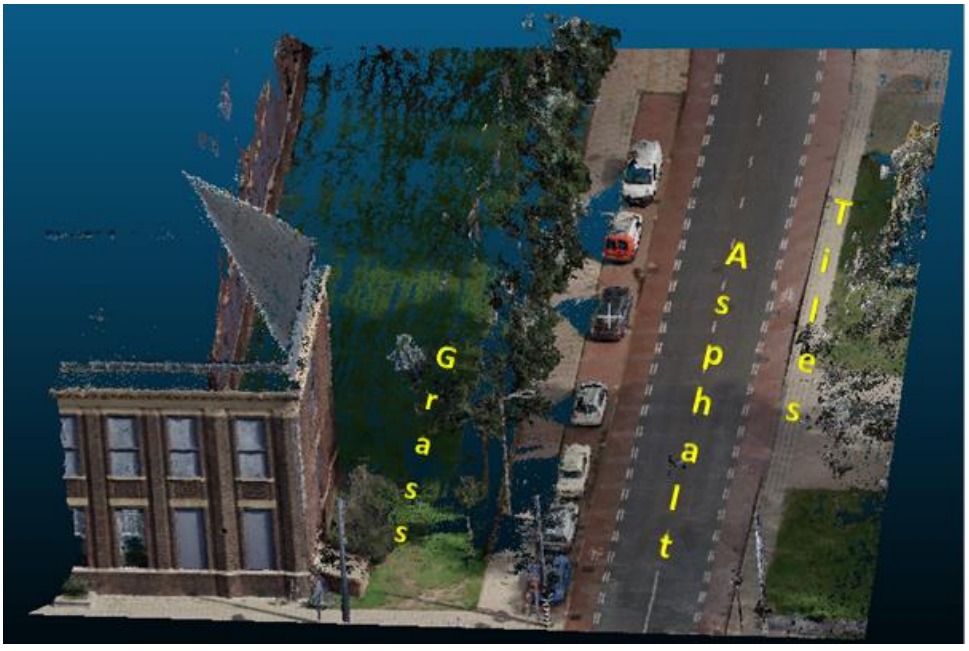


Figure 43 Classes on material

7.2. Colour classification attributes

In order to get statistics of four different classes in the point cloud, all RGB values are transformed (Table 5), using the formula in figure x, page x. Next a distribution is made for the different classes, which is shown in the Figure 44, Figure 45 and Figure 46.

Table 5 The different classes in for the HSV and CIELab colour spaces

		Hue	**Saturation**	**Value**	**Cie-L**	**Cie-a**	**Cie-b**
Cycle path	Min	0	0,0526	59	55,2111	-57,4633	2,7727
	Max	411,42	0,5897	156	82,4702	5,3931	4,8921
	Avg	15,4	0,33	91,7	66,3081	-28,6104	27,4657
	Std	32,2	0,07	12,6	7,4912	14,4717	9,4237
Road	Min	0	0,0101	1	0	-40,1297	-4,8109
	Max	405	1	106	69,1864	25,3001	24,0775
	Avg	119,1	0,05	67,6	58,0781	2,6113	10,5766
	Std	125	0,028	6	4,8163	6,3889	4,57125
Grass	Min	0	0,0192	3	10,3818	-48,7278	-6,1960
	Max	400	0,8571	184	84,5309	69,7676	35,9179
	Avg	87	0,12	124,7	46,4062	32,8621	21,7082
	Std	16,9	0,4	32,4	27,6225	33,5761	7,4386
Tiles	Min	0	0,0164	42	47,5838	-24,8339	10,1187
	Max	400	0,3968	186	88,4162	11,8330	28,5078
	Avg	29,7	0,53	57,9	74,6892	-1,5551	18,3044
	Std	9,6	0,857	29,3	16,0538	3,5283	5,1541

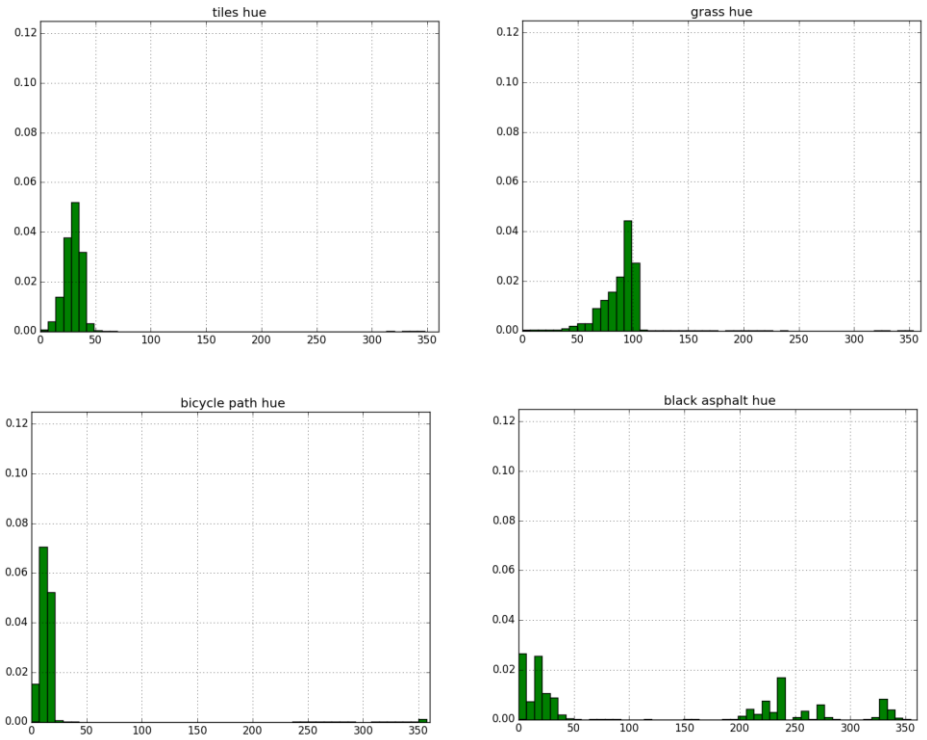


Figure 44 A distribution of the 'hue' attribute in the different classes.

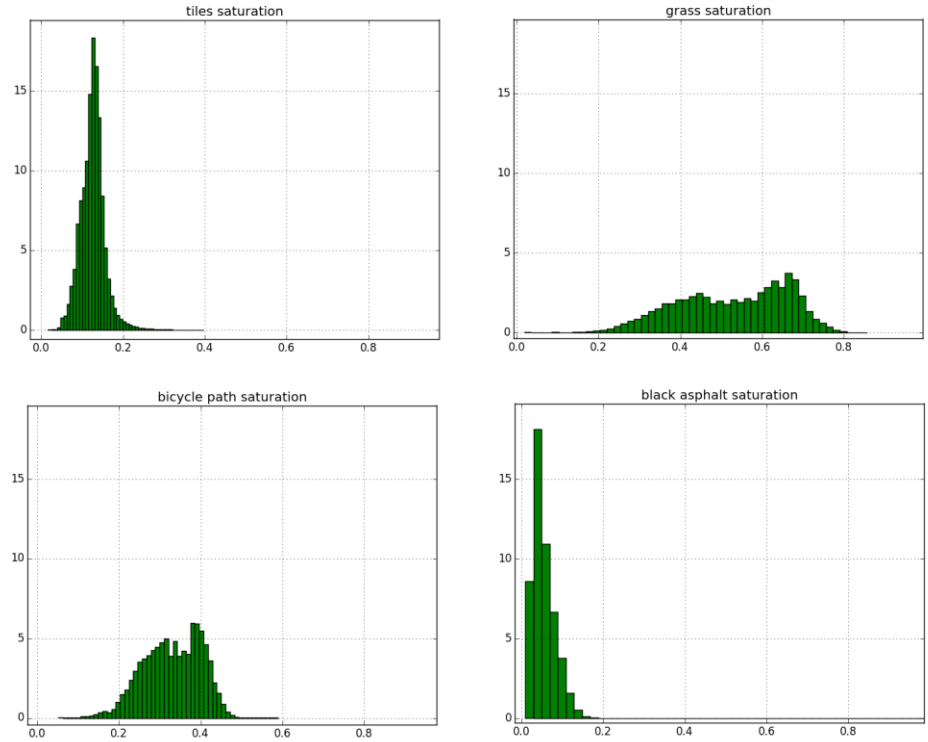


Figure 45 A distribution of the 'saturation' attribute in the different classes.

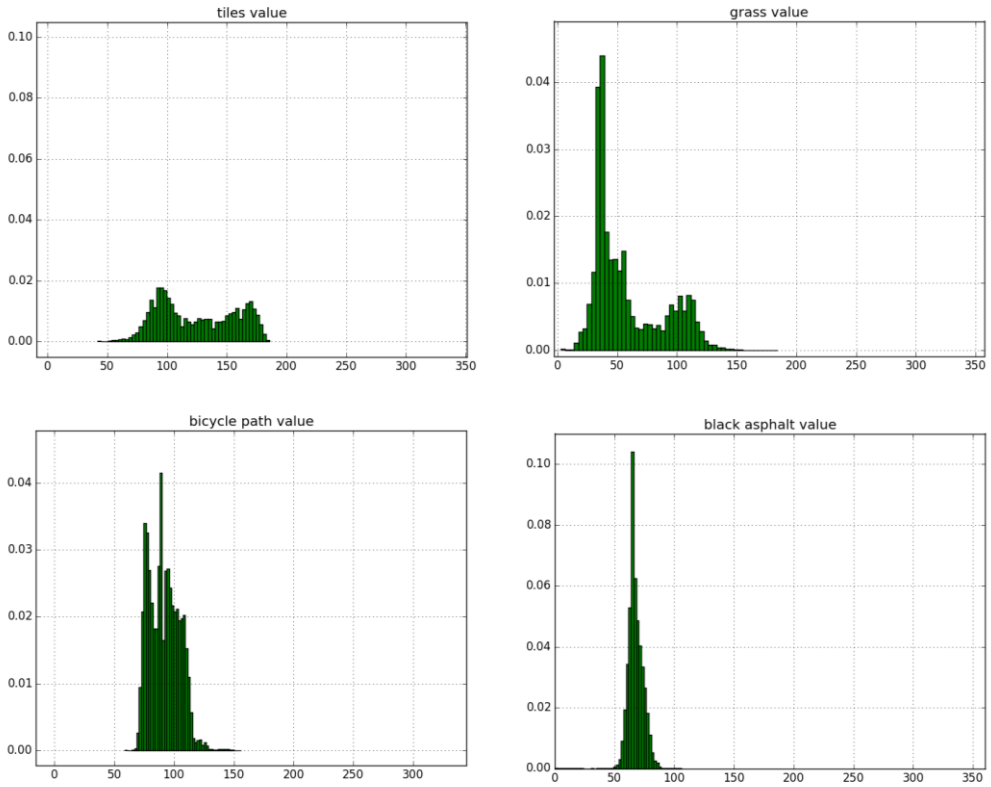


Figure 46 A distribution of the 'value' attribute in the different classes.

7.3. Geometrical classification attributes

The geometric attributes, as described in Section 6 of the Literature Overview, are calculated for, 5, 9 and 20 nearest neighbours. As it becomes clear from Table 6, there are no clear distinctions between the geometrical attributes and therefore they are not sufficient to be used as the only criteria for the classification. This is because these attributes describe aspects that are not present in our scene. For example, the linearity characteristic is usually used to detect power lines which is not relevant in our application. In addition to that, the different classes belong to roughly the same plane and thus cannot be distinguished by just using geometrical properties. This leads that the investigation of colour information for labelling point clouds is the best alternative.

Table 6 The geometrical properties of the different classes

CLASS	Black asphalt/road surface			Grass			Red asphalt/bicycle path			Tiles		
K- nearest neighbours	***5***	***9***	***20***	***5***	***9***	***20***	***5***	***9***	***20***	***5***	***9***	***20***
linearity average	0,563	0,404	0,292	0,608	0,485	0,446	0,565	0,404	0,292	0,555	0,402	0,303
linearity st. dev.	0,194	0,166	0,14	0,194	0,19	0,187	0,191	0,169	0,14	0,189	0,169	0,16
planarity average	0,435	0,593	0,704	0,388	0,508	0,547	0,432	0,591	0,704	0,445	0,598	0,696
planarity st. dev.	0,194	0,166	0,14	0,194	0,19	0,188	0,19	0,169	0,14	0,189	0,169	0,16
scattering average	0,0018	0,002	0,003	0,004	0,006	0,006	0,003	0,004	0,003	0,0006	0,0009	0,001
scattering st. dev.	0,0056	0,007	0,006	0,012	0,014	0,012	0,007	0,009	0,006	0,0018	0,002	0,0015
omnivariance average		0,061	0,656	0,062	0,078	0,081	0,054	0,069	0,656	0,036	0,0446	0,048
omnivariance st. dev.		0,026	0,025	0,036	0,359	0,032	0,029	0,029	0,025	0,017	0,016	0,015
anisotropy average	0,999	0,997	0,997	0,996	0,994	0,993	0,997	0,996	0,997	0,999	0,999	0,999
anisotropy st. dev.	0,006	0,007	0,006	0,012	0,014	0,012	0,007	0,009	0,006	0,002	0,002	0,001
change of curvature average	0,0012	0,001	0,002	0,0031	0,004	0,003	0,0017	0,002	0,002	0,0004	0,0005	0,0006
change of curvature st. dev.	0,0036	0,004	0,004	0,0078	0,008	0,007	0,0047	0,005	0,004	0,001	0,0014	0,0009
sum of eigenvectors average	0,0003	0,0005	0,001	0,0037	0,006	0,014	0,0003	0,0005	0,001	0,0009	0,0016	0,0037
sum of eigenvectors st. dev.	0,0001	0,0002	0,0003	0,0404	0,04	0,1	0,0001	0,0002	0,0003	0,00041	0,0005	0,001

8. Point cloud labelling: Point based methods

Labelling a point cloud in this research means getting those parts out of the point cloud necessary for assessing a certain task. In this case it concerns the points of the point cloud belonging to the earth's surface, or more explicit to the points belonging to the "earth's" surface within cities. The perception of this research is that the "earth's" surface within cities is categorized in black asphalt roads, tile roads, red cycle asphalt roads and grass. As explained in chapter XX classes are made from these categories of surface.

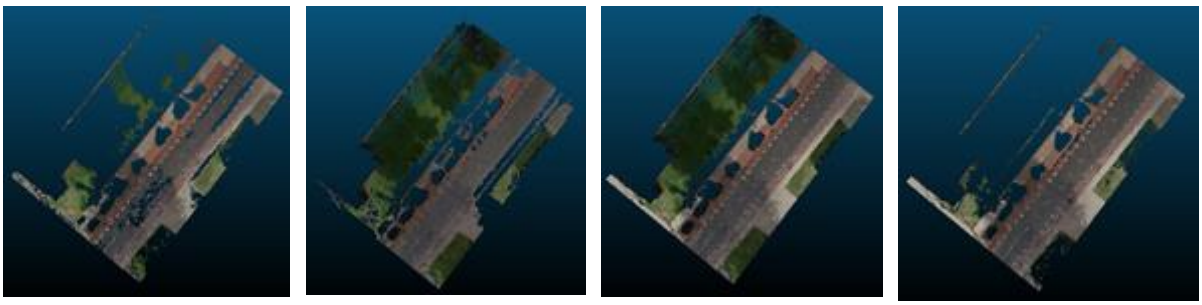
Since it was unclear whether it was possible to label point cloud segments mainly based on colour a point based method was set up. The point based methods will check for every point within the point cloud if it belongs to a certain class due to its HSV or CIE-Lab colour values. This can be done with minimum, maximum, mean and standard deviation but also with methods explained in the literature overview like the Support Vector Machine classifier and the decision tree.

8.1. H, S and V values, minimum and maximum

The training data are used to calculate the minimum and the maximum H, S and V value for every class. Based on these values it is possible to assign the points each to a specific class.

For every point in the point cloud it is checked through its H, S and V values whether it belongs to a certain class. When all points belonging to this class are extracted and written to a text file the explained process is repeated. This means checking all points again but now for another class.

```
for each H-value in the list that contains all H-values of the points in the point cloud:
        check for one point if the H-value is between the min and max H value of a certain
        class. iff it is:
                check for the same point if the S-value is between the min and max S value
                of a certain class. if it is:
                        check for the same point if the V-value is between the min and the
                        max V-value of a certain class. if it is:
                        append this point to a list
write this list of points belonging to a certain class to a txt-file
```



Cycle path Road Grass Tiles

Figure 47 Point based labelling based on minimum and maximum HSV values

This method is time consuming because all points within the point cloud are checked multiple times and the ranges that were checked were wide. The results were therefore not the expected quality. As is visualized in Figure 47 a lot of points are classified into several classes. The class grass for example contains points which actually belong to the classes cycle path, tiles and or road.

A possible reason for this could be that the used training data consists colours that are that not only belong to one class but also occur in other classes, it might even be the case that this colour occurs more often in other classes. The Grass training data for example also contains parts that used to be grass but when acquiring the point cloud were "bare ground" (Figure gras). The colour of this bare ground is "grey" instead of the expected green and therefore is easily classified wrong or at least not as grass. To minimize the points that are wrongly classified due to different colours in the trainings data instead of using the minimum and maximum value, in the next method the mean and the standard deviation will be used to reduce the colour value ranges.

Figure 48 Part of the training samples for grass

8.2. H, S and V values, mean and 2 * standard deviation

Since results of using the maximum and minimum HSV values were not accurate a next option is tried, instead of using the minimum and maximum HSV values, the mean plus and minus 2* the standard deviation is used as range in which the HSV values need to be in order to be labelled as a certain class. It is assumed that the values are Gaussian distributed (shown in figure xxx chapter of the training samples). This mean plus and minus 2 * standard deviation is a 95,5% range of all H, S and V values, the biggest outliers within the trainings data are therefore not taken into account anymore with the classification of the points.

For every class the mean H, S and V values and the standard deviation of each value are calculated from the training data. As the following pseudo code explains, for each point in the point cloud it is checked whether its H, S and V values lie within the range between the mean H, S and V values and 2 times the standard deviation.

```
for each sample the H, S and V values are calculated and put in a list and the average and
the standard deviation of these H, S and V values are calculated.
        check for each H value of each point if the H-value is between the average minus 2
        * the standard deviation and the average plus 2 * the standard deviation of a
        certain class. if so:
                check for the same point if its S-value is between the average minus 2 *
                the standard deviation and the average plus 2 * the standard deviation of a
                certain class. if it is:
                        check for the same point if the V-value is between the average minus
                        2 * the standard deviation and the average plus 2 * the standard
                        deviation of a certain class. if it is:
                        append this point to a list
        format this list
write this list of points belonging to a certain class to a txt-file belonging to this
class
```

The results per class are shown in Figure 49. All classes contain less points and seem more precise than the results when using the minimum and maximum values. The classes now don't contain much points that actually belong to other classes, but still the labels are not that precise and correct yet. Especially the classes Cycle path and Tiles contain points that should have been classified differently. The error of omission of the Tiles class shown in Table 7is therefore 68% which led to a mapping accuracy of just 32%. This table also shows a mapping accuracy of 77% for Road 73% for Cycle path and 69% for Grass. This means that only 77%, 73% and 69% of the points within Road, Cycle path and Grass are correctly classified.

*Table 7 The confusion matrix for the H, S and V values, mean and 2 * standard deviation*

		Predicted classes				
Actual Classes		Road	Cycle	Tiles	Grass	Total
	Road	21522	969	432	0	22923
	Cycle	0	14877	0	0	14877
	Tiles	9047	25	13160	38	22270
	Grass	282	0	397	3424	4103
	Total	30851	15871	13989	3462	

	Omissions	Commissions	Mapping Accuracy
Road	5%	23%	77%
Cycle	15%	15%	73%
Tiles	68%	1%	32%
Grass	100%	46%	69%

Besides the method using the minimum and maximum values from the training samples, this method is already promising in the field of labelling point clouds just on colour values even though the results could be even more precise.

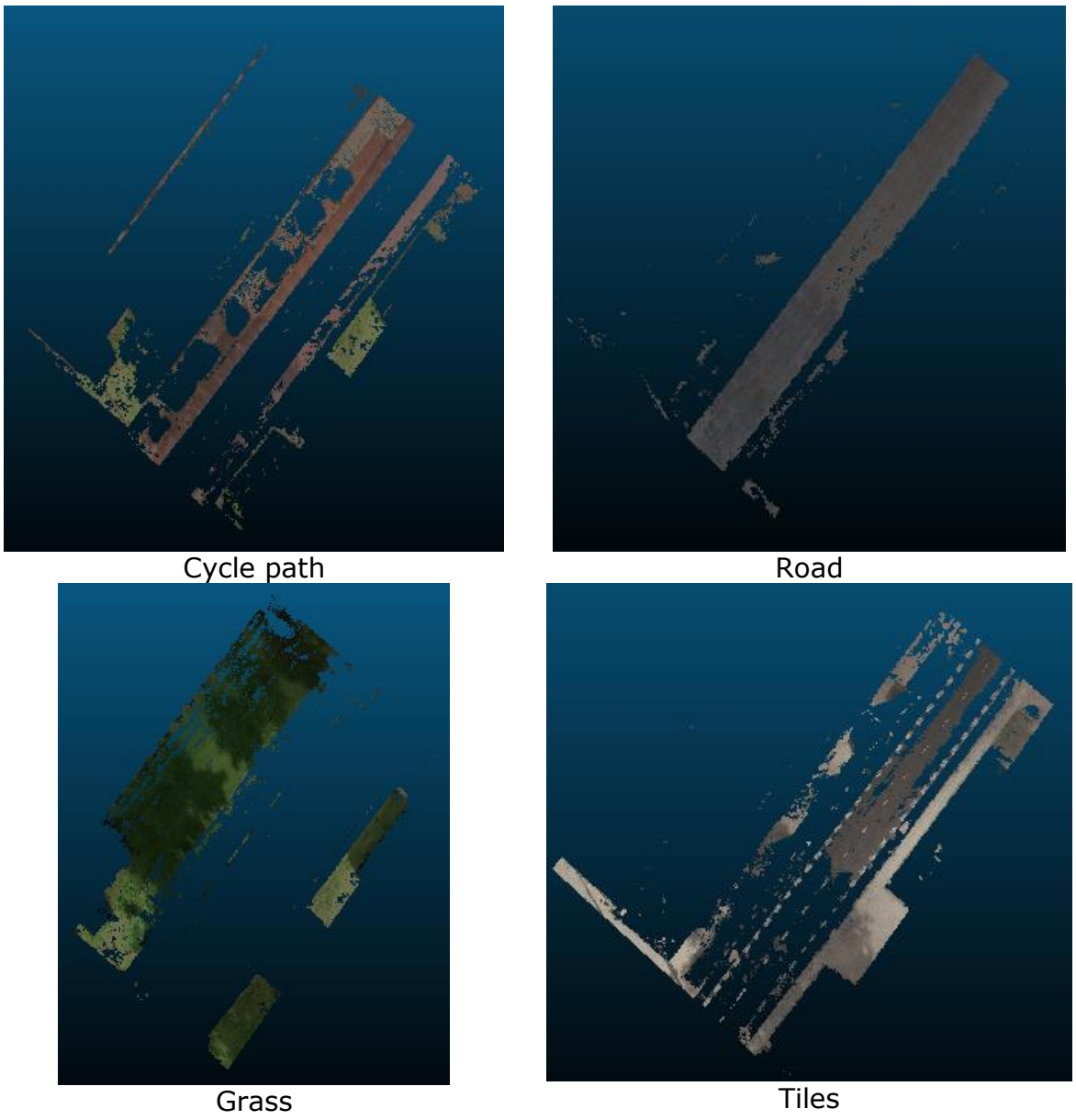


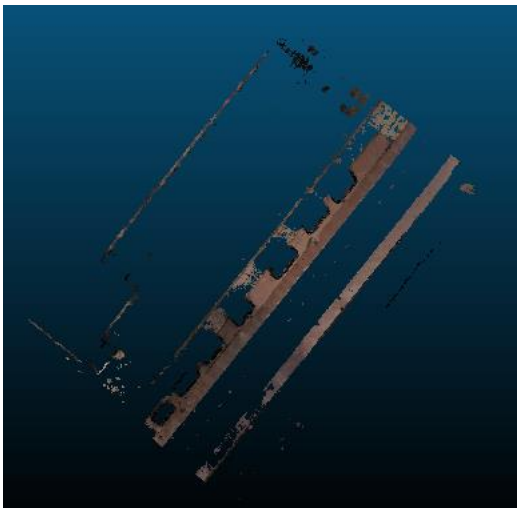
*Figure 49 Point based labelling based on mean and 2 * standard deviation*

8.3. Cie-LAB values, mean and 2 * standard deviation

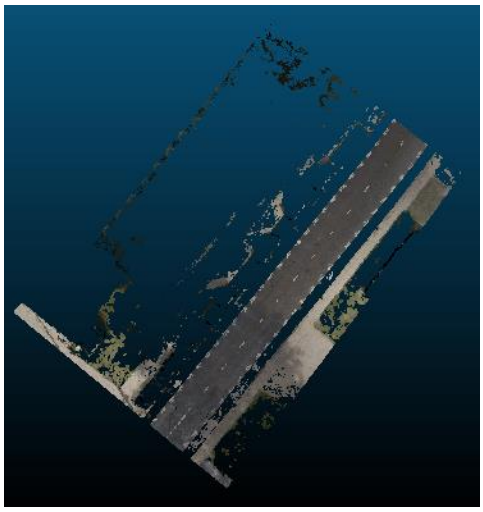
In order to check whether Cie-LAB colour values are more accurate within classifying point clouds based on colours also a classification, labelling a point cloud, is done based on Cie-LAB values.

```
for each sample the Cie L, A and B values are calculated and put in a list and the average
and the standard deviation of these L, A and BV values are calculated.
```

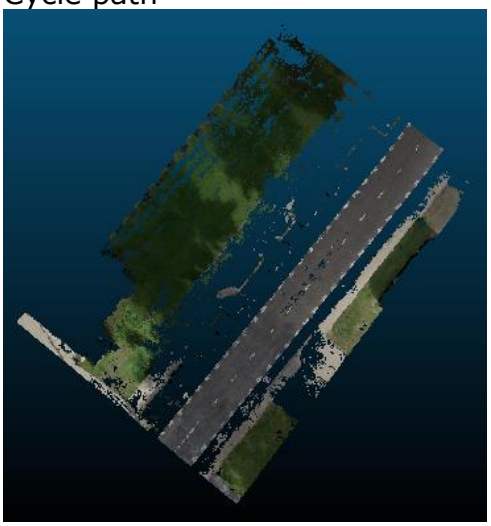
```
        check for each L value of each point if the L-value is between the average minus 2
        * the standard deviation and the average plus 2 * the standard deviation of a
        certain class. if so:
                check for the same point if its A-value is between the average minus 2 *
                the standard deviation and the average plus 2 * the standard deviation of a
                certain class. if it is:
                        check for the same point if the B-value is between the average minus
                        2 * the standard deviation and the average plus 2 * the standard
                        deviation of a certain class. if it is:
                        append this point to a list
        format this list
write this list of points belonging to a certain class to a txt-file belonging to this
class
```



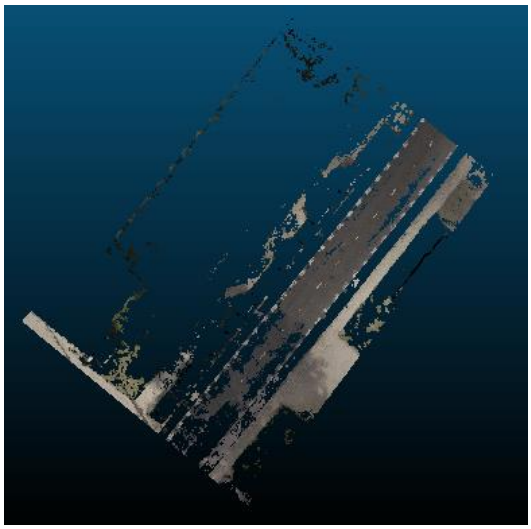
Cycle path



Road



Grass



Tiles

*Figure 50 point based labelling based on mean and 2 * standard deviation*

When comparing to the method using the H, S, v values with the mean and 2 * standard deviation range, the results of using the CIELab values shown in Figure 50 are less adequate. Instead of a Tiles class with a very small mapping accuracy of 32%, Using the CIElab colour values result in the classes Road, Grass and Tiles with an approximate mapping accuracy of just 37%, 23% and 20% even though

the same range of values is used (Table 8). This result therefore shows that using different colour spaces give different results.

*Table 8 The confusion matrix for the CIELab values, mean and 2 * standard deviation*

		Predicted classes				
Actual classes		**Road**	**Cycle**	**Tiles**	**Grass**	**Total**
	Road	77786	1186	20223	2079	101274
	Cycle	0	16556	0	0	16556
	Tiles	69412	2258	24136	578	96384
	Grass	37040	216	3382	12761	53399
	Total	184238	20216	47741	15418	

	Omissions	**Commissions**	**Mapping Accuracy**
Road	23%	105%	37%
Cycle	0%	22%	82%
Tiles	75%	24%	20%
Grass	31%	5%	23%

8.4. CIELab a value, mean and 2 * standard deviation

When visualizing a histogram plot for the CIELab values (Figure 51) it is concluded that the A value distinguishes the classes grass and cycle path the most. Therefore, a classification is made based on only the A value, its mean and 2 * standard deviation. With this method it is checked whether only using the a-value is enough in distinguishing different parts of a point cloud.

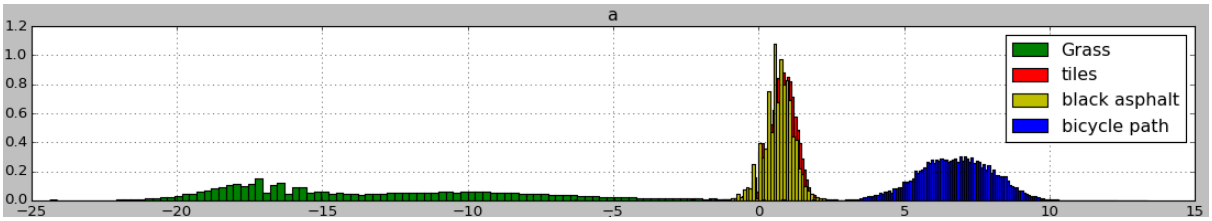
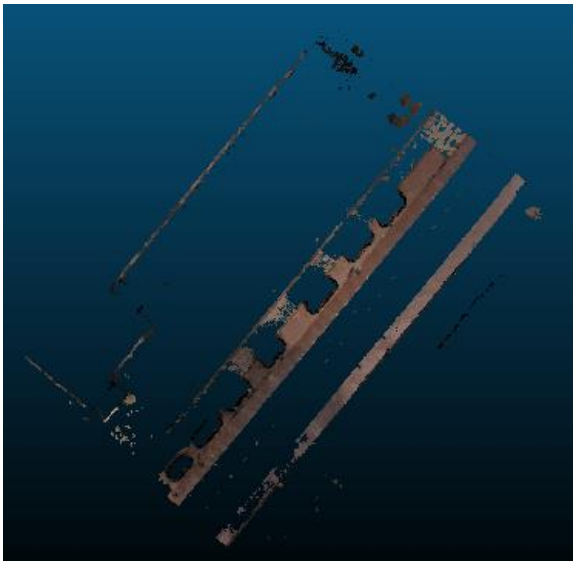


Figure 51 The distribution of the CIE - a value for the different classes

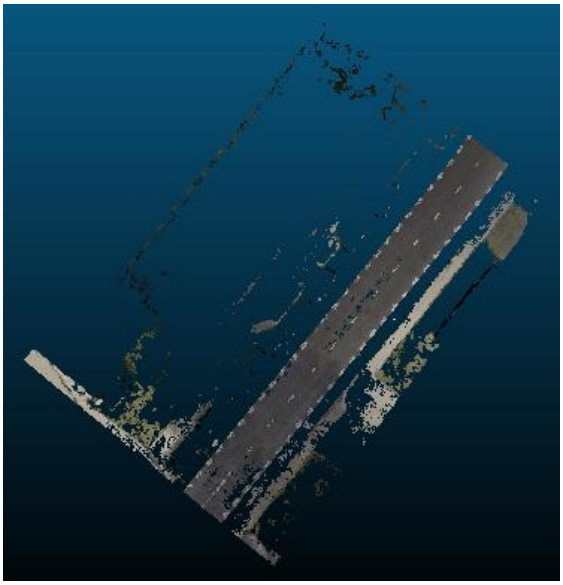
```
for each sample the Cie L, A and B values are calculated and put in a list and the average
and the standard deviation of these L, A and A values are calculated.
        check for each A value of each point if the A-value is between the average minus 2
        * the standard deviation and the average plus 2 * the standard deviation of a
        certain class. if so:
                append this point to a list
        format this list
write this list of points belonging to a certain class to a txt-file belonging to this
class
```

The results, shown in Figure 52, should be most accurate for the Cycle path and Grass classes. Table 9 confirms on the one hand the high, as in 100%, mapping accuracy for the Cycle path class but on the other hand the low, as in 26%, mapping accuracy for Grass. So even though the cycle path is distinguished from

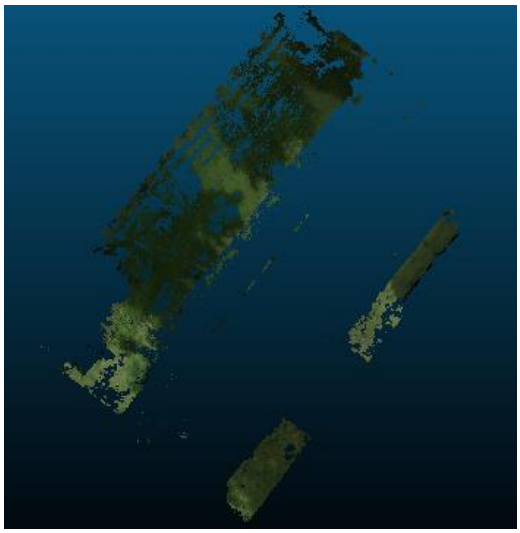
the point cloud with a very high accuracy, The Grass and other classes are not, using only the a-value therefore might not be most accurate.



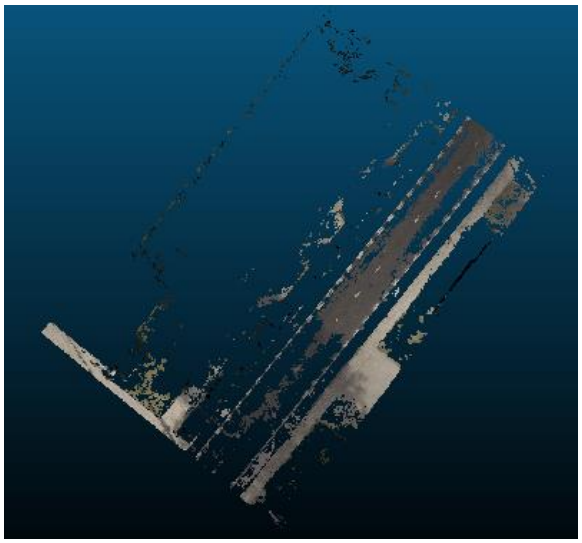
Cycle path



Road



Grass



Tiles

Figure 52 Point based labelling based on mean and 2 standard deviation on the A value*

*Table 9 The confusion matrix for the CIE a value, mean and 2 * standard deviation*

		Predicted classes				
		Road	**Cycle**	**Tiles**	**Grass**	**Total**
Actual Classes	**Road**	35207	0	10820	548	46575
	Cycle	0	8060	0	0	8060
	Tiles	26364	0	13814	77	40255
	Grass	24651	0	1664	9713	36028
	Total	86222	8060	26298	10338	

	Omissions	Commissions	Mapping Accuracy
Road	24%	110%	36%
Cycle	0%	0%	100%
Tiles	66%	31%	26%
Grass	32%	2%	26%

8.5. SVM classifier

Apart from the self-made, range based, point based point cloud classification methods, the SVM classifier is easily implemented on point cloud data with known training data using the Python sklearn module. This module is a data mining and data analysis tool, built on NumPy, SciPy, and matplotlib.

The SVM classifier uses several parameters in order to predict the class for each point (Table 10). Most of these parameters are already set within the sklearn method, just a few are adapted to get the best results. The adapted parameters are the kernel and the class weight for the Road. The class weight of the asphalt is set to two and the kernel is changed from 'rbf' to 'poly' so that the decision space can be formed by every imaginable shape.

Table 10 SVC Parameters

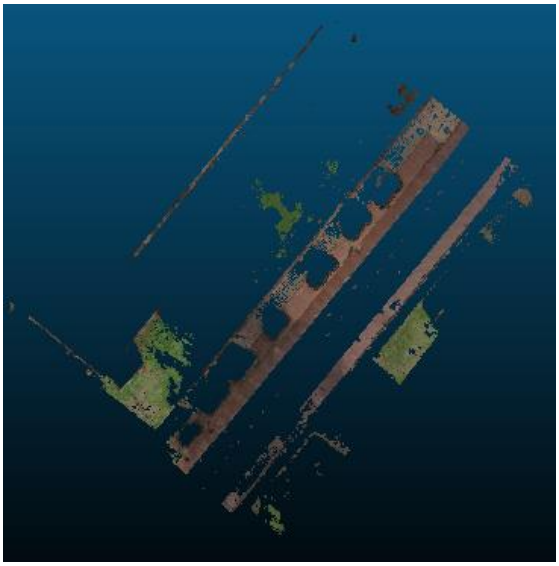
C	Cache_size	Class_weight	Coefficient	Degree
Gamma	Kernel	Max_iter	Probability	Random_state
Shrinking	Tol	Verbose		

The input of the SVC are two arrays, one with the H, S and V values and one with the classes. The algorithm creates a polygonal decision space, whereby the properties of the regions are used as classification criteria. The classification criteria are derived from samples taken from the scene. In the classification algorithm, the input is an array that holds 12 values:

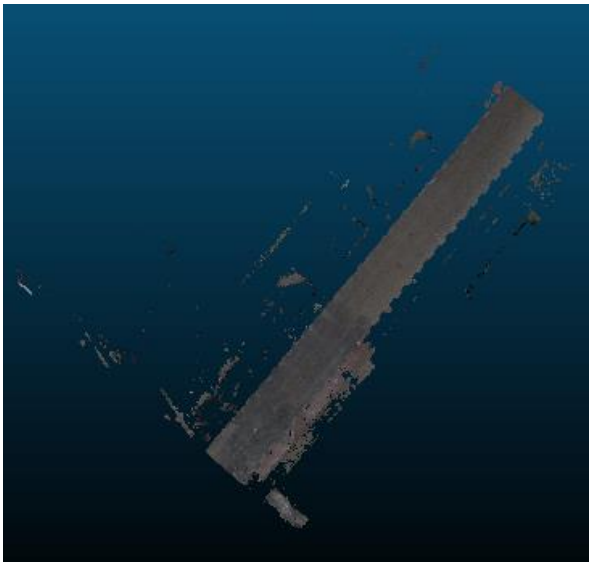
- The minimum of the L, a and b values.
- The maximum of the L, a and b values.
- The average of the L, a and b values.
- The standard deviation of the L, a and b values

The same 12 values of the different regions are used to calculate the probability that the region belongs to a certain class.

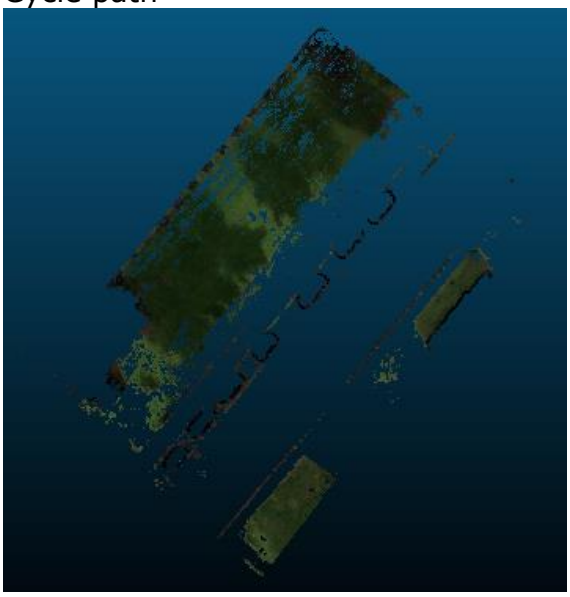
The result of the SVC classifier is shown in Figure 53. It visualizes the fact that the grass and the road is classified above expectations but also that the cycle path also includes points that belong to grass and tiles also include points that belong to the asphalt road. But even though classes exist that contain points that should have been classified differently the accuracy of this classification method is quite high. Especially the Road class which almost contain none wrongly classified points, it has an approximately mapping accuracy of 92% (table SVM)



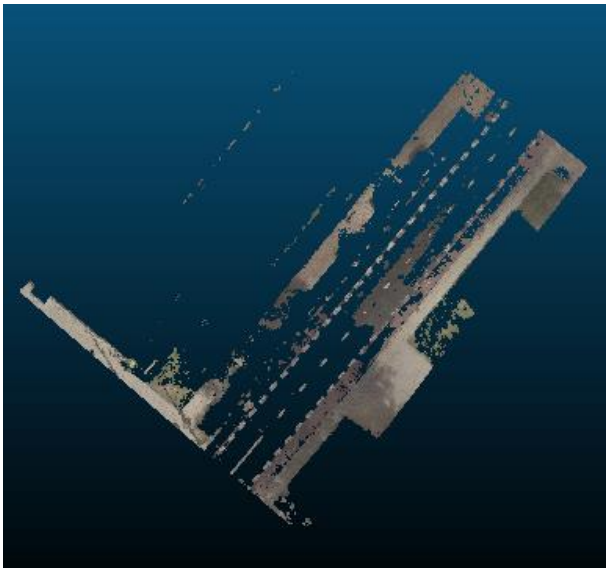
Cycle path



Road



Grass



Tiles

Figure 53 Point based labelling from the SVC classifier

Table 11 The confusion matrix for the SVC classifier

		Predicted classes				
Actual Classes		Road	Cycle	Tiles	Grass	Total
	Road	167447	9371	874	0	177692
	Cycle	189	16980	15	4179	21363
	Tiles	4262	6753	27567	0	38582
	Grass	0	0	0	6218	6218
	Total	25135	20752	12309	10397	

	Omissions	Commissions	Mapping Accuracy
Road	6%	3%	92%
Cycle	21%	75%	45%
Tiles	29%	2%	70%
Grass	100%	67%	60%

8.6. Decision Tree

Apart from the SVM classification the python sklearn module also contains a method for applying a decision tree classification. The sklearn module creates a model that predicts the value of a target variable by learning decision rules from the training samples. The decision tree function takes the same input as the SVC classifier, but does not create a multidimensional decision space. Instead, it makes a tree structure, whereby the algorithm traverses this tree and the leaf node, eventually classifying the points.

The results of the decision tree classification seem in comparison with the SVM classification according Figure 54 are not satisfying. The Tiles and Road classes distinguished with the decision tree classification show a lot of points that are wrongly classified. But even though figure DC visualizes the wrongly classified points, the mapping accuracy of the different classes are higher, they are approximately 85% for the Road, 74% for the Cycle path, 75% for Grass and the Tiles class has as expected the lowest accuracy, 57% (Table 12).

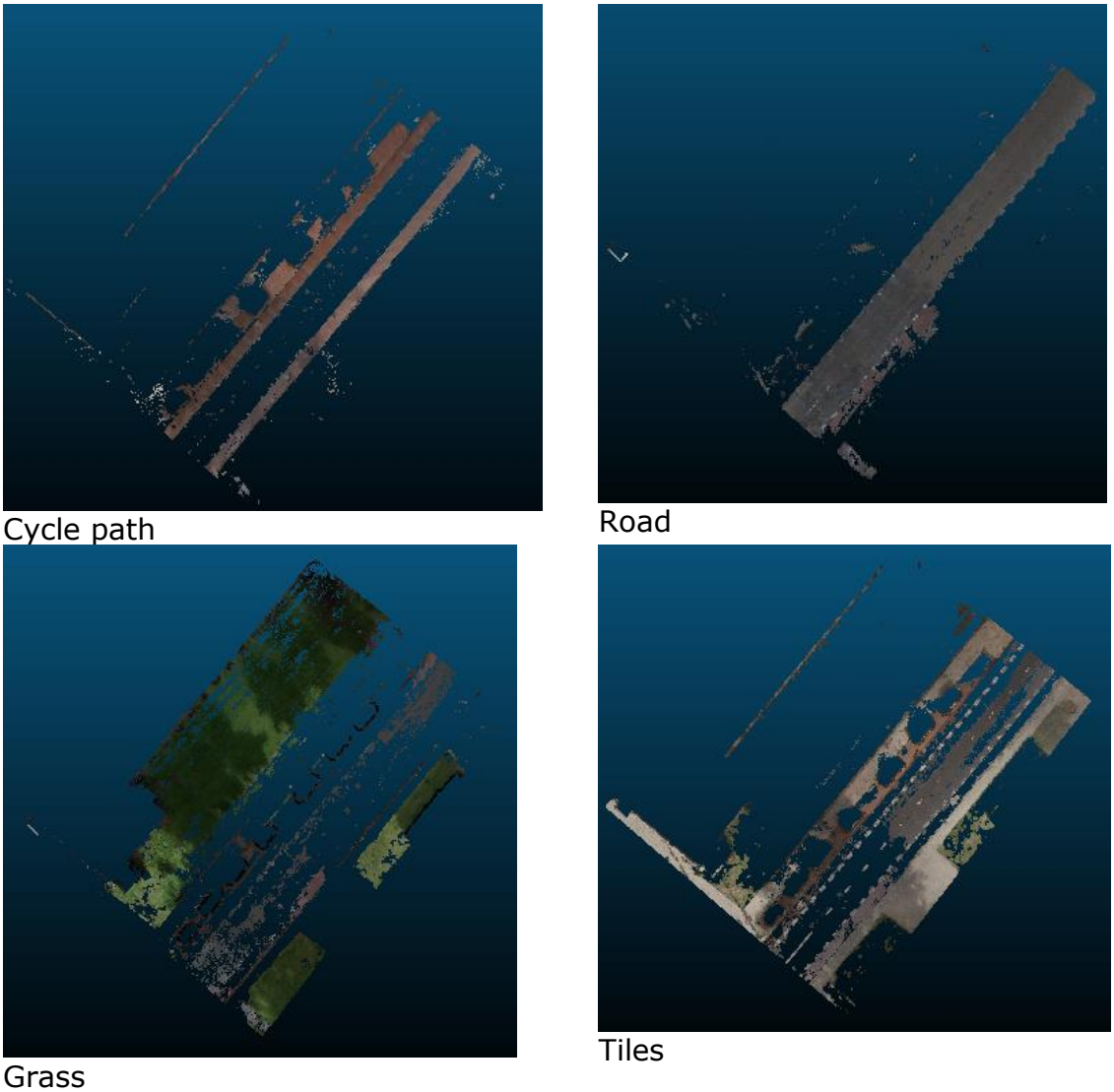


Figure 54 Point based labelling from the decision tree

Table 12 Confusion matrix for the decision tree

	Road	Cycle	Tiles	Grass	Total
Road	124689	5383	234	0	130306
Cycle	288	32368	0	0	32656
Tiles	14989	3343	26480	1230	46042
Grass	617	2366	0	12593	15576
Total	140583	43460	26714	13823	

	Omissions	Commissions	Mapping Accuracy
Road	4%	12%	85%
Cycle	1%	34%	74%
Tiles	42%	1%	57%
Grass	96%	8%	75%

8.7. Concluding point based methods

It can be said that the overall (mapping) accuracy of the point based classification methods is not outstanding due to training samples with multiple colours and a point cloud with a lot of colour inconsistencies. The methods compared conclude the decision tree classification as the best point based classification method. This method though uses harsh distinguishments while the boundaries between the classes are not that harsh. A problem with the point based methods is the fact that it is possible that one point is multiple times classified this means that one point could be classified as Grass but also as Tiles. Therefore within the omission and commision errors those points might be classified rightly while actually they are classified wrong, because it should not be possible to classify a point multiple times.

9. Point cloud labelling: Region based methods

Region based methods allow to first grow regions before classification, the grown regions are then the ones to be classified instead of the raw points used in the point based methods for point cloud classification.

Region growing can be referred as an unsupervised classification method. It is called unsupervised because the user does not need to define anything about the classes, at least at the initial step. The region growing algorithm that is used in this project is a combination of the colour based segmentation algorithm of Zhana et al., (2009) and the algorithm of Rabbani, et al., (2006). The variation is found in the fact that the point cloud is first converted to a different colour space than the RGB and then according to those values colour similarity together with angle are the defining parameters to whether the point belongs to the region or not. The curvature value is also used to define more seeds for each region. Therefore, the algorithm is defined as follows:

```
Sort the point cloud according to the curvature value and initialise an Available
points list and a region list.
For each point in the Available points list:
```

```
    If the point does not belong yet to a region:
        initialise a region and a seeds set
        Take the point P with the minimum curvature
        Append the point P to the seeds set and the region set
        Until the current seeds set is empty:
            remove the current seed form the seed set.
            Find the neighbours of the seed using a kD-tree data structure
            For each of the neighbours
                if the angle between the normal of the seed and that point
                is less than a threshold value and the single axis colour
                distance is within a threshold.
                    add the point to the current region
                    if the difference between the curvature of the seed
                    and its neighbour's is less than a threshold,
                        add the point to the list of seeds
    Add the region to the region list
If the seeds set has no more seeds, the region has grown and the algorithm is
repeated from the beginning.
```

The designed algorithm is tested for its output for the test dataset that is part of the whole point cloud provided by cyclomedia. The thresholds used for the angle, curvature and colour difference were discovered after some testing with the point cloud. The thresholds that gave the best distinction between the classes are:

angle	45
curvature	0.001
colour	0.2

The explanation for such a low curvature threshold lies in the structure of the scene. The majority of the classes are found inside the same plane and therefore, raising the curvature threshold leads to under-segmentation. By under-segmentation it is meant that two different classes are labelled as one. This was the case for curvature thresholds over 0.001. Therefore, decision was made that over-segmentation is more acceptable than under-segmentation. Over-segmentation occurs when one surface is divided into really small surfaces. Intentionally causing over-segmentation is, at least in theory, much simpler to correct than under-segmentation, since human intervention cannot be avoided in the second case. In addition to that, lower thresholds will allow the same algorithm to work for the other scenes of the whole point cloud. Also, our test scene holds properties that created region growing errors, if only curvature and normals were used. As shown in Figure 55, the road is not always separated from the footpath (tiles) by curbs, resulting that the region growing algorithm created one region out of the road and the footpath (tiles).

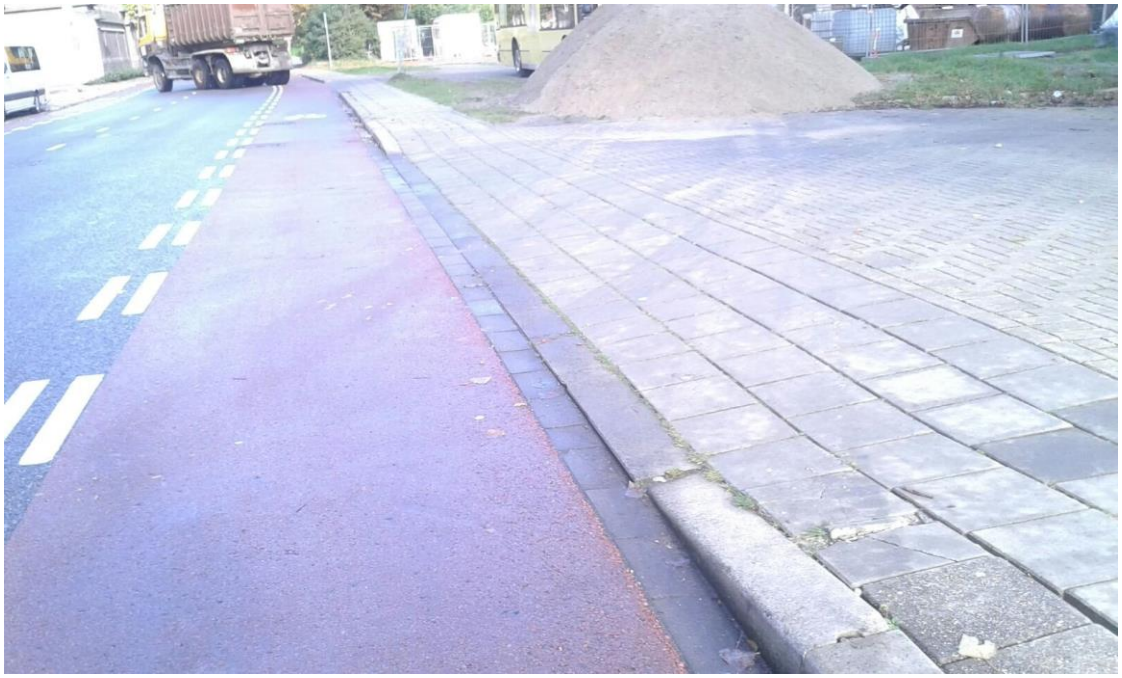


Figure 55 How error in the region growing on curvature and normals occurred

The over-segmentation in the point cloud, however, is not only caused by the low curvature threshold. It is mostly because of the colour changing very rapidly inside the scene. In reality those colour changes are non-existent. For example, in Figure 56, the variation of the colour of the cycle path is shown. In this case, because the car shooted the pictures in a rainy day, the presence of water on the asphalt has a "graying" effect on the colour if viewed from a distance. Therefore, at the side where the car was driving, because the cycle path is being shooted from the short distance has no effect on the colour. However, the opposite side is more distant and thus affected by the presence of water.

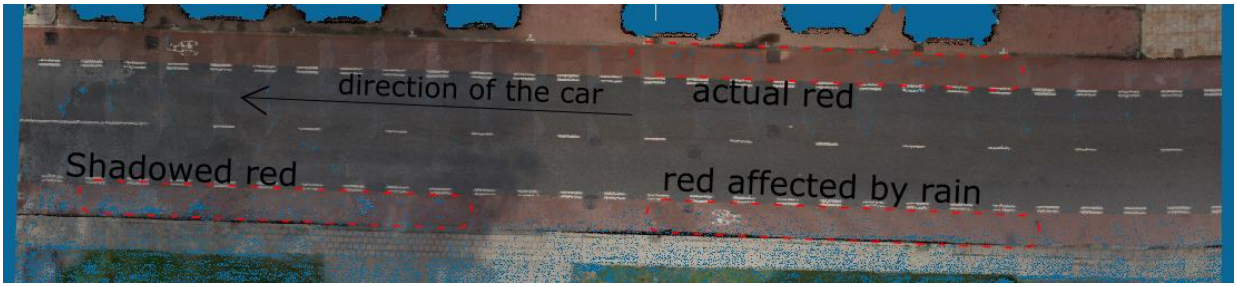


Figure 56 The variation of the colours in one scene

In addition to the variations because of the rain, the shadow of the BK city building also causes differences in the colour. This is also visible in figure. The rest of the colours are also affected by same or similar reasons. All those reasons lead to the result of the region growing algorithm that are shown in the next Figure 57. It has to be noted that not all regions are visualised because they contain less than 100 points. This means that there are even more smaller regions that compose one region. The merging of those smaller regions into semantically meaningful ones is the topic of the next sections.

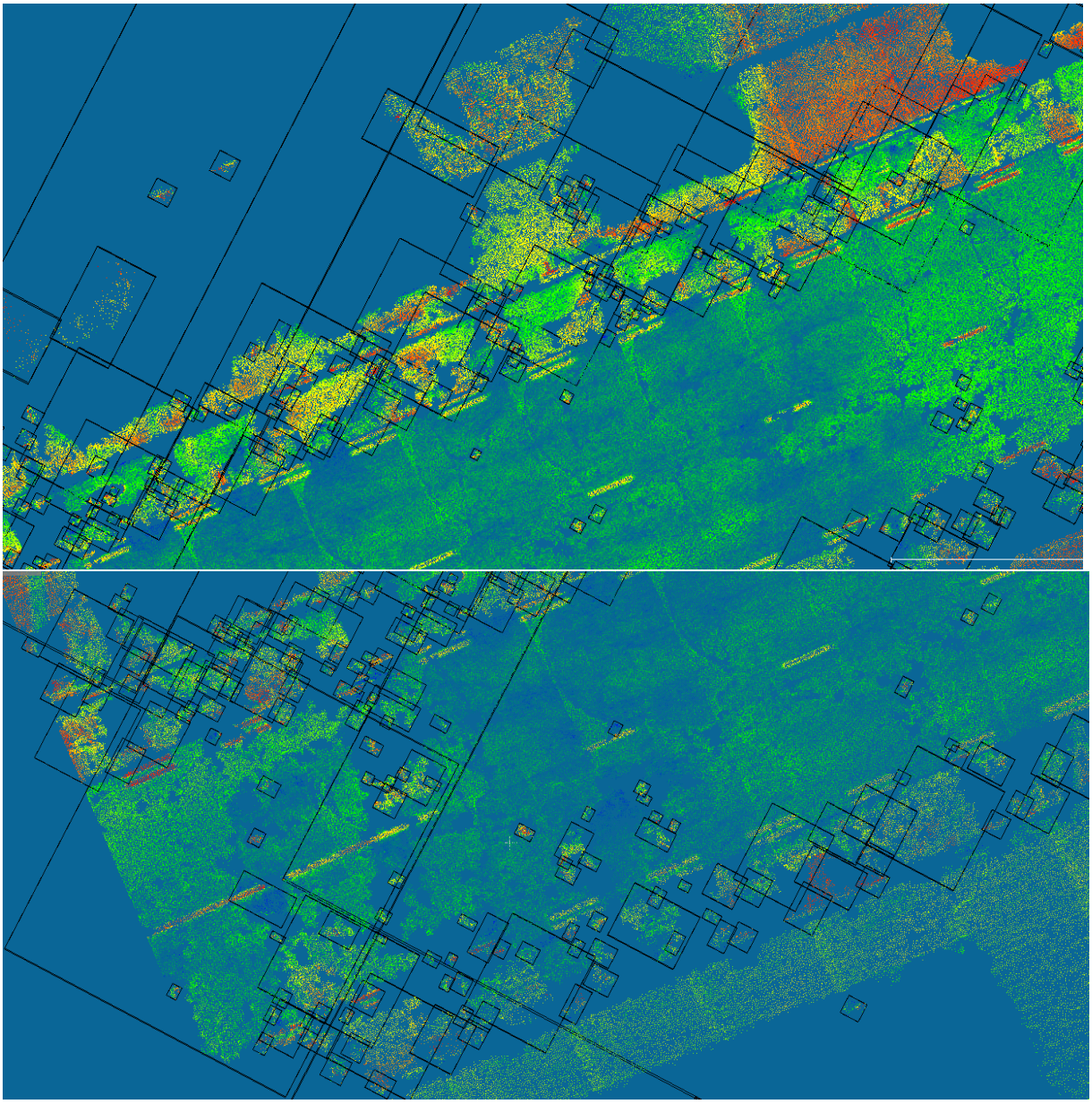


Figure 57 The presence of over-segmentation in the results of the region growing algorithm

9.1. Unsupervised merging based on colour similarity

In this section, the over-segmentation that was created in the previous section by the region growing algorithm is tackled by creating a merging algorithm that is based on colometrical similarity. The main idea is that the small regions that are found in the region growing are merged with their neighbouring regions that have the most similar colour. This region merging process could be seen as a variation of the region growing process.

For each small region, the neighbours of it are found by traversing the neighbours of the points that are present in the region. This, of course, means that there will be points that their neighbours are in the region currently being checked. However, the algorithm will take into consideration only the regions that are different from itself.

Having defined the neighbouring regions the colometrical distance is calculated with reference to the current region. The region that has the minimum distance is then the best candidate and the two regions are merged. If the newly created regions are again considered small, the algorithm recursively tries to merge more points to them. If the region is big enough, then the region has grown to a satisfactory level. The algorithm repeats until there is not a small region present in the scene.

The colometrical distance is calculated as the euclidean distance of the components of the CIELab space. The L component, however, since it represents lightness is not taken into account due to the changing shadow conditions in the scene.

The pseudocode of the merging algorithm is as follows:

```
Input: a list {A} with all the regions
Initialise a list {R} that will contain the regions to be merged
Initialise a Set {N} that will contain all the region numbers smaller than a threshold
Initialise a list {D} to keep track of the region numbers that are merged with other regions

For each region in {A} that has less points than a specified threshold
      Append the region to {R}
      Add the region number to {N}
While there are regions to be merged
       Initialise a Set {KN} that will hold the region numbers of the neighbours of the
       points in the current region that are not the current region itself
      Take one region from {R}, append its region number to {D} and remove it from {N}
       For each point in the current region
             For each neighbour of the point
                      Find the number of the region the neighbour belongs to
                     If the region number is different than that of the current region
                            Add the region number to {KN}
       Find the region numbers that are in {KN} but not in {D}
       If this difference contains regions
              Find the colometrical distance of these regions with the current region
              The points of the current region are appended to the points of the region that
              has the most similar colour
              If this region is in {R} and contains more points than the threshold
                     Remove it from {R} and {N} and append it to {D}
      Empty the points of the current region in {A}
```

The results of the algorithm are presented in Figure 58. The threshold of what is considered a small region is taken to 5000 points. There is can be visible that the over-segmentation is reduced. The output, however, is not without faults. A closer inspection can show that parts of the grass are not returned from this method. This has to do probably with the fact that grass regions are composed from 2 or 10 points each and the colour variation is very big. This would mean that the regions would never contain points more than the defined threshold. In addition to that, parts of the grass on the other side are merged with parts of the tiles. This leads now to under-segmentation. Finally, the cycle path although it is split in less regions, it still presents over-segmentation. Therefore, trying to find the best threshold that would maximise the cycle path merging and minimise the under-segmentation between tiles and grass is not possible. The method, nevertheless, provides a good distinguish of the road, if some artifacts are ignored (Figure 59).

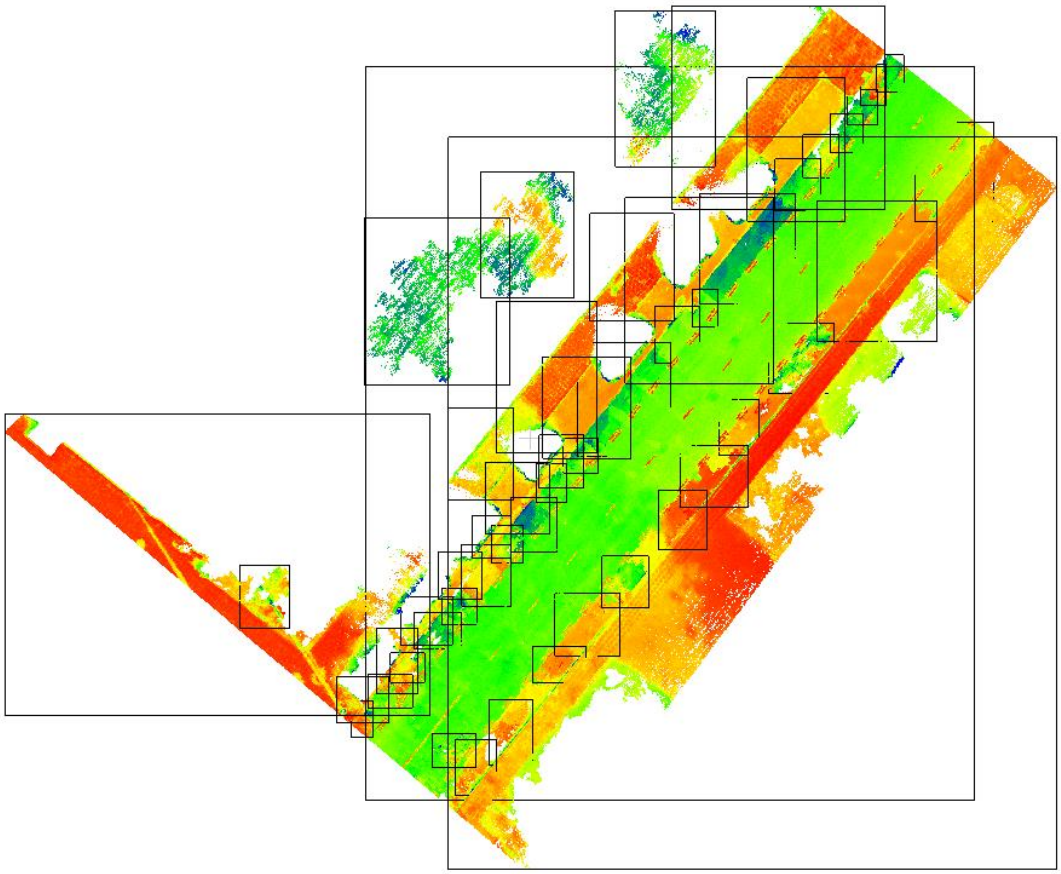


Figure 58 The results of the region merging algorithm

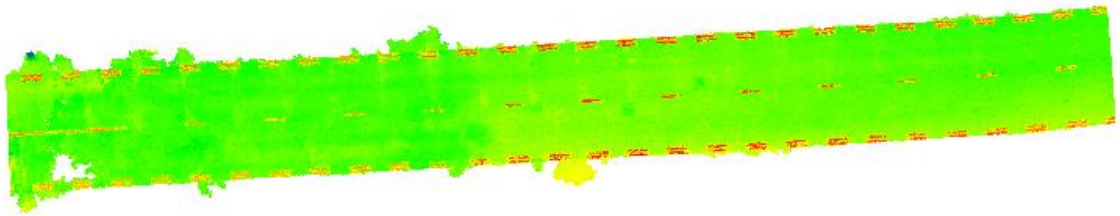


Figure 59 The merged road

9.2. Supervised Classification

The supervised approach to classifying a point cloud, requires a set of training sample data that have labels associated with them, this sample data comes from a scene and is selected with prior knowledge about the scene. This means that, for the area of interest of this project, there are points that belong to the road, cycle path, tiles and grass. Those training data sets are then used to estimate parameters of the classifier that will be used. In the developed algorithm for this project, the distance to the parallelepiped classifier is used.

9.2.1. Training samples

Collecting training samples requires a good understanding of the scene. This step is performed by the user of the algorithm. This step is very important since without

representative samples the output might not be acceptable. The point cloud used for the process has some disadvantages that were discussed in the region growing algorithm: the colour values inside the scene have a big variation that does not correspond to reality. In order to tackle with these disadvantages, more samples were carefully taken. In Figure 60 the distribution of the samples is displayed.

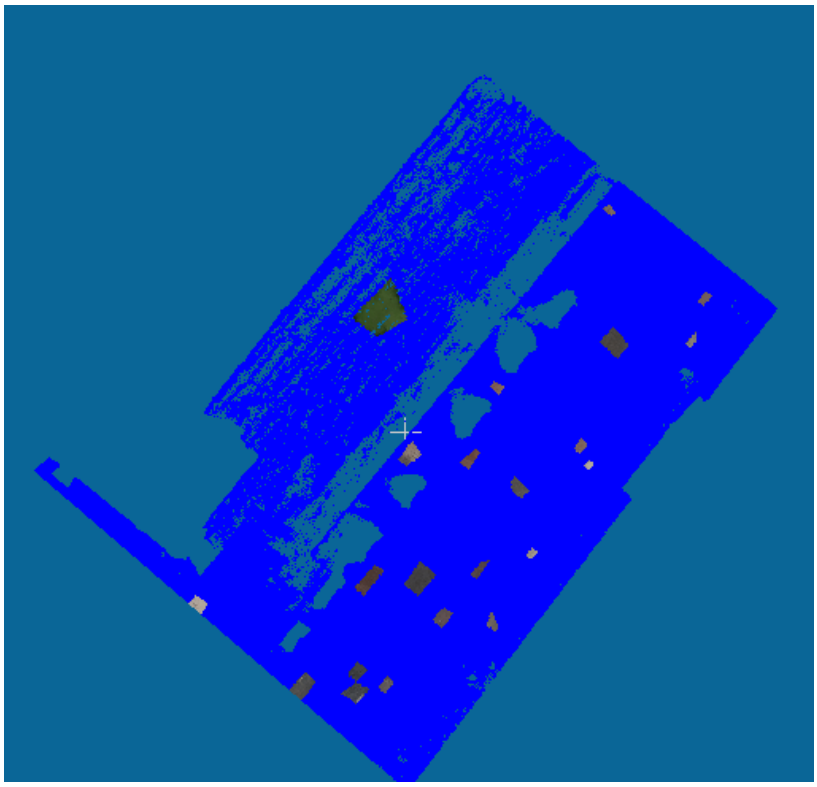


Figure 60 The training samples for the supervised classification

9.2.2. Classifier

The samples collected in the previous step are statistically analysed for their mean and standard deviation of each component of the CIELab space. For each class the statistical distribution per sample is shown in Figure 61, Figure 62, Figure 63 and Figure 64. As it is easy to realise the values have a big variation range.

Then assuming that the samples are Gaussian distributed the classifier for each sample is defined as 3 standard deviations from the mean. Therefore, for each sample an unclassified region belongs to a class if its mean colour value per component is between $\mu - 3\sigma \leq avg \leq \mu + 3\sigma$ which corresponds to 99.7% confidence level. Such a high confidence level is needed because of the noise present in the data set.

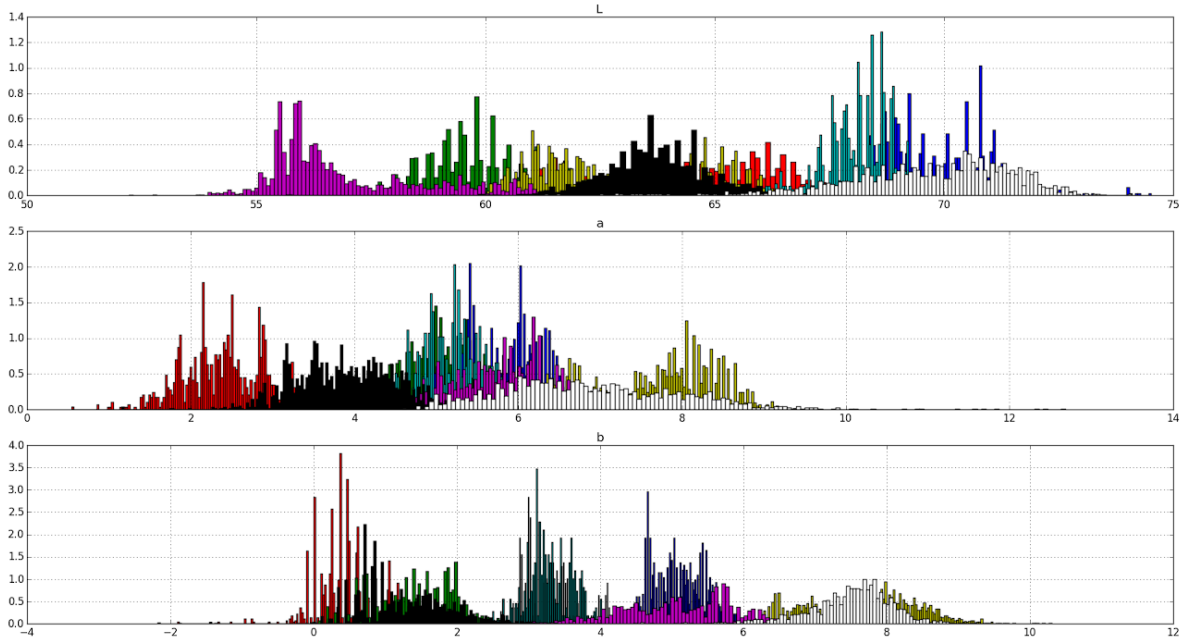


Figure 61 The distribution of the training samples for the cycle path

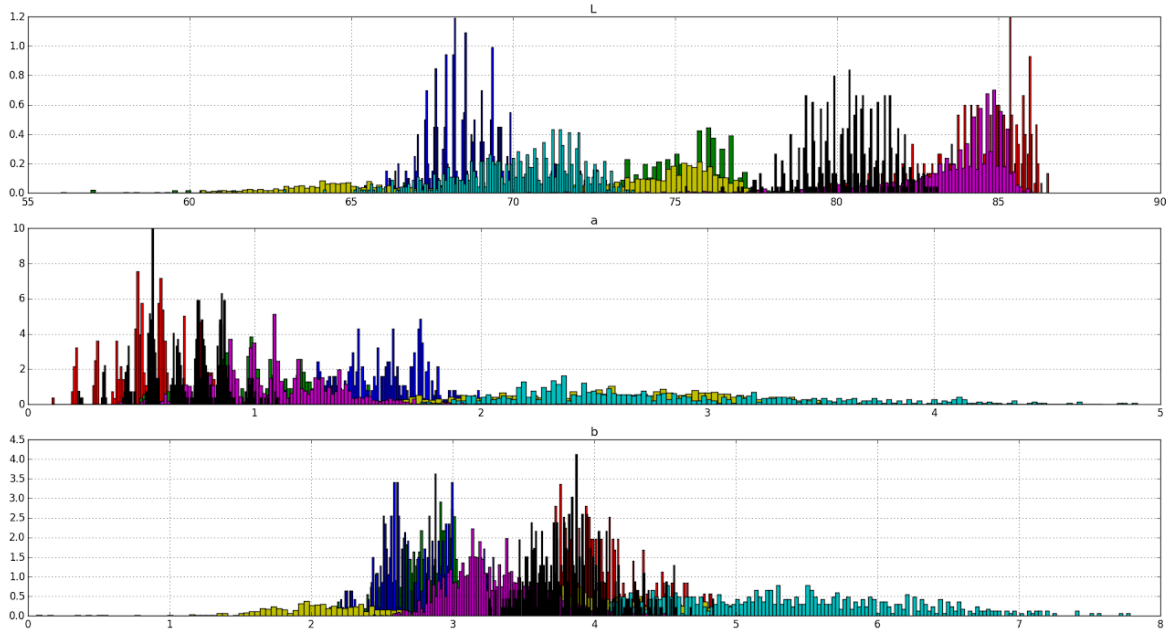


Figure 62 The distribution of the training samples for the tiles

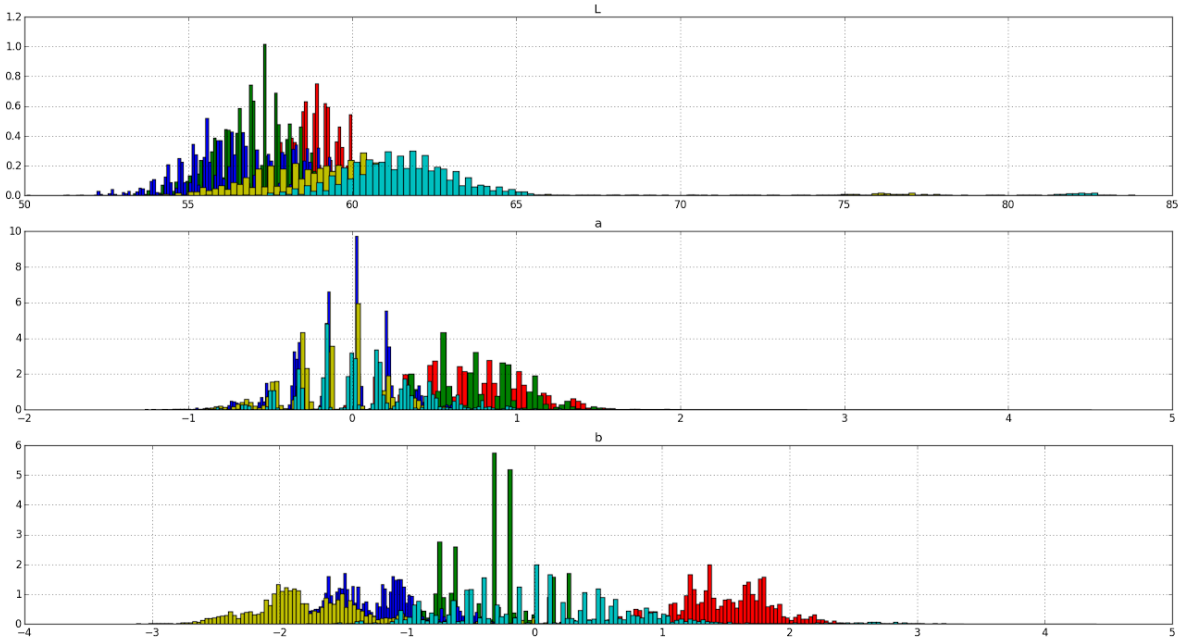


Figure 63 The distribution of the training samples for the road

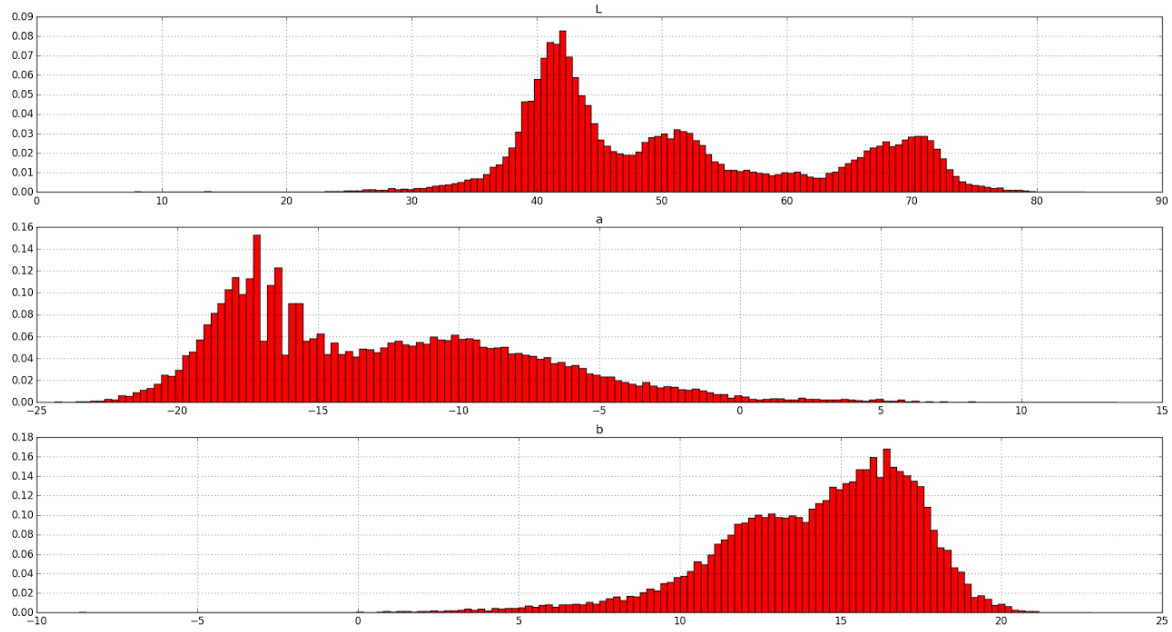


Figure 64 The distribution of the training samples for the grass

It is also defined that the samples per class are not combined into one big class. Because the colours per class have a big variation this would create the wrong statistical values. This reasoning of working could be considered as corresponding to the object based nearest neighbour classification used for image classification used in many commercial software like Trimble's eCognition.

9.2.3. Classification result

The classification algorithm designed is a collection of if… elif … else statements that compare per region the average component of the CIELab space with the limits defined by the 99,7% confidence level. If the average of the region is within

the limits of one of the samples, then the points of a region are given the appropriate classification. Of course, it is possible that for one region, the values are outside the limits of all the samples. In this case, this region is given the class unknown. The results of the supervised classification are displayed in Figure 65. Visually the results are quite representative. There are, however, misclassification problems that are present. For example, the road markings are misclassified as belonging to the tile class. This is caused because the colour information for those regions is the same as the tiles. Some artefacts are present in every class and cannot be completely avoided as no classification is 100% correct. To evaluate how well the classification performs an evaluation is performed in the next section.

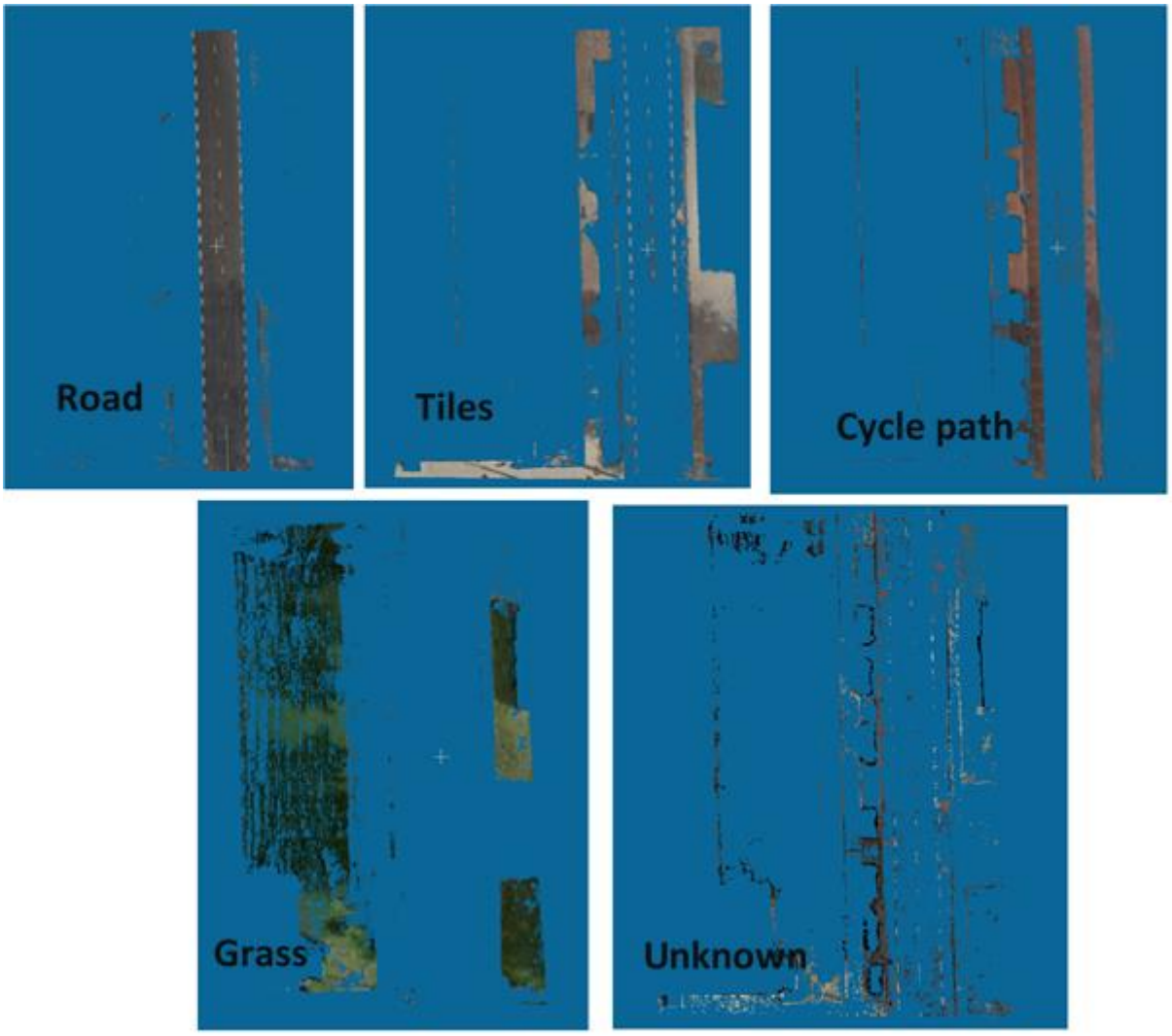


Figure 65 The results of the supervised classification per class

9.2.4. Evaluation

The evaluation of the result created by the supervised classification is performed by creating the confusion matrix (Table 13 & Table 14). In general, we can see that the most omissions are present in the case of the cycle path and tiles. The most commissions are present in the case of the cycle path. The best classified class is the grass and then the road.

Table 13 The confusion matrix for the supervised classification

		Predicted classes					
Actual Classes		Road	Cycle	Tiles	Grass	Other	Total
	Road	8652	101	206	0	14	8973
	Cycle	0	4591	2	0	926	5519
	Tiles	401	321	3760	0	2	4484
	Grass	0	0	0	5619	50	5669
	Other	0	0	0	0	0	0
	Total	9053	5013	3968	5619	992	

Table 14 The errors of omission, commission and mapping accuracy for the supervised classification

	Omissions	Commissions	Mapping Accuracy
Road	4%	4%	92%
Cycle	17%	8%	77%
Tiles	16%	5%	80%
Grass	1%	0%	99%

9.2.5. Machine learning classification

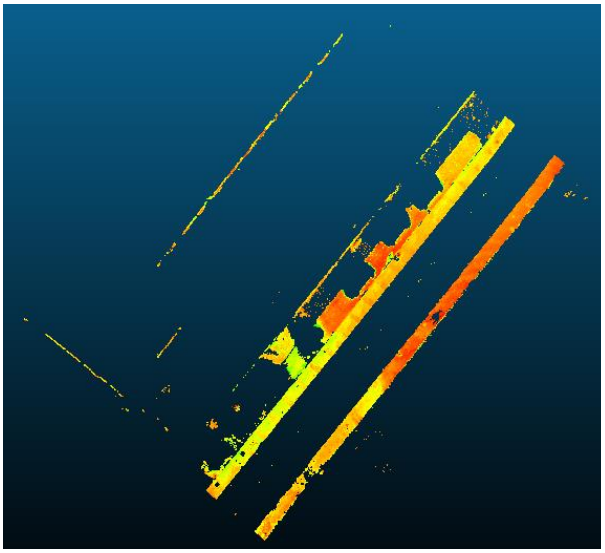
The merging of the regions is also tested using machine learning classification algorithms of the sklearn module in python. This Python module is described in earlier chapters, where individual points were classified, depending on their colour values. The algorithm creates a polygonal decision space, whereby the properties of the regions are used as classification criteria. The classification criteria are derived from samples taken from the scene. More information about the samples is provided in the next section. In the classification algorithm, the input is an array that holds 12 values:

- The minimum of the L, a and b values.
- The maximum of the L, a and b values.
- The average of the L, a and b values.
- The standard deviation of the L, a and b values

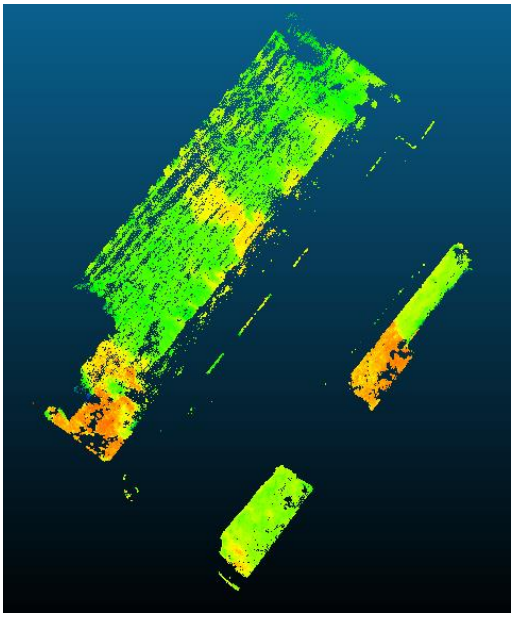
The same 12 values of the different regions are used to calculate the probability that the region belongs to a certain class.

The results of the machine learning algorithm are not satisfactory (Figure 66), this is caused by the absence of recognizing that regions do not belong to a class at all. Because some regions in the point cloud should not be assigned a class, as they don't belong to that class. It is impossible to pre-define samples of regions for regions that should not be assigned, and therefore the sklearn machine learning classification algorithm classifies with some major errors.

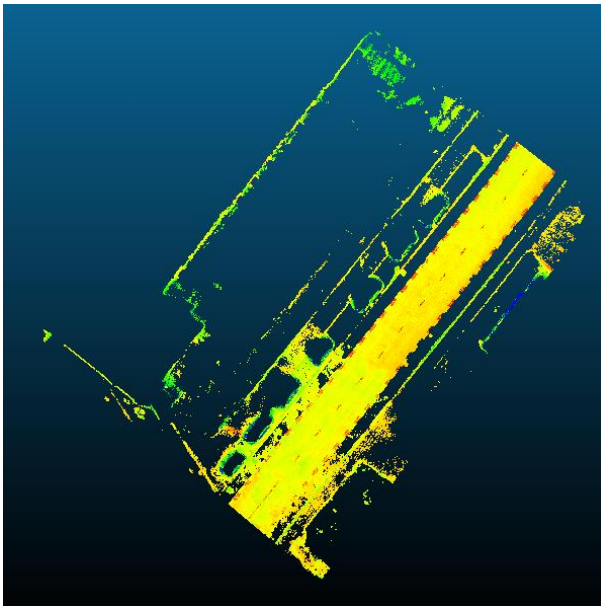
The developed algorithm is applied to the regions created in the region growing step, therefore, it is not a point-based method but a region-based one. The classification rules are based solely on the colour in the CIELab colour space.



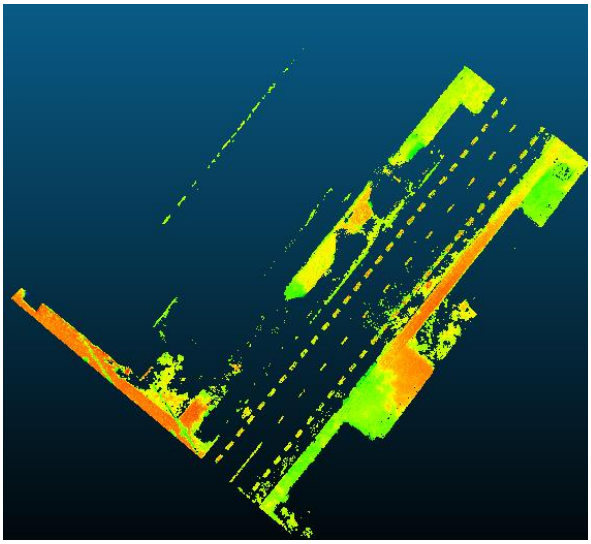
Cycle path



Grass



Road



Tiles

Figure 66 The result of the machine learning classification

9.2.6. Smoothing algorithm

To tackle the errors of omission and commission, a smoothing algorithm is developed for the supervised classification method. This method bears a resemblance on the smoothing algorithms applied to images in order to remove noise. The algorithm traverses the classified points, and for its neighbourhood it checks which is the most frequent class (road, tiles, cycle path, grass) among them. This most frequent class is then used to re-classify the point. This, of course, requires a result of adequate quality from the classification itself.

```
Initialise a list {G} for each class
For each point in the point cloud:
        Initialise a counter for each class
        For neighbour in the neighbourhood of the point
```

```
                    Check the classification of the neighbour and add one to the
                    counter of its class

            Find the most frequent class from the counters
            Change the classification of the point to the most frequent
Return all the classes
```

The updated evaluation for the smoothing algorithm in comparison to the non-smoothed situation is depicted in Table 15 and Table 16. It is apparent that, the errors of omission and commission are reduced in most of the cases. Only the error of omission for the cycle path persists. Overall, the result can be considered better not only in numbers but also visually in Figure 67, Figure 68 and Figure 69.

Table 15 The evaluation of the smoothing algorithm

		Predicted classes					
Actual Classes		Road	Cycle	Tiles	Grass	Other	Total
	Road	8973	0	0	0	0	8973
	Cycle	0	4581	0	0	938	5519
	Tiles	7	20	4457	0	0	4484
	Grass	0	0	0	5627	42	5669
	Other	0	0	0	0	0	0
	Total	9053	5013	3968	5619	992	

Table 16 The errors of omission and commission for the smoothing algorithm

	Omissions	Commissions	Mapping Accuracy
Road	0%	0%	100%
Cycle	17%	0%	83%
Tiles	1%	0%	99%
Grass	1%	0%	99%

Figure 67 The smoothing result on the road

Figure 68 The smoothing result on the tiles, cycle path and grass

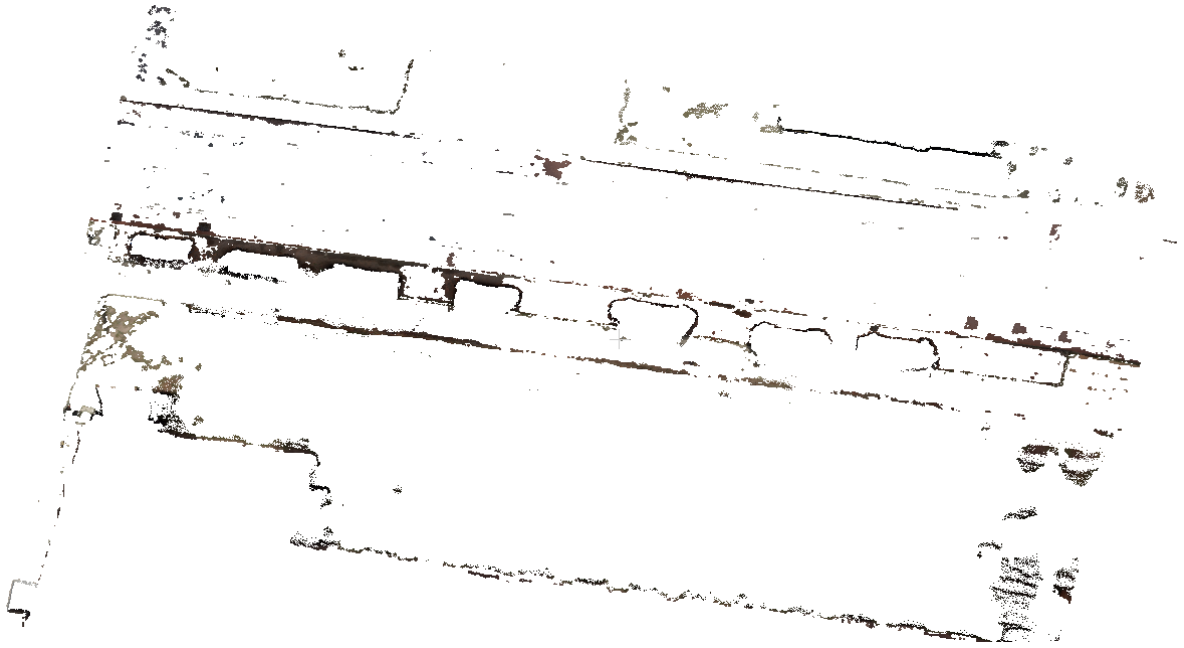


Figure 69 The smoothing result on the unknown class

9.3. The chosen method

Having applied both point based and region based techniques in order to classify the scene, it is clear that the best result is derived from the supervised classification. This can be confirmed, also, by the evaluation with the truth data and the confusion matrix, where it was obvious that a very good distinction between the classes was present. In addition to that, the errors of omission and commission are not big, that is, the classes are not misclassified. Furthermore, with the smoothing algorithm it is possible to even more reduce the misclassification errors.
However, the biggest disadvantage of this method lies in the fact that human involvement is unavoidable in this method. The success of the classification strongly depends on the experience of the user to identify the most representative samples for each class. This process at the beginning might be trial and error but at the end with the appropriate training of both the user and the algorithm the results can be really accurate. Finally, although human involvement is considered a disadvantage before, it might be the case that the result is more refined and satisfying the exact user needs compared to the unsupervised method where small regions persist or classes are mixed together.

9.3.1. Applicability to other ground point scenes

Since the method is designed for one part of the point cloud, the sub point cloud, it is also important to investigate whether it is applicable for other scenes in the total point cloud.

In Figure 70 an overview of other parts of the point cloud is shown. The red circle shows the sub point cloud, which has some different colours than other parts of the point cloud. For example, the colour of the road is not for every scene the same. Therefore, the samples made for the specific test area are not applicable for

other parts of the point cloud. For each scene specific samples must be made to distinguish the pre-defined classes. Only the class 'grass' is not based on samples and is correct in the other scenes.

If another part of the point cloud is extracted, see Figure 71, the same code and samples can be used to demonstrate that the samples are scene specific. The used part of the point cloud is the part indicated by the blue circle in Figure 70. The output is shown in Figure 72, Figure 73 and Figure 74. Here it is visible that the grass classification is still good. The road classification only gets the bus/tram road and not the road where the cars drive. The Tiles classification is quite good, but it also takes a part of the road. The classification of the cycle path only has some correct points and a lot of incorrect points, so it is therefore an incorrect classification.

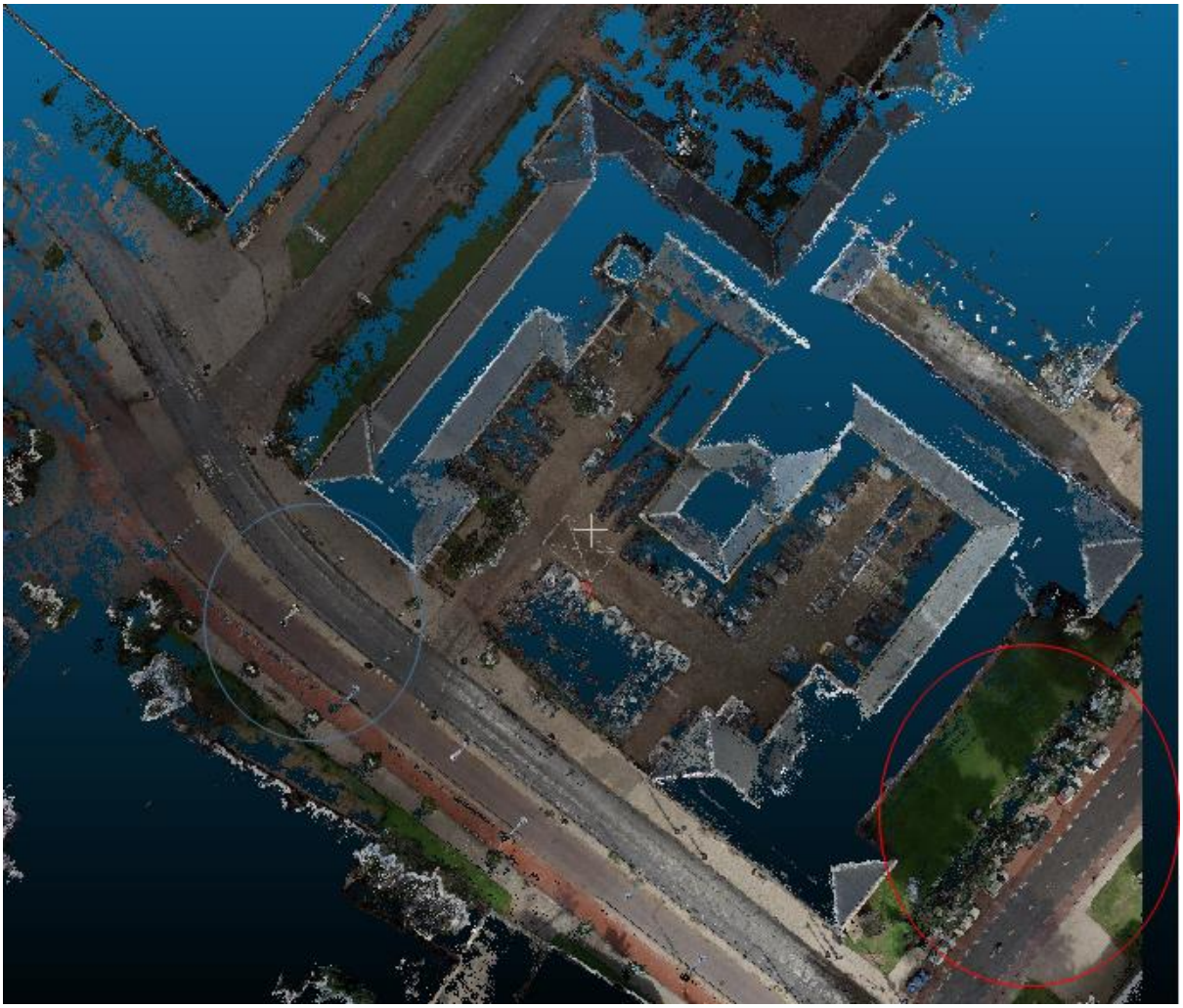


Figure 70 Other parts of the whole point cloud selected

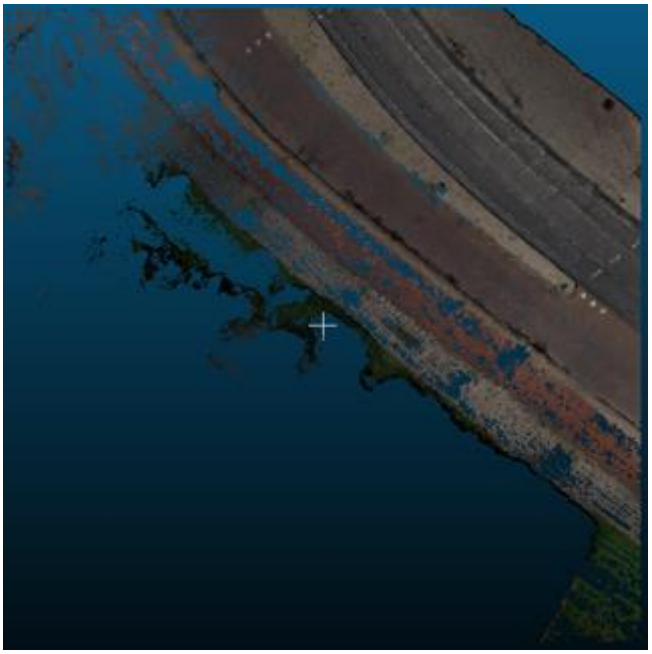


Figure 71 A different selection of point cloud

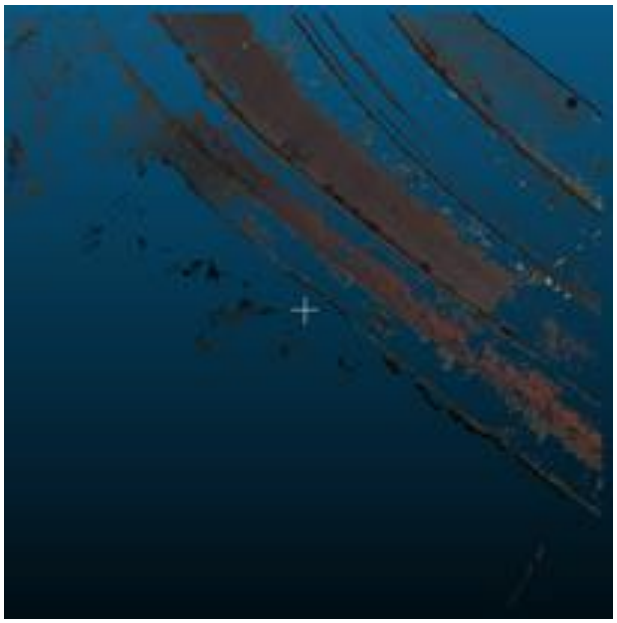


Figure 72 The unknown class of the scene

Figure 73 The cycle path and grass

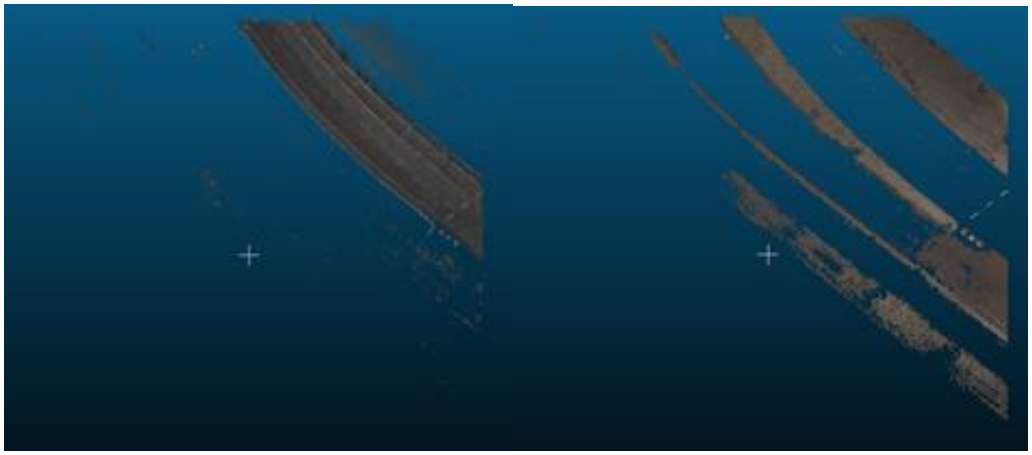


Figure 74 The road and the tiles of the scene

9.3.2. Applicability to non-ground points

Now that the method is tested for other scenes in the point cloud and is partially successful with using exact the same samples, it is also possible to use the same method to classify specific classes in the non-ground points. These non-ground points are the result of the filtered point cloud by using LAStools and were outputted in another file. Due to testing purposes two facades (Figure 75.1 and 76.1) from the sub point cloud are selected using the software package CloudCompare.

The exact same method is used (region based supervised classification), but with new samples of only the stone parts in the scene. In this way it is possible to detect the windows. According to the samples the parts of the building that consist out of stone are classified (Figure 75.2 and 76.2). The remaining parts are classified as the windows (Figure 75.3 and 76.3). Also vegetation is classified (Figure 75.4 and 76.4), according to the old samples. These are either plants within the building, or the reflection of vegetation in the windows on the pictures. In this way the method used is also useful to classify non-ground points, e.g. for façade detection.

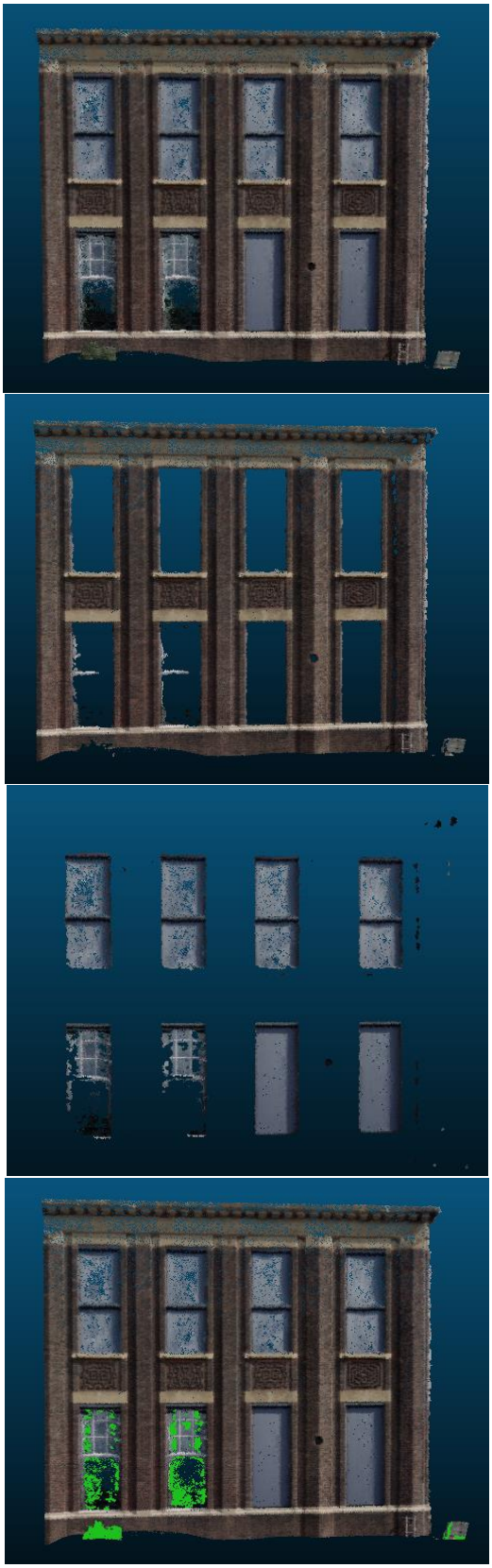


Figure 75 From top to bottom:
1. *Façade 1*
2. *Stone*
3. *Window*
4. *Impact of vegetation*

Figure 76 From top to bottom:
5. *Façade 2*
6. *Stone*
7. *Window*
8. *Impact of vegetation*

10. The viewer

For the visualisation of the semantically enriched two options were identified. The first option is that the user has a point cloud view in his/her personal computer, e.g. Cloud Compare. Therefore, this viewer can be used as a means to visualise the semantically enriched point cloud. The second option is providing the semantically enriched point cloud with a web browser. The potree library is (http://potree.org/), making use of webgl, can be used for this.

For the purposes of this project a web viewer is created to cover a wide variety of users. Unfortunately, it is not possible to yet have a layer control panel and therefore the viewer depicts the different classes with different colour. The result is provided in Figure 77. In addition to viewing, the user can perform calculate lengths, areas and height profiles very easily (Figure 78).

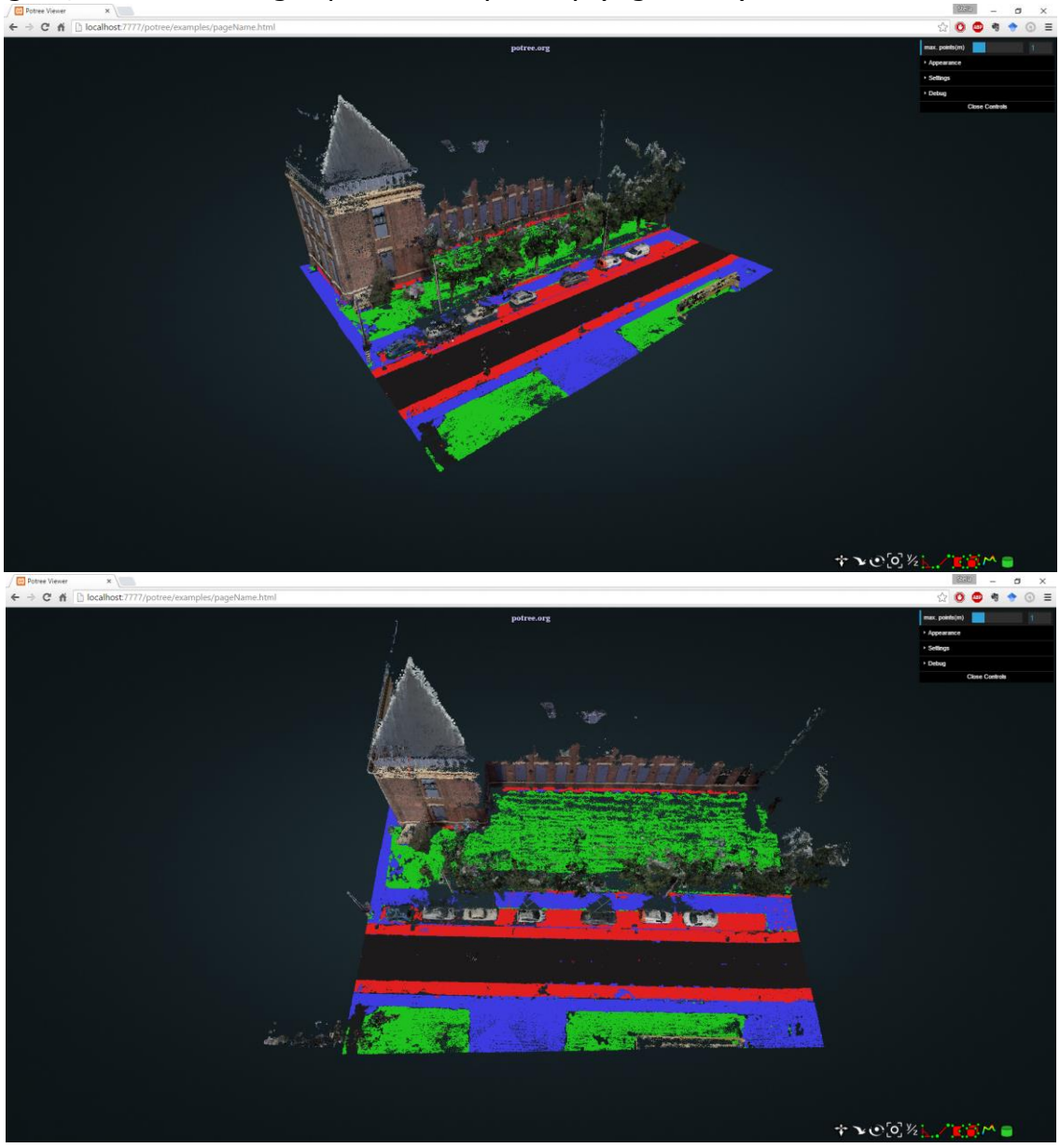


Figure 77 The potree viewer for the classified scene

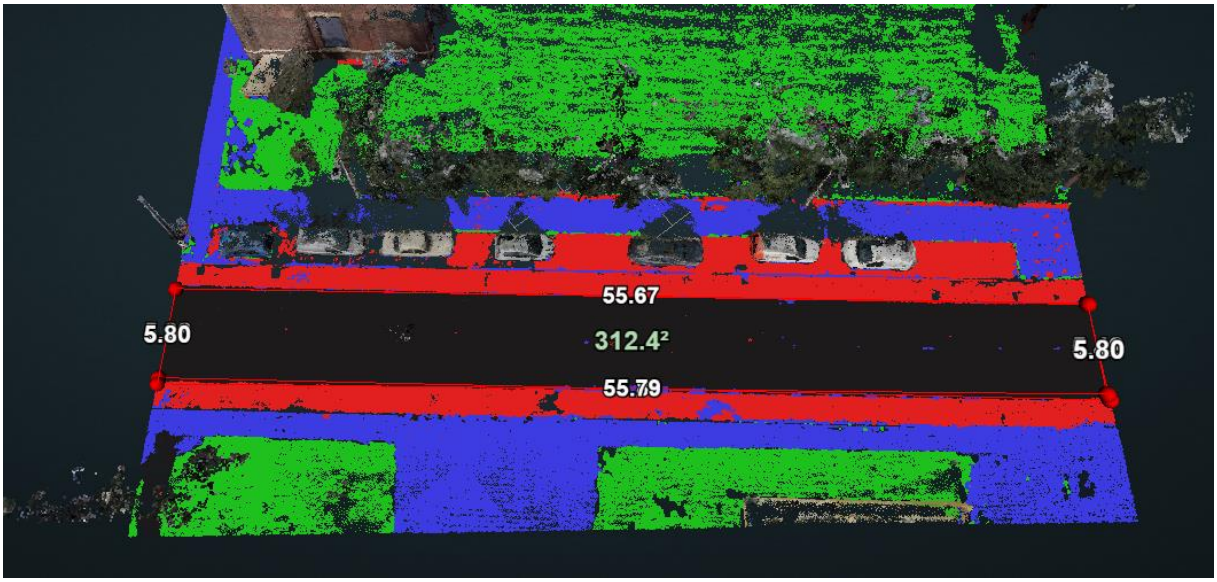


Figure 78 Calculations using the potree viewer

Conclusions

This last chapter concludes on the used techniques for semantically enriching a point cloud. In order to do so, the aim is to answer the research question: *how can a point cloud semantically be enriched by providing a labelling process, using geometrical properties and colour values?* This question is subdivided in smaller sub questions, that together form an answer on the research question.

How can the non-ground points be filtered out?

Filtering non-ground points can be challenging for urban point clouds acquired from the street level. This is mainly because many human constructions are present on the scene. In addition to that, terrestrial laser scanners can acquire millions of points in a short time period. In order to extract the ground points from the point cloud, the lasground_new tool from the LAStools suite is used. LAStools makes use of the progressive densification algorithm. The algorithm identifies ground points by progressively densifying a sparse TIN from the points. For a point to be classified as ground, it must be within the threshold of a minimum distance from the triangle that contains this point. This tool classifies the LAS or LAZ files imported into ground points and non-ground points by giving each point a label. Next, the point cloud is split on the recently added ground label, creating a ground point and non-ground point cloud.

How can a point cloud be indexed?

Point clouds are big data. For quick access and processing of the points in the point cloud, a spatial index is needed. After experimenting with the kD-tree and the octree, the kD-tree was proven faster and more efficient. This is because the kD-tree creates a balanced data structure and it is optimised for the k-nearest neighbours search.

What subcategories of ground are important?

Having a specific application in mind is very important when designing a process. In this case road extraction and materials focused on street level are examined. Therefore, the most relevant subcategories of ground (thus classes) are:

- **Grass**, or vegetation in general plays a crucial role in city analysis, urban planning, surface infiltration capacity analysis etc.. Vegetation is considered aesthetically important but also vital for the health of the city itself. Therefore, identifying the grass or the lack of grass in an urban environment can support urban planning decisions for a better life.
- **Road**, is the kernel of the transportation system in cities. The extraction of road is an important feature for governments and planning authorities. It is relevant for safety reasons, for calculating its capacity, for identifying the water resilience of the city. In addition to that, the road is the main feature for further extraction of other 'objects'. A part of the road is the **Cycle path**.

- **Tiles (or footpath)**, are also important to identify, since they are the area that protects pedestrians from cars. Therefore, this information can be used to analyse where it is hard for pedestrians to walk, important points where pedestrians have to cross roads etc.

How can subcategories be classified?

Bridging the gap between raw information and semantics can be rather challenging. Therefore, a classification scheme needs to be developed in order to provide the relevant information according to the context. The use of colour in combination with geometrical properties in today's laser scanning procedures can offer invaluable assistance in classifying objects. Within this research, different processes were tested for their relevance, namely the point based and region based classification methods.

Using point based classification methods, the colour properties seemed to be a very good criterion to distinguish between the different classes. However, there can be some overlapping colour information that will lead to misclassification of the points. This overlapping colour information can be reduced to a level by utilising a colour space different from RGB. Therefore, CIElab, HSV and HSI were examined. Point based classification with colour properties is done through analyzing the colour values. This method checks whether a point belongs to a certain class due to its colour values, while comparing them to the colour values of carefully selected training samples representing the different classes.
In using region based classification methods, neighbouring points, that hold the same geometrical and colour properties, are grouped together by the region growing algorithm. Next, these complete regions are used to give the points a label by comparing its properties to the properties of the selected trainings samples which represent the different classes.

What is the best way to label a point cloud?

In this research, where the application is surface based, the point based classification methods proved to be the less accurate even though the decision tree point based classification has a tolerable accuracy. This can be explained by the fact that a lot of colour inconsistencies are present in the point cloud, which makes it hard to classify all points correctly using just one sample for each class. Also a single point in the point cloud does not, or has less of a semantic meaning on its own and therefore it is the neighbourhood of the points that creates more valuable information and can smooth the colour inconsistencies. This contiguity is exploited by providing a segmentation technique that utilises both the geometrical and colour characteristics of the neighbourhood.

However, in the majority of the scene, the different classes, or in other words objects, are located on the same plane (like the case of cycle path and road) and therefore no immediate distinction can be made between these properties. In addition to that, there are no major differences between the geometrical properties of the grown regions, that enable clear and straightforward labelling on the ground

points of the point cloud. It can also be the case that tiles and road overlap because they are on the same level, in order for freight trucks to be able to take the ramp and, therefore, it is not possible to region grow only by curvature and normals. Normals and curvature are therefore used to region grow, but not for directly labeling the point cloud. Hence, growing regions with CIElab colour values seemed the best option because of the differences between colours that can be assessed easily by calculating the euclidean distance of the CIELab components.

From the outcomes of this project, it becomes clear that a region-based supervised classification using training samples is the best way to distinguish between the contextually relevant classes. This is because the colour information of the point cloud is often less accurate than the positional information. The colours of our point cloud show some differences in relation to reality and therefore unsupervised schemes present over-segmented results. Furthermore, over-segmentation within the unsupervised classification can caused by the existence of shadow and water on surfaces, which then appear darker, worn out colours of the road/cyclepath markings, different materials of tiles and other noise in general. To work around those inaccuracies, training samples can be provided. This way a supervised classification captures all appearances of a class whereas an unsupervised classification would create different regions. However, during the project, the researchers realised that one training sample per class is not sufficient. Therefore, the user has to provide the best combination of training samples per scene that will not be excessive in number and amount, but representative enough of the distinction of the classes present. Finally, extending the method to other areas can be successful, but only if the provided training samples are representative. The schematic overview of the final workflow is depicted in Figure 79.

Finally, the best way to provide the labelled point cloud to its users depends on the application and the characteristics of the users of the product. For more inexperienced users, providing different las files for each class might be easier to visualise and understand. For more experienced users, one las file with the classification stored as an attribute might be more convenient and easier, and form a basis to build more applications on top of it.

Recommendations

Unsupervised classification of point clouds results in multiple regions which are similar on chosen properties. A disadvantage lies in the fact that no semantics is given to these regions, it is only known that they probably belong to the same feature or surface. It is up to the user to select what is relevant. To enrich such a result we propose to explore the possibilities to match the regions with land use maps of some sort. This mapping technique could have the possibilities to create maps with a very high resolution.

Another interesting option is to provide the positional information of the mapping vehicle/device with (each part of) the point cloud. This way, the position can be matched with the processed regions, and for example, if we know that this position

relates to a car, we can assume that the concerning region is accessible with a car, i.e. we classify it as road.

Finally, the colour based region growing would be greatly improved if some simple contrast and brightness enhancement techniques would take place before the generation of the point cloud from the cycloramas. This would minimise the shadowing problems and the quality of the colours present in the point cloud, leading to less training samples being needed per class.

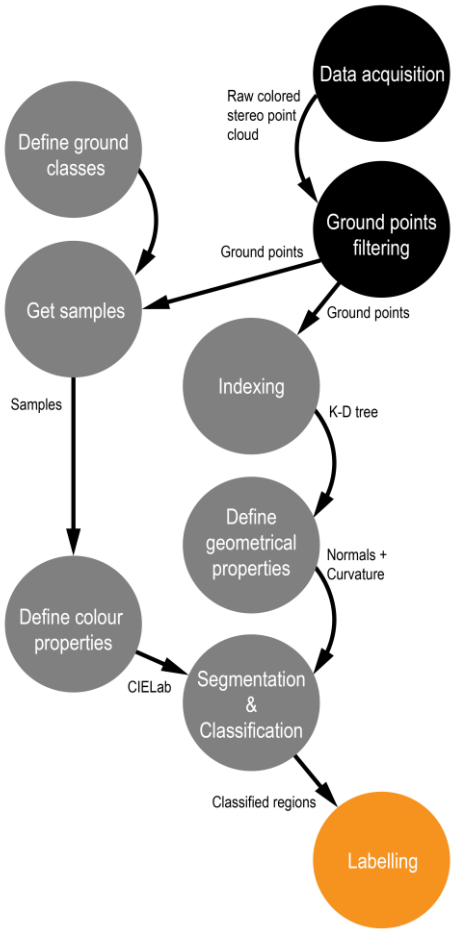


Figure 79 The schematic overview of the developed workflow

Future work

A next step would be to extend the workflow to fit a larger region like the Netherlands. In order to make this feasible ways of automatic sampling need to be investigated. This way the method of supervised classification as described in this research would be scalable.

In the trend of explorative point clouds a next step could be to relate the point clouds, in particular semantically labelled parts of a point cloud to over simplified models of reality. For example, for the inspection of the road network the maintenance department would check a map (google maps), and by clicking on the map would retrieve the (most up-to-date) detailed point cloud that correlates to this area. In the case of an urban planner, this would mean that the point cloud

would be matched with the 3D models that currently exist, like the 3D model of the netherlands based on AHN2. A possibility would be to store the points in an octree structure and find intersections/overlaps with these over simplified models. The client/browser would then only retrieve the semantically relevant points, which is desirable since the processing power of the current equipment is still insufficient to work with the whole point cloud at once.

References

Axelsson, P. (2000) 'DEM generation from laser scanner data using adaptive TIN models', International Archives of Photogrammetry and Remote Sensing, 33(B4/1; PART 4), pp. 111-118.

Bean, E. Z., Hunt, W. F., Bidelspach, D. A. and Smith, J. 'Study on the surface infiltration rate of permeable pavements'. Proceedings of the American Society of Civil Engineers and EWRI 2004 world water and environmental resources congres, Salt Lake City, UT, USA.

Behley, J., Steinhage, V. and Cremers, A. B. 'Efficient Radius Neighbor Search in Three-dimensional Point Clouds'.

Beers, B., (2015), personal contact about the procedures of Cyclomedia.

Bolles, R. C., & Fischler, M. A. (1981, August). A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data. In IJCAI(Vol. 1981, pp. 637-643).

Borrmann, D., Elseberg, J., Lingemann, K., & Nüchter, A. (2011). The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. 3D Research, 2(2), 1-13.

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), pp. S2-S12.

Butler, H. and Gerlek, M. 'PDAL-Point Data Abstraction Library. 2014', URL: http://www.pdal.io/index.html.

Cheng, H.D., Jiang, X.H., Sun, Y. and Wang, J., (2001). Color image segmentation: advances and prospects. Pattern Recognition, 34 (12): 2259-2281.

Cristiani, N. (2001) Support Vector and Kernel Machines. [online] http://www.support-vector.net/icml-tutorial.pdf

Cortes, C. and V Vapnik (1995) Support Vector Network. Machine Learning, 20, 273-297. Kluwer Academic Publishers, Boston. Manufactured in The Netherlands. Choudhury, A. K. R. (2014). Principles of Colour and Appearance Measurement: Object Appearance, Colour Perception and Instrumental Measurement. Elsevier.

Elseberg, J., Borrmann, D. and Nüchter, A. 'Efficient processing of large 3d point clouds'. Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on: IEEE, 1-7.

Elseberg, J., Magnenat, S., Siegwart, R. and Nüchter, A. (2012) 'Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration', Journal of Software Engineering for Robotics, 3(1), pp. 2-12.

Evans, J. S. and Hudak, A. T. (2007) 'A multiscale curvature algorithm for classifying discrete return lidar in forested environments', Geoscience and Remote Sensing, IEEE Transactions on, 45(4), pp. 1029-1038.

Ford, A., & Roberts, A. (1998). Colour space conversions. Westminster University, London, 1998, 1-31.

Geosignum (2015) 'Geosignum website. [online] < http://geosignum.nl/ > (last visited on september 21 2015) '.

Hirschmuller, H. (2005) Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual. Information. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, June 20-26.

Hsu, C.W., Chang, C.C. and Lin, C.J. (2010) A practical guide to support vector classification. Department of Computer Science, National Taiwan University, Taipei. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

James, R. C., & James, G. (19 92). The Mathematics Dictionary. Springer Science & Business Media.

Jolliffe, I. T., (1986). Principal Component Analysis, Series: Springer Series in Statistics, Springer, NY, 1986, ISBN 978-1-4757-1906-2.

Jolliffe, I. T., (2002). Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, 2002, XXIX, 487 p. 28 illus. ISBN 978-0-387-95442-4.

Kabali, M., Stroila, M., Cho, J., Shaffer, E. and Hart, J.C., Robust Classification of Curvilinear and Surface like Structures in 3D Point Cloud Data. University of Illiniois Urbana-Champaign, NAVTEQ

Kammerl, J. and Muja, M. (2011) 'PCL :: Search, PointCloudLibrary. [online] http://www.pointclouds.org/assets/rss2011/06_search.pdf'.

Korzeniowska, K., Pfeifer, N., Mandlburger, G. and Lugmayr, A. (2014) 'Experimental evaluation of ALS point cloud ground extraction tools over different terrain slope and land-cover types', International Journal of Remote Sensing, 35(13), pp. 4673-4697.

Labatut P. et al (2007) Efficient Multi-View Reconstruction of Large-Scale Scenes using Interest Points, Delaunay Triangulation and Graph Cuts.

Lai, J. Z., Liaw, Y.-C. and Liu, J. (2007) 'Fast k-nearest-neighbor search based on projection and triangular inequality', Pattern Recognition, 40(2), pp. 351-359.

Landa, J. and Prochazka, D. (2014) 'Automatic Road Inventory Using LiDAR', Procedia Economics and Finance, 12, pp. 363-370.

LASTools (Accessed on 18/09/2015) 'http://rapidlasso.com/'.

Lee, D. (2013) Mapping Urban Surface Infiltration Capacity: Segment-based land cover classification with VHR imagery for urban water management and design. TU Delft, Delft University of Technology.

Luccheseyz, L., & Mitray, S. K. (2001). Color image segmentation: A state-of-the-art survey. Proceedings of the Indian National Science Academy (INSA-A),67(2), 207-221.

Maneewongvatana, S., & Mount, D. M. (2001). On the efficiency of nearest neighbor searching with data clustered in lower dimensions (pp. 842-851). Springer Berlin Heidelberg.

Mäntylä, M. (1987) An introduction to solid modeling. Computer Science Press, Inc., p. 1.

Mathworks (2015) Supervised learning. [online] < http://nl.mathworks.com/discovery/supervised-learning.html > (last visited on october 2, 2015)

Matti, E. K. and Nebiker, S. (2014) 'Geometry and Colour Based Classification of Urban Point Cloud Scenes Using a Supervised Self-Organizing Map', Photogrammetrie-Fernerkundung-Geoinformation, 2014(3), pp. 161-173.

Nguyen, A., & Le, B. (2013). 3D point cloud segmentation: A survey. In Robotics, Automation and Mechatronics (RAM), 2013 6th IEEE Conference on (pp. 225-230). IEEE.

Niemeyer, J., Rottensteiner, F. and Soergel, U. (2012) Conditional Random Fields for Lidar Point Cloud Classification in Complex Urban Areas. ISPRS Annals of the Photogrammetry, Remote sensing and Spatial information Sciences, Vol. 1-3, 2012.

Oude Elberink, S. and H. Maas, 2000. The use of anisotropic height texture measures for the segmentation of airborne laser scanner data. IAPRS, vol. 33, part B3, Amsterdam, pp. 678- 684.

Pfeifer, N. and Mandlburger, G. (2009) 'LiDAR data filtering and DTM generation', Topographic Laser Ranging and Scanning: Principles and Processing. CRC/Taylor & Francis, pp. 307-333.

Point Cloud Library (PCL) - Estimating Surface Normals in a PointCloud: Available at: http://bit.ly/1YQBvnc (Accessed: October 1, 2015).

Quora (2014) What are the differences between RGB, HSV and CIE-lab? [online]https://www.quora.com/What-are-the-differences-between-RGB-HSV-and-CIE-Lab

Rabbani, T., van den Heuvel, F., & Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36(5), 248-253.

Rokach, L. and Maimo, O. Decision Trees. Department of industrial engineering Tel-Aviv University. [online] http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf

Rusu R. B., 2010. Semantic 3d object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz, 24(4)

Rusu, R. B. and Cousins, S. '3d is here: Point cloud library (pcl)'. Robotics and Automation (ICRA), 2011 IEEE International Conference on: IEEE, 1-4.

Samadzadegan, F., Bigdeli, B. and Ramzi, P. Classification of Lidar data based on Multi-Class SVM. Department of Geomatics Engineering, Faculty of Engineering, University of Teheran, Iran.

Sankaranarayanan, J., Samet, H. and Varshney, A. (2007) 'A fast all nearest neighbor algorithm for applications involving large point-clouds', Computers & Graphics, 31(2), pp. 157-174.

Sapkota, P.P. (2008). Segmentation of Coloured Point Cloud Data. MSC Thesis. International Institute for geo-information science and earth observation, Enschede, Netherlands.

Sareen, K., KNOPF, G. & CANAS, R., 2010: Rapid Clustering of Colorized 3D Point Cloud Data for Reconstructing Building Interiors. ISOT 2010 International Symposium on Optomechatronic Technologies, Toronto, Canada: 1–6.

Schnabel, R., Wahl, R., & Klein, R. (2007, June). Efficient RANSAC for point-cloud shape detection. In Computer graphics forum (Vol. 26, No. 2, pp. 214-226). Blackwell Publishing Ltd.

Shan, J.; Sampath, A. Urban DEM generation from raw Lidar data: a labeling algorithm and its performance. Photogramm. Eng. Remote Sensing 2005, 71, 217–226.

Shekhar, S. and Xiong, H. (2008) 'Light Detection and Ranging', Encyclopedia of GIS, pp. 612-612.

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(1), pp. 85-101.

Smadja, L., Ninot, J. and Gavrilovic, T. (2010) 'Road extraction and environment interpretation from Lidar sensors', IAPRS, 38, pp. 281-286.

Snavely, N., Seitz, S. M. and Szeliski, R. 'Photo tourism: exploring photo collections in 3D'. ACM transactions on graphics (TOG): ACM, 835-846.

Sorkine, O., (2013): Normal Estimation in Point Clouds. 2D/3D Shape Manipulation, 3D Printing.

Sutton, O. (2012) Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction. [online] http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN_Talk.pdf

Tinkham, W.T.; Huang, H.; Smith, A.M.S.; Shrestha, R.; Falkowski, M.J.; Hudak, A.T.; Link, T.E.; Glenn, N.F.; Marks, D.G. A Comparison of two open source LiDAR surface classification algorithms. Remote Sens. 2011, 3, 638–649.

van Oosterom, P. (1999) 'Spatial access methods', Geographical information systems, 1, pp. 385-400.

Velodyne Lidar (2015) HDL 32E. [online] < http://velodynelidar.com/ > (last visited in november 13 2015)

Vosselman, G., & Dijkman, S. (2001). 3D building model reconstruction from point clouds and ground plans. International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences, 34(3/W4), 37-44.

Vosselman, G. and Maas, H.-G. 2010. Airborne and Terrestrial Laser Scanning. Whittles Publishing.

Waldhauser, C., Hochreiter, R., Otepka, J., Pfeifer, N., Ghuffar, S., Korzeniowska, K., & Wagner, G. (2014). Automated classification of airborne laser scanning point clouds. In Solving Computationally Expensive Engineering Problems (pp. 269-292). Springer International Publishing.

Wang, G., Li, M. and Zhou, T. (2012) The automatic classification 3d point clouds based associative markov network using context information. In: He, X. et al. (eds), computer Informatics, Cybernetics and Applications, Lecture notes in electrical engineering 107. Springer Science+Business Media B.V. 2012

Weinmann, M., Jutzi, B. and Mallet, C. (2014) 'Semantic 3D scene interpretation: a framework combining optimal neighborhood size selection with relevant features', ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 3, pp. 181-188.

Weinmanna, M., Schmidtb, A., Malletc, C., Hinza, S., Rottensteinerb, F. and Jutzia, B. (2015) 'Contextual Classification of Point Cloud Data by Exploiting Individual 3d Neigbourhoods', ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, 1, pp. 271-278.

Westoby, M., Brasington, J., Glasser, N., Hambrey, M. and Reynolds, J. (2012) ''Structure-from-Motion'photogrammetry: A low-cost, effective tool for geoscience applications', Geomorphology, 179, pp. 300-314.

Wikipedia (A) (2015) Supervised learning. [online] < https://en.wikipedia.org/wiki/Supervised_learning > (last visited on october 5 2015)

Wu, C.C. and Lin, S.F. (2011) Efficient Model Detection in Point Cloud Data Based on Bag of Words Classification. Journal of Computational Information Systems 7: 12 (2011) 4170-4177

Wu, C., (2013) 'Towards linear-time incremental structure from motion'. 3D Vision-3DV 2013, 2013 International Conference on: IEEE, 127-134.

Yang, B., Fang, L., Li, Q. and Li, J. (2012) 'Automated extraction of road markings from mobile LiDAR point clouds', Photogrammetric Engineering & Remote Sensing, 78(4), pp. 331-338.

Zhana, Q., Liangb, Y., & Xiaoa, Y. (2009). Color-based segmentation of point clouds. Int Arch Photogrammetry, Remote Sens Spat Inf Sci, 38, 248-252.

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', Geoscience and Remote Sensing, IEEE Transactions on, 41(4), pp. 872-882.

Zhang, J., Lin, X. and Ning, X (2013) SVM-Based Classification of Segmented Airborne LiDAR Point Clouds in Urban Areas. Remote Sens. 2013, 5, 3749-3775

Appendix A - Project management and process

A.1. The Team

Adrie Rovers (Technical Manager), Irene de Vreede (Report Manager), Merwin Rook (Quality Manager), Stella Psomadaki (Coordinator) and Tim Nagelkerke (Communicator)

A.2. Technical Requirements

2.1. What is known? What is unknown?

Table 1 The known and unknown requirements

Known
Software: • PostgreSQL and PostGIS (Making a (geo-)database) • Python (Programming) • CloudCompare (Processing & Visualizing point clouds) • FME (Process point clouds) • Google Drive (Sharing documents) Knowledge, information: • Data acquisition (Cyclomedia, pictures and lidar) • Importing .xyz files in Python Organisational: • The capabilities of the team members
Unknown
Software: • LAS tools • Python to PostGIS connection • Point cloud extension of PostGIS • Other "new" software applicable to the project, for handling point clouds. • Use of Point Cloud Library (http://pointclouds.org/) Knowledge, information: • The test area that is going to be used from the point clouds • The difference between the two point clouds provided by Cyclomedia • The data formats used by Cyclomedia • The attributes of the points of the provided point clouds • How to derive the area of the surface out of the points • The use of geometric properties of points in a point cloud • Indexing and clustering point clouds in Python or PostGIS • How to classify the different surfaces (grass, soil, road, stones, pavement etc.) • How to perform the labelling of the point cloud Organisational:

- The schedule of the team members

9.4. Identification and contact with stakeholders and experts

Possible Clients
Municipality of Delft, to automatically detect the road and urban surface and utilise it for maintenance purposes.
City planners, in order to find weak spots in the city water management.
Facilitair Management & Vastgoed TU Delft, to keep track of the facilities' status around the campus.

Domain Experts

The main expert group is Cyclomedia. Cyclomedia performs the data acquisition using mobile mapping techniques and processes the data into a manageable format.
Geosignum could give some basic advice about how to start the process.
Geomatics Msc. personnel could provide their ICT and Geomatics expertise.

9.5. MoSCoW prioritization

MoSCoW is a method for assisting the team in understanding the priorities of the project. The letters stand for:

- **M**ust Have: A must have is a critical component of the software. If one of the must haves is missing, the project can be considered a failure.
- **S**hould Have: A should have is also important, but is not a necessary for the project.
- **C**ould Have: Could haves are desirable, but not necessary for the project.
- **W**on't Have this time: A won't have is not planned in the schedule, and can be considered in further developments.

The MoSCoW prioritisation for this project is depicted in Table 2.
Table 2, The MoSCoW prioritisation

MUST have this	**SHOULD have this if at all possible**
• Data gathering: data gathering is the most important task in the project, as the data forms the basis for our analysis. The data gathering involves the point cloud from the laser scanner and the point cloud from Cycloramas. • Filter the point cloud: separating the ground points from buildings, vegetation and other objects is one of the main goals of the project, as the ground points are	• Cyclorama images.

the main objects of interest in our analysis. • Classify the point cloud: the classification of the point cloud means that point that have similar (geometri or colourc) properties are recognised and classified the same. • Semantically label the point cloud: the labelling of the points gives a meaning to the points and the point cloud.	
COULD have this if it doesn't affect anything else	**WON'T have this this time but WOULD like in the future**
• Applying the labelling method on different point clouds to see if it is possible to apply this method for the whole of the Netherlands. • Identifying involved parties/ applications: in order to optimise the use of points, new stakeholders and their demands must be identified. • Applications: New applications help the demand and development of classification and labelling algorithms to develop and evolve. • A comparison of the two point clouds provided by Cyclomedia.	• Database of the original point clouds. • Database design: the semantically labelled points should be stored, in order to disseminate the points to the actual users of the data. • Classification and labelling of non-ground points. • Semantic point cloud viewer; In order to present and select the points and their semantics, a (web)viewer should be developed. This viewer should give users the possibility to show and select points on their semantic meaning. This could preferably be done in WebGL. • A website presenting the project's results. • Calculations of the (expected) amount of sunlight in hours, in order to calculate exact heat effects. • The street surface of the whole of the Netherlands.

9.6. Killer requirements

A killer requirement is a requirement that, if it is not met, brings the whole project in danger. Identifying the killer requirements could lead to a steady project execution.

- Processing power of personal laptops.
- Obtaining the data on time.
- A minimal accuracy of the geometry and colour attributes of the point cloud in order to distinguish between classes.
- Time limitation.
- Programming language knowledge

9.7. Competitors evaluated

A competitor can be defined as a party providing the same products or services as .XYZ solutions.
The other synthesis project teams, all handling raw point cloud data for certain applications, are our competitors. Although companies like GeoSignum are our main competitors. Geosignum is a small innovative startup company which provides an automatic feature extraction technology from LIDAR datasets. Feature extraction can be seen as a form of semantic labelling and therefore this company could provide the same service. The way they automatically extract features is done using only the geometric properties of the points within the point cloud, in this way the provided service is slightly different.
Since processing the raw point cloud for certain applications is quite a new way of providing certain products or services, lots of research is taking place in this domain but not many companies already provides such products or services.

9.8. Rich picture

The rich picture represents a schematic view of the problem that will be solved during the project. The main factors that affect the problem are presented and their interactions are displayed. The rich picture of the project in this report is composed as follows (Figure 1): The team .XYZ is in the middle and is the basis for the whole view. The team members want to apply their knowledge obtained the last year to compose a project that will bring great value to the community. In addition to that, they want to have fun during the whole process. The coaches want and encourage the students to explore new ways to use point clouds as the main source of information. There are also the rules and the deadlines that need to be followed and the team has to deal with those time limitations. The competitors, just like team .XYZ want to make the best out of their product, eliminating the competition if possible. Three types of reports need to be made which have to cover specific points and be as clear as possible. The most important aspect is then the clients and users. This group of people want much as detailed information as possible, to save resources and solutions that are feasible in the everyday life. The media outreach, on the other hand, is interested in promoting new ideas and new scientific results. Cyclomedia (the data provider) is aiming at helping the education and to make their products more appealing to new kinds of users. To do this, they provide state of the art mobile mapping solution. Finally, the result of this project should be a user friendly and automated as possible.

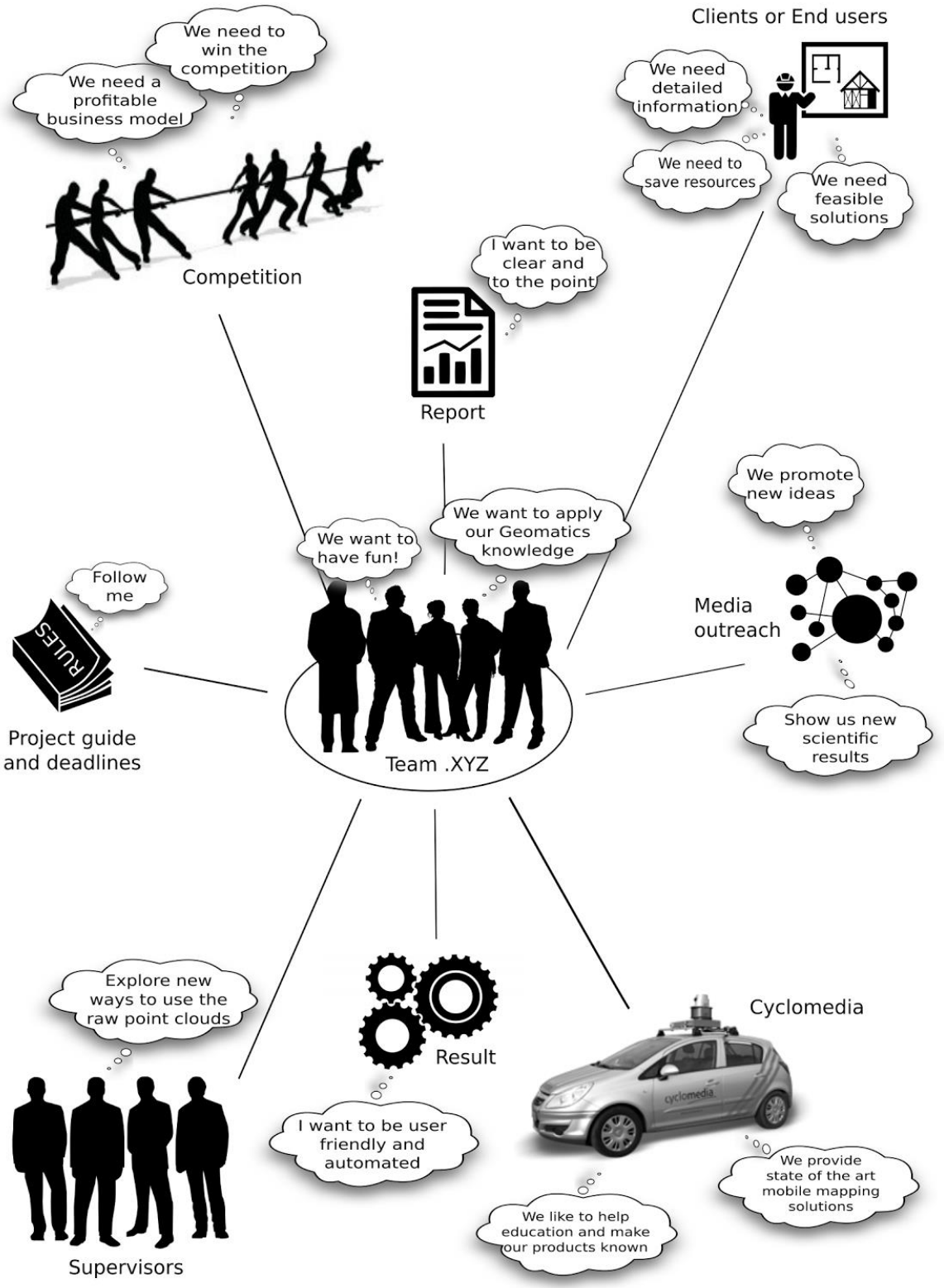


Figure 1, The Rich Picture

A.3. Project Plan

The purpose of the project plan is to strategically define the process towards the completion of the project. The project plan takes place at the early stages of the

project to identify all the requirements, resources and time required for the execution of the work. Lack of a project plan might lead to the failing of the project.

3.1. Organisational Breakdown Structure

Prosperous project management requires detailed planning and an organized structure. When starting a project, tasks have to be identified and decisions have to be made of who will carry out the work. The Organisational Breakdown Structure (OBS) is a model which allows projects organisations to be broken down, providing a framework for project planning. This framework is created to reflect the strategy for managing various aspects of the project. Responsibilities within the project management will be defined for every team member. Team member tasks will be linked to the project management as well as to work packages defined within the Work Breakdown Structure (Section 5.2). The Organisational Breakdown Structure is presented in Figure 2.

The team

Adrie Rovers (Technical Manager), Irene de Vreede (Report Manager), Merwin Rook (Quality Manager), Stella Psomadaki (Coordinator) and Tim Nagelkerke (Communicator)

Team tasks and functions

Coordinator - Defines the Agenda, Leads meetings.
Communicator - Defines the planning, Does the communication between the production team and client, Sets meetings between the project team and the client and or coaches.
Quality Manager - Defines the overall work quality, Reviews the quality of delivered work.
Report Manager - Defines report and document requirements, deals with the organisational aspects regarding the deliverable items.
Technical Manager - Deals with technical aspects, Defines algorithms.
The work packages as defined in section (5.2 and 5.3) are performed by all team members. This will ensure that the learning process has an equal effect to all team members and all members work equally throughout the project. Nevertheless the person first mentioned will be the final responsibility.

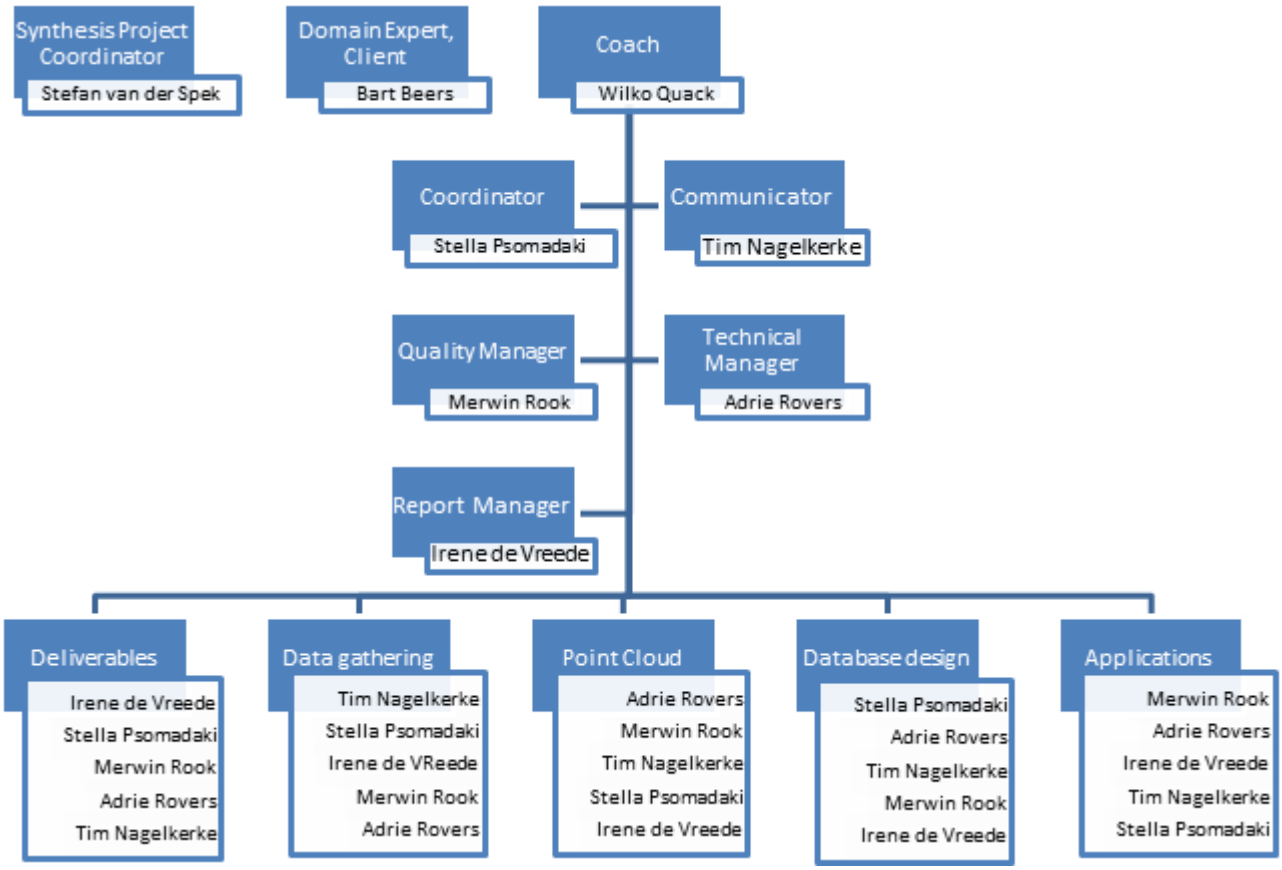


Figure 2, The Organisational Breakdown Structure

3.2. Work Breakdown Structure

The WBS is used to provide the team members a common view of the project scope. In other words, WBS is used to analyse the aim of the project according to its outcomes or deliverables.

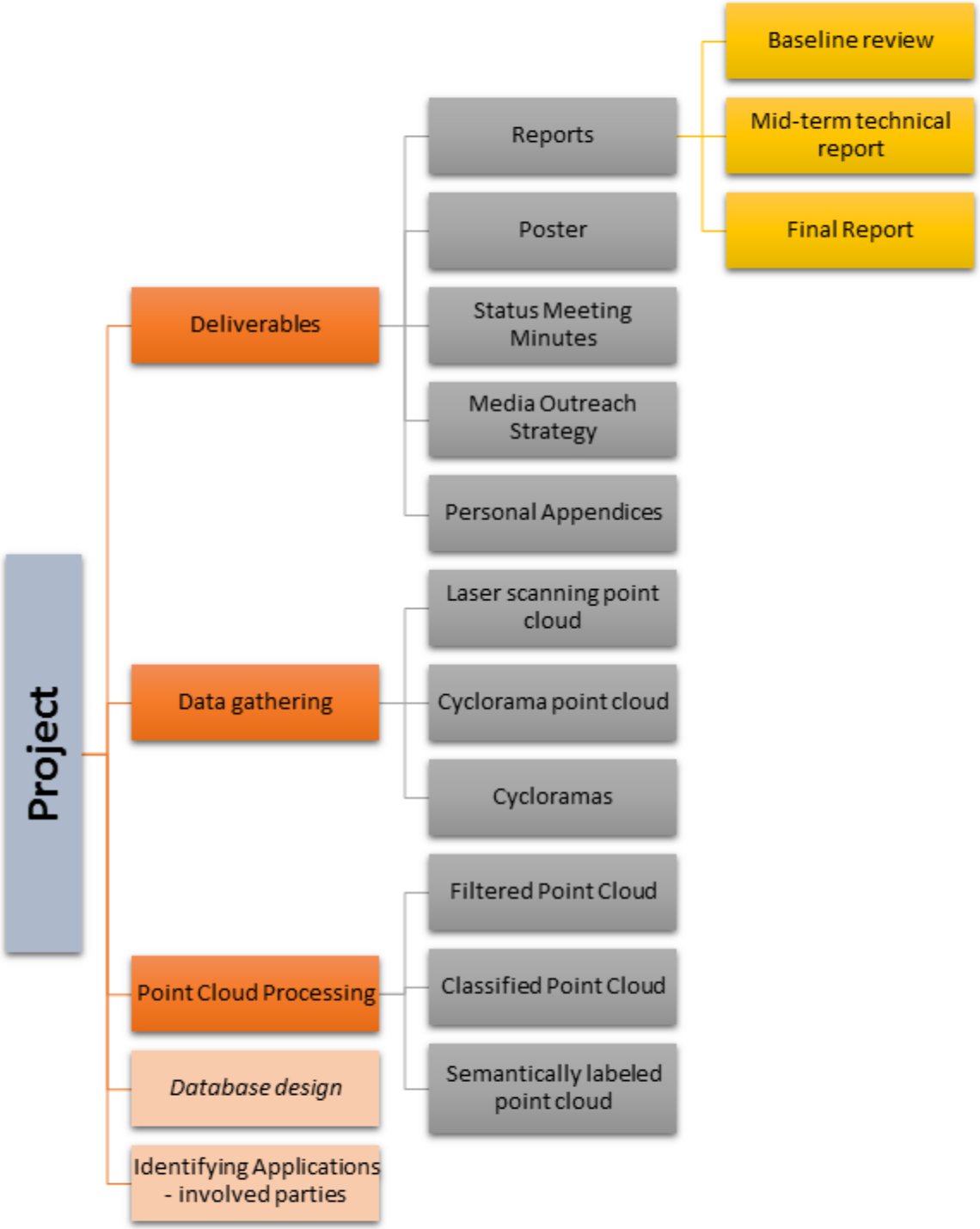


Figure 3, The Work Breakdown structure

According to the common understanding of the project by the team members, the project is divided into 4 levels (Figure 2). The first level represents the whole Project. The project is subdivided into the deliverables, the data gathering, the point cloud processing and, optionally, to the database design and application and parties identification. The deliverables represent the reports and other miscellaneous products.

3.3. Work Package Descriptions

A work package is highly related to the breakdown structure as defined in the previous section. The main goal of WPD is to define all steps that are important for the completion of a project. In the situation discussed in this report the packages presented, if combined with each other, form the complete project. Each work package is defined in Table 4 and are also followed with a specified deadline, as will be presented later in the GANNT chart.

Table 4, Work Package Descriptions

Work package	Title	Description	Is needed for WP
WP1	Data gathering	Arrange the data acquisition with Cyclomedia and obtain the point cloud. output: Coloured LIDAR point cloud, coloured point cloud from photogrammetry, images/cycloramas.	WP2 WP3 WP4
WP2	Filter the point cloud	Investigate ways to filter point clouds, split the ground points from the other points. output: A file with the ground points and a file with the other points.	WP3 WP4
WP3	Classify the point cloud	Determine ways to classify the ground points in different classes. The classes are formed with points that have similar properties, like: colour, normals and other geometric properties. output: classification of the ground points.	WP4
WP4	Semantically label point cloud	Put semantics to the classified ground points. These semantics are based on the function of the points, like: road, grass or sidewalk. Other labels are based on the composition of the points material, like: asphalt, vegetation or pavement.	(WP5) (WP7)

		output: ground points with types, vegetation, road, pavement and water.	
WP5	Database design (optional)	Design a database to store the semantically labelled point cloud and put it in the database. output: database with semantically labelled points.	(WP7)
WP6	Identifying involved parties/ applications (optional)	With the enriched point cloud, more applications and people can benefit. This task focuses on identifying possible users of the enriched data. output: list of possible users/applications of the enriched point cloud.	
WP7	Applications (optional)	To show the potential of the enriched point cloud an application listed in WP6 will be executed. output: application.	

3.4. Project Logic Diagram

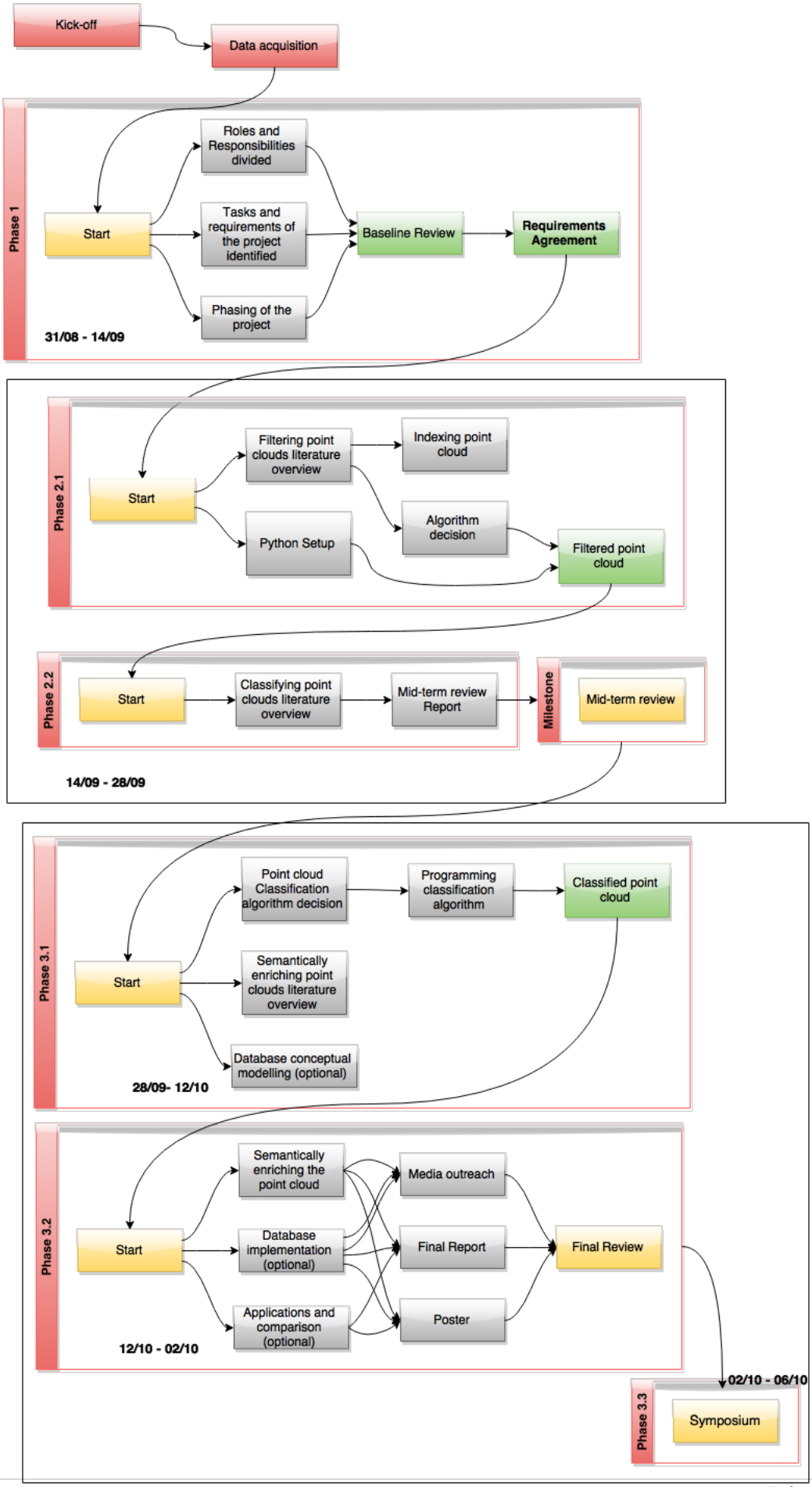
The project logic diagram is a schematic representation of the sequence of activities that will take place in timeline restricted phases in order to produce the final product (Figure 4). In order to provide a logic diagram the project is divided in phases. This timeline restricted phasing of the project takes place after the identification of the WBS and the WPD. Phasing the project means that the project is broken down into smaller pieces which will be easier to handle. According to the experience gained during the first year of Geomatics the team estimated the effort that needs to be put in those packages. This resulted in a general (time-restricted) phasing that is explained in Table 5.

Table 5, Project Phasing

Phase	**Date (Start - End)**	**Goal**	**Deadline(s)/ Milestones**
1: requirements analysis and planning	31/08 - 14/09	During this phase the scope and the problem is identified. The data gathering will take place in cooperation with Cyclomedia. In addition to that, a detailed list of the project's requirements is identified and the project planning takes place.	Baseline Review (14-09)
2: Conceptual analysis, information	14/09 - 28/09	During this phase, the different algorithms that are utilised will be identified and reviewed for their	Mid-Term Review (28-09)

gathering and combination		outputs. The algorithms aim, as a first step, at subdividing the point cloud in chunks or blocks for easier processing and data handling. Next, the filtering of the non-ground data will take place. Finally classification algorithms will be reviewed. The possibilities to use colours as a classification parameter will be mostly investigated. At the end of this phase, the point cloud will be filtered and possibly also classified into the needed categories.	
3: Detailed analysis	28/09 - 2/11	During this phase, the labelling of the point clouds will be researched and performed. This will then be followed by a case study on surface detection in an urban environment. Thereby, different classification and labelling methods will be tested. In this surface detection, the possibilities to use colours will be further investigated.	Final Review (2-11)
4: Final Review and Symposium		The final review will provide those who are interested with a complete and formal briefing on the whole project. The final review will include the process and results followed by details, conclusions and a statement of compliance with the requirements. The symposium will be the opportunity to present the completed project.	Final Review (2-11) Symposium (6-11) Report

Within those four defined phases, several activities are planned. The activities are linked together forming a path through the different project phases until the final product. These activities will be visualised within the Logic diagram in Figure 4.


Kick-off
Data acquisition
Phase 1
Start
Roles and Responsibilities divided
Tasks and requirements of the project identified
Phasing of the project
Baseline Review
Requirements Agreement
31/08 - 14/09
Phase 2.1
Start
Filtering point clouds literature overview
Indexing point cloud
Python Setup
Algorithm decision
Filtered point cloud
Phase 2.2
Start
Classifying point clouds literature overview
Mid-term review Report
Milestone
Mid-term review
14/09 - 28/09
Phase 3.1
Start
Point cloud Classification algorithm decision
Programming classification algorithm
Classified point cloud
Semantically enriching point clouds literature overview
Database conceptual modelling (optional)
28/09- 12/10
Phase 3.2
Start
Semantically enriching the point cloud
Database implementation (optional)
Applications and comparison (optional)
Media outreach
Final Report
Poster
Final Review
12/10 - 02/10
02/10 - 06/10
Phase 3.3
Symposium

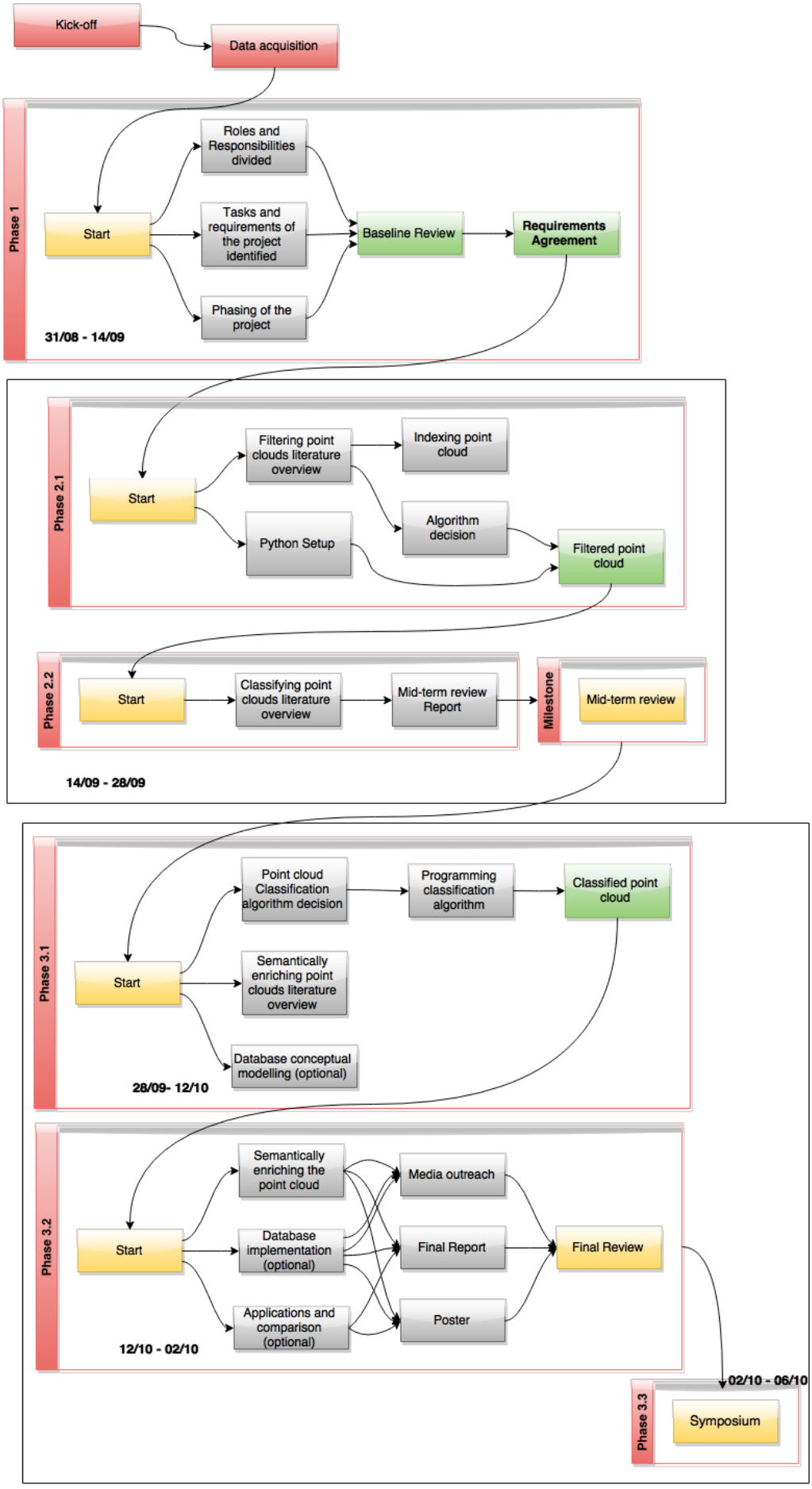

Kick-off
Data acquisition
Phase 1
Start
Roles and Responsibilities divided
Tasks and requirements of the project identified
Phasing of the project
Baseline Review
Requirements Agreement
31/08 - 14/09
Phase 2.1
Start
Filtering point clouds literature overview
Indexing point cloud
Python Setup
Algorithm decision
Filtered point cloud
Phase 2.2
Start
Classifying point clouds literature overview
Mid-term review Report
Milestone
Mid-term review
14/09 - 28/09
Phase 3.1
Start
Point cloud Classification algorithm decision
Programming classification algorithm
Classified point cloud
Semantically enriching point clouds literature overview
Database conceptual modelling (optional)
28/09- 12/10
Phase 3.2
Start
Semantically enriching the point cloud
Database implementation (optional)
Applications and comparison (optional)
Media outreach
Final Report
Poster
Final Review
12/10 - 02/10
02/10 - 06/10
Phase 3.3
Symposium

Figure 4, Logic diagram

3.5. GANTT Chart

The GANTT Chart (Figure 5). In a GANTT Chart the packages defined previously are displayed against the time factor. The deliverables are also displayed, in bold.

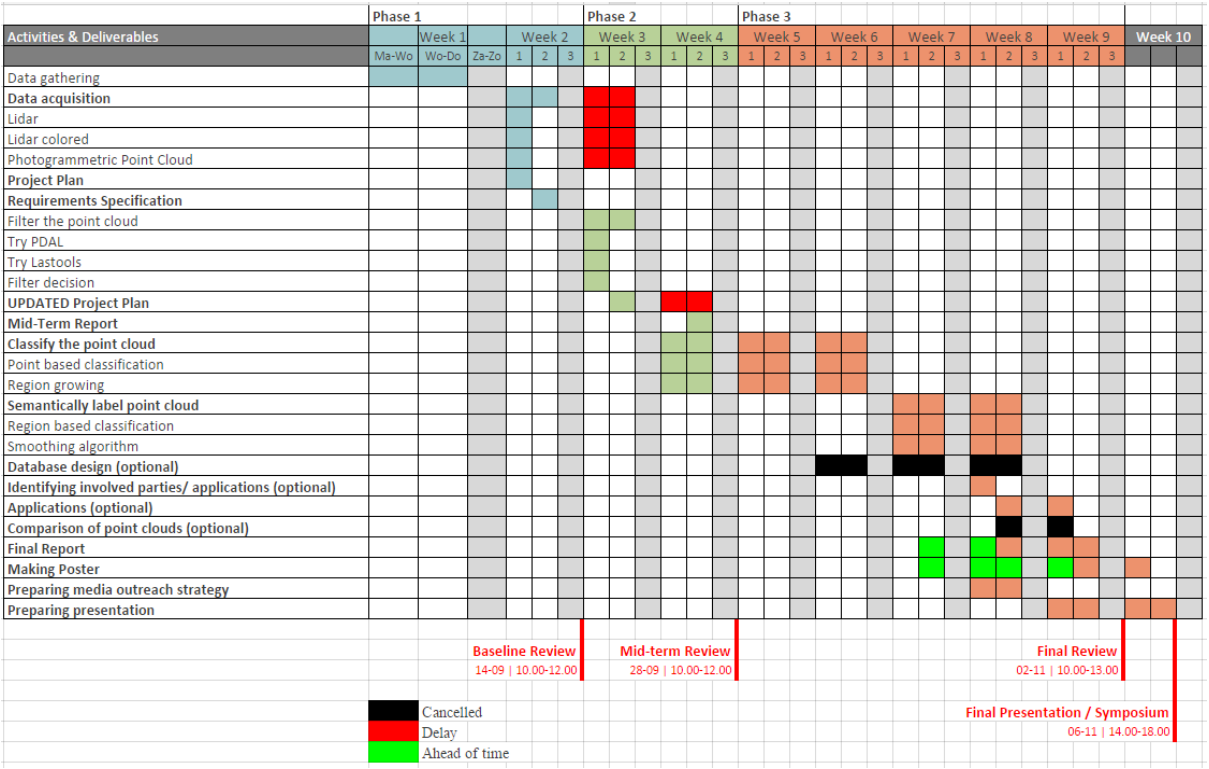


Figure 5, The updated GANTT chart

Appendix B - Python code

B.1. Supervised classification with Smoothing

```
import numpy as np, scipy.spatial as sp, matplotlib.pyplot as plt, timeit,
math
from scipy import linalg
from sets import Set
from color_analysis import * #Import our color_functions
from neighbourhood_characteristics import * #Import function to calculate
characteristic values of a point cloud
from getStatistics import * #import sample statistics

class Point(object):
  __slots__ = ('index', 'xyz', 'lab', 'curv', 'normal', 'nn_indices',
"region_number", 'rgb', 'c')
  def __init__(self, index, xyz, lab, curv, normal, nn_indices, rgb):
    self.index = index
    self.xyz = xyz
    self.lab = lab
    self.curv = curv
    self.normal = normal
    self.nn_indices = nn_indices
    self.region_number = -1
    self.rgb = rgb

class Region(object):
  __slots__ = ('number', 'points', 'lab_avg')
  def __init__(self, number, points, lab_avg):
    self.number = number
    self.points = points
    self.lab_avg = lab_avg

def unpack_xyz(infile):
    points_xyz = []
    count = 0
    for each in infile:
        values = each.split(' ')
        xyz = (float(values[0]), float(values[1]), float(values[2]))
        rgb = (float(values[4]), float(values[5]), float(values[6])) # ----
-depends on RGB position-----
        c,i,e = CIEXYZ(np.array([rgb[0]/255.0, rgb[1]/255.0,
rgb[2]/255.0]).T)
        l,a,b = CIELab(c,i,e)
        dic[xyz] = ((l,a,b),rgb)
        points_xyz.append(xyz)
        count += 1
    print "Total points in this scene: ", count
    return np.array(points_xyz)

def region_growing(l):
    c_threshold = 0.001 #Curvature threshold
    a_threshold = 45*math.pi/180 #Angle threshold
    available_points = sorted(l, key=lambda x: x.curv) #Copy of point list
(because list is mutable)
    regions = []
    region_index = 0
```

```
    for point in available_points:
        if point.region_number == -1:
            point.region_number = region_index
            current_region = Set([point])
            suma = point.lab[1]
            sumb = point.lab[2]
            sumL = point.lab[0]
            current_seeds = Set([point])
            while len(current_seeds) != 0:
                seed = current_seeds.pop()
                seeds_nn_indices = seed.nn_indices
                for index in seeds_nn_indices:
                    n = all_points[index]
                    if n.region_number == -1:
                        #Calculate angle between normals:
                        p = (seed.normal[0]*n.normal[0]) +
(seed.normal[1]*n.normal[1]) + (seed.normal[2]*n.normal[2]) #Dot product of
normal vectors
                        q = math.sqrt((seed.normal[0])**2 +
(seed.normal[1])**2 + (seed.normal[2])**2) * math.sqrt((n.normal[0])**2 +
(n.normal[1])**2 + (n.normal[2])**2) #Length normal product
                        if p/q > 1.0: #Make sure values are not bigger than
1 (otherwise cos function doesn't work)
                            p = 1.0
                            q = 1.0
                        if q < 0.00000001:
                            q = 0.0001
                        angle = math.acos(p/q)
                        #dp = abs((seed.normal[0]*n.normal[0]) +
(seed.normal[1]*n.normal[1]) + (seed.normal[2]*n.normal[2])) #Dot product
of normal vectors
                        if abs(seed.lab[1] - n.lab[1]) < 0.2 and angle <
a_threshold:
                            n.region_number = region_index
                            suma = suma + n.lab[1]
                            sumb = sumb + n.lab[2]
                            sumL = sumL + n.lab[0]
                            current_region.add(n)
                            if abs(n.curv - seed.curv) < c_threshold:
#abs(seed.curv - n.curv) < c_threshold:
                                current_seeds.add(n)
            l_region = len(current_region)
            regions.append(Region(region_index, current_region,
(sumL/l_region, suma/l_region, sumb/l_region)))
            region_index += 1
    return regions

def smoothing(l):
    g_list = []
    r_list = []
    c_list = []
    road_list = []
    t_list = []
    for point in l:
        g = 0
        r = 0
        c = 0
        road = 0
        t = 0
        for index in point.nn_indices:
            n = all_points[index]
```

```
            if n.c == 'road':
                road += 1
            elif n.c == 'c':
                c += 1
            elif n.c == 't':
                t += 1
            elif n.c == 'g':
                g += 1
            elif n.c == 'r':
                r += 1
        maximum = max((g,'g'),(r,'r'),(c,'c'),(road,'road'),(t,'t'))
        if maximum[1] == 'road':
            road_list.append(point)
        elif maximum[1] == 'c':
            c_list.append(point)
        elif maximum[1] == 't':
            t_list.append(point)
        elif maximum[1] == 'g':
            g_list.append(point)
        elif maximum[1] == 'r':
            r_list.append(point)
    return g_list, r_list, c_list, road_list, t_list

if __name__ == "__main__":
    timer1 = timeit.default_timer() #Start of processing
    dic = {} #Declare global dictionary. Used to lookup RGB and LAB values
after K-D tree indexing
    File = open('StereoGroundclassification.xyz')
    points = unpack_xyz(File)
    File.close()
    kdtree = sp.cKDTree(points) #Build K-D tree
    n = 20 #Define size of local neighbourhood
    nn_query = kdtree.query(kdtree.data, k=n, eps=0, p=2) #Find nearest
neighbours, eps:exact search, p:euclidean distance
    nn_indices = nn_query[1] #Neighbour indices in the K-D tree
    nn = kdtree.data[nn_indices] #Retrieve neighbour values
    nn_T = nn.transpose(0,2,1)
    centroids = np.average(nn_T, axis=2).reshape(len(nn_T),3,1)
    vectors = nn_T - centroids #Diff vectors (between neighbour points and
centroid)
    VP = np.array([594088, 5762450, 80]) #Viewpoint


    all_points = []
    for i in range(len(kdtree.data)):
        cov = np.cov(vectors[i]) #Covariance matrix 3x3
        eig = linalg.eig(cov) #Eigenvalues and eigenvectors
        #The plane normal n is the eigenvector of S with the smallest
eigenvalue >>>> min(eig[0])
        normal = np.array(eig[1][np.where(eig[0] == min(eig[0]))[0][0]])
        if np.dot(normal, VP - kdtree.data[i]) < 0: #Check consistent
global orientation
            normal = (-1)* normal #Flip normal
        l = sorted(eig[0].real, reverse=True) #Sorted eigenvalues
(ascending)
        C = characteristics(l) #Eigenvalue-based 3D features
        #curvature_list.append(C)
        xyz = kdtree.data[i]
        all_points.append(Point(i, xyz, dic[tuple(xyz)][0], C, normal,
nn_indices[i],dic[tuple(xyz)][1]))
```

```
    regions = region_growing(all_points)


    cwd = os.getcwd()
    path = cwd + "\Samples"
    s = gatherStatistics(statistics(unpack(path, os.listdir(path))))

    for each in regions:
        avea = each.lab_avg[1]
        aveb = each.lab_avg[2]
        aveL = each.lab_avg[0]


        ## Samples for road
        if ((s["Road"][0][0]<aveL<s["Road"][0][1] and
s["Road"][0][2]<avea<s["Road"][0][3] and
s["Road"][0][4]<aveb<s["Road"][0][5])
                or (s["Road"][1][0]<aveL<s["Road"][1][1] and
s["Road"][1][2]<avea<s["Road"][1][3] and
s["Road"][1][4]<aveb<s["Road"][1][5])
                or (s["Road"][2][0]<aveL<s["Road"][2][1] and
s["Road"][2][2]<avea<s["Road"][2][3] and
s["Road"][2][4]<aveb<s["Road"][2][5])):
            for point in each.points:
                point.c = 'road'

        ##Samples for cyclepath
        elif ((s["Cyclepath"][0][0]<aveL<s["Cyclepath"][0][1] and
s["Cyclepath"][0][2]<avea<s["Cyclepath"][0][3] and
s["Cyclepath"][0][4]<aveb<s["Cyclepath"][0][5])
                or (s["Cyclepath"][1][0]<aveL<s["Cyclepath"][1][1] and
s["Cyclepath"][1][2]<avea<s["Cyclepath"][1][3] and
s["Cyclepath"][1][4]<aveb<s["Cyclepath"][1][5])
                or (s["Cyclepath"][2][0]<aveL<s["Cyclepath"][2][1] and
s["Cyclepath"][2][2]<avea<s["Cyclepath"][2][3] and
s["Cyclepath"][2][4]<aveb<s["Cyclepath"][2][5])
                or (s["Cyclepath"][3][0]<aveL<s["Cyclepath"][3][1] and
s["Cyclepath"][3][2]<avea<s["Cyclepath"][3][3] and
s["Cyclepath"][3][4]<aveb<s["Cyclepath"][3][5])
                or (s["Cyclepath"][4][0]<aveL<s["Cyclepath"][4][1] and
s["Cyclepath"][4][2]<avea<s["Cyclepath"][4][3] and
s["Cyclepath"][4][4]<aveb<s["Cyclepath"][4][5])
                or (s["Cyclepath"][5][0]<aveL<s["Cyclepath"][5][1] and
s["Cyclepath"][5][2]<avea<s["Cyclepath"][5][3] and
s["Cyclepath"][5][4]<aveb<s["Cyclepath"][5][5])
                or (s["Cyclepath"][6][0]<aveL<s["Cyclepath"][6][1] and
s["Cyclepath"][6][2]<avea<s["Cyclepath"][6][3] and
s["Cyclepath"][6][4]<aveb<s["Cyclepath"][6][5])
                or (s["Cyclepath"][7][0]<aveL<s["Cyclepath"][7][1] and
s["Cyclepath"][7][2]<avea<s["Cyclepath"][7][3] and
s["Cyclepath"][7][4]<aveb<s["Cyclepath"][7][5])):
            for point in each.points:
                point.c = 'c'

        ##Samples for tiles
        elif ((s["Tiles"][0][0]<aveL<s["Tiles"][0][1] and
s["Tiles"][0][2]<avea<s["Tiles"][0][3] and
s["Tiles"][0][4]<aveb<s["Tiles"][0][5])
                or (s["Tiles"][1][0]<aveL<s["Tiles"][1][1] and
s["Tiles"][1][2]<avea<s["Tiles"][1][3] and
s["Tiles"][1][4]<aveb<s["Tiles"][1][5])
```

```
                or (s["Tiles"][2][0]<aveL<s["Tiles"][2][1] and
s["Tiles"][2][2]<avea<s["Tiles"][2][3] and
s["Tiles"][2][4]<aveb<s["Tiles"][2][5])
                or (s["Tiles"][3][0]<aveL<s["Tiles"][3][1] and
s["Tiles"][3][2]<avea<s["Tiles"][3][3] and
s["Tiles"][3][4]<aveb<s["Tiles"][3][5])
                or (s["Tiles"][4][0]<aveL<s["Tiles"][4][1] and
s["Tiles"][4][2]<avea<s["Tiles"][4][3] and
s["Tiles"][4][4]<aveb<s["Tiles"][4][5])
                or (s["Tiles"][5][0]<aveL<s["Tiles"][5][1] and
s["Tiles"][5][2]<avea<s["Tiles"][5][3] and
s["Tiles"][5][4]<aveb<s["Tiles"][5][5])
                or (s["Tiles"][6][0]<aveL<s["Tiles"][6][1] and
s["Tiles"][6][2]<avea<s["Tiles"][6][3] and
s["Tiles"][6][4]<aveb<s["Tiles"][6][5])):
            for point in each.points:
                point.c = 't'

        # Green
        elif avea < -2:
            for point in each.points:
                point.c = 'g'

        ##The unknown class
        else:
            for point in each.points:
                point.c = 'r'


    print len(all_points)
    g, r, c, road, t = smoothing(all_points)
    print len(g), len(r), len(c), len(road), len(t)
    fg = open("grassa3.xyz" , 'w')
    fr = open("resta3.xyz", 'w')
    fc = open("cyclea3.xyz", 'w')
    froad = open("roada3.xyz", 'w')
    ft = open("tilesa3.xyz", 'w')

    fg1 = open("grass.xyz" , 'w')
    fc1 = open("cycle.xyz", 'w')
    froad1 = open("road.xyz", 'w')
    ft1 = open("tiles.xyz", 'w')

    for point in g:
        fg.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],point.rgb[0],point.rgb[1],point.rgb
[2])))
        fg1.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],30,193,30)))
    for point in r:
        fr.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],point.rgb[0],point.rgb[1],point.rgb
[2])))
    for point in c:
        fc.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],point.rgb[0],point.rgb[1],point.rgb
[2])))
        fc1.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],226,32,32)))
    for point in road:
```

```
        froad.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],point.rgb[0],point.rgb[1],point.rgb
[2])))
        froad1.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],28,27,27)))
    for point in t:
        ft.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],point.rgb[0],point.rgb[1],point.rgb
[2])))
        ft1.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],60,60,225)))

    timer4 = timeit.default_timer()
    fg.close()
    fr.close()
    fc.close()
    froad.close()
    ft.close()
    print 'Done! (in', (timer4 - timer1)/60.0, 'minutes)'
```

B.2. Get training samples statistics

```
import os
import numpy as np
from color_analysis import *

def unpack(path,filename):
    lengte = len(filename)
    L_end = []
    B_end = []
    A_end =[]
    l1 = tuple()
    a1 = tuple()
    b1 = tuple()
    files = []
    for each in range(lengte):
        if filename[each].endswith(".xyz"):
            files.append(filename[each])
            Llist =[]
            Alist =[]
            Blist =[]
            minl = []
            points = open(path+ "//" + filename[each])

            for i in points:
                splitted = i.split(' ')
                R = float(splitted[3])
                G = float(splitted[4])
                B = float(splitted[5])
                [X,Y,Z] =  CIEXYZ(np.array([R/255.0,G/255.0,B/255.0]).T)
                (L,a,b) = CIELab(X,Y,Z)
                Llist.append(L)
                Alist.append(a)
                Blist.append(b)
            L_end.append(Llist)
            B_end.append(Blist)
            A_end.append(Alist)
    return L_end,A_end,B_end, files
```

```
def statistics(lst):
    """Gets the samples and returns a list of lists with the name of the
file and for each component of the Lab
    color space the domain.

    The list is structured as follows:

    ["name.xyz",L_min,L_max,a_min,a_max,b_min,b_max]
    """
    sublist = []
    for i in range(len(lst[3])):
        sublist.append([lst[3][i],np.mean(lst[0][i])-
3*np.std(lst[0][i]),np.mean(lst[0][i])+3*np.std(lst[0][i]),np.mean(lst[1][i
])-
3*np.std(lst[1][i]),np.mean(lst[1][i])+3*np.std(lst[1][i]),np.mean(lst[2][i
])-3*np.std(lst[2][i]),np.mean(lst[2][i])+3*np.std(lst[2][i])])
    return sublist

def gatherStatistics(l):
    dic = {}
    for i in l:
        name = i[0].split('_')
        if name[0] not in dic:
            dic[name[0]] = [i[1:]]
        else:
            dic[name[0]].append(i[1:])
    return dic

if __name__=="__main__":
    #Path with the samples
    cwd = os.getcwd()
    path = cwd + "\\Samples\\"
    s = statistics(unpack(path, os.listdir(path)))
    print gatherStatistics(s)
```

B.3. CIE-Lab transformations

```
import numpy as np

def f(t):
    if t > 0.008856:
        return t**(1/3.0)
    else:
        return 7.787*t + (16.0/116.0)

def CIEXYZ(c):
    a = [[0.4125, 0.3576, 0.1804],[0.2127, 0.7152,
0.0722],[0.0193,0.300,0.9502]]
    return np.dot(a,c)

def CIELab(X,Y,Z):
    y = f(Y/Y0)
    L = (116*y)-16
    a = 500*(f(X/X0)-y)
    b = 200*(y-f(Z/Z0))
    return L,a,b

white = np.array([1.0,1.0,1.0]).T
X0,Y0,Z0 = CIEXYZ(white)
```

B.4. Geometrical Characteristics

```
def characteristics(l):
    """
    Calculate characteristic values of a point cloud.
    Input: the eigenvectors in ascending order
    Output: The charateristics
    """
    S = l[0] + l[1] + l[2] #Sum of eigenvalues
    e = [l[0]/S,l[1]/S,l[2]/S] #normalised eigenvalues

    L = (e[0]-e[1])/e[0] #Linearity
    P = (e[1]-e[2])/e[0] #Planarity
    S = e[2]/e[0] #Scattering
    O = (e[0]*e[1]*e[2])**(1.0/3.0) #Omnivariance
    A = (e[0]-e[2])/e[0] #Anisotropy
    E = -(l[0]*log(l[0])+l[1]*log(l[1])+l[2]*log(l[2])) #Eigenentropy
    C = l[2]/S #Change of curvature
    return C
```

B.5. Unsupervised Classification with Merging algorithm

```
import numpy as np, scipy.spatial as sp, matplotlib.pyplot as plt, timeit,
math
from scipy import linalg
from sets import Set
from color_analysis import * #Import our color_functions
from neighbourhood_characteristics import * #Import function to calculate
characteristic values of a point cloud

class Point(object):
  __slots__ = ('index', 'xyz', 'lab', 'curv', 'normal', 'nn_indices',
"region_number", 'rgb', 'c')
  def __init__(self, index, xyz, lab, curv, normal, nn_indices, rgb):
    self.index = index
    self.xyz = xyz
    self.lab = lab
    self.curv = curv
    self.normal = normal
    self.nn_indices = nn_indices
    self.region_number = -1
    self.rgb = rgb


class Region(object):
  __slots__ = ('number', 'points', 'l', 'lab_avg')
  def __init__(self, number, points, l, lab_avg):
    self.number = number
    self.points = points
    self.l = l
    self.lab_avg = lab_avg

def unpack_xyz(infile):
    points_xyz = []
    count = 0
    for each in infile:
        values = each.split(' ')
        xyz = (float(values[0]), float(values[1]), float(values[2]))
        rgb = (float(values[3]), float(values[4]), float(values[5])) # ----
-depends on RGB position-----
```

```
        c,i,e = CIEXYZ(np.array([rgb[0]/255.0, rgb[1]/255.0,
rgb[2]/255.0]).T)
        l,a,b = CIELab(c,i,e)
        dic[xyz] = ((l,a,b),rgb)
        points_xyz.append(xyz)
        count += 1
    print "Total points in this scene: ", count
    return np.array(points_xyz)

def region_growing(l):
    c_threshold = 0.0015 #Curvature threshold
    a_threshold = 45*math.pi/180 #Angle threshold
    available_points = sorted(l, key=lambda x: x.curv) #Copy of point list
(because list is mutable)
    regions = []
    region_index = 0
    for point in available_points:
        if point.region_number == -1:
            point.region_number = region_index
            current_region = [point]
            lab_a = point.lab[1]
            lab_b = point.lab[2]
            current_seeds = Set([point])
            while len(current_seeds) != 0:
                seed = current_seeds.pop()
                seeds_nn_indices = seed.nn_indices
                for index in seeds_nn_indices:
                    n = all_points[index]
                    if n.region_number == -1:
                        #Calculate angle between normals:
                        p = (seed.normal[0]*n.normal[0]) +
(seed.normal[1]*n.normal[1]) + (seed.normal[2]*n.normal[2]) #Dot product of
normal vectors
                        q = math.sqrt((seed.normal[0])**2 +
(seed.normal[1])**2 + (seed.normal[2])**2) * math.sqrt((n.normal[0])**2 +
(n.normal[1])**2 + (n.normal[2])**2) #Length normal product
                        if p/q > 1.0: #Make sure values are not bigger than
1 (otherwise cos function doesn't work)
                            p = 1
                            q = 1
                        angle = math.acos(p/q)
                        #dp = abs((seed.normal[0]*n.normal[0]) +
(seed.normal[1]*n.normal[1]) + (seed.normal[2]*n.normal[2])) #Dot product
of normal vectors
                        if abs(seed.lab[1] - n.lab[1]) < 0.2 and angle <
a_threshold:
                            n.region_number = region_index
                            current_region.append(n)
                            lab_a += n.lab[1]
                            lab_b += n.lab[2]
                            if abs(seed.curv - n.curv) < c_threshold:
#n.curv < c_threshold:
                                current_seeds.add(n)
            l_region = len(current_region)
            regions.append(Region(region_index, current_region, l_region,
(lab_a/l_region, lab_b/l_region)))
            region_index += 1
    return regions

def dist(x1,y1,x2,y2):
    return (x1-x2)**2+(y1-y2)**2
```

```
if __name__ == "__main__":
    timer1 = timeit.default_timer() #Start of processing
    dic = {} #Declare global dictionary. Used to lookup RGB values after K-
D tree indexing
    File = open('CIELab.xyz')
    points = unpack_xyz(File)
    File.close()
    kdtree = sp.cKDTree(points) #Build K-D tree
    n = 20 #Define size of local neighbourhood
    nn_query = kdtree.query(kdtree.data, k=n, eps=0, p=2) #Find nearest
neighbours, eps:exact search, p:euclidean distance
    nn_indices = nn_query[1] #Neighbour indices in the K-D tree
    nn = kdtree.data[nn_indices] #Retrieve neighbour values
    nn_T = nn.transpose(0,2,1)
    centroids = np.average(nn_T, axis=2).reshape(len(nn_T),3,1)
    vectors = nn_T - centroids #Diff vectors (between neighbour points and
centroid)
    VP = np.array([-60906, 22449, 4300]) #Viewpoint


    all_points = []
    for i in range(len(kdtree.data)):
        cov = np.cov(vectors[i]) #Covariance matrix 3x3
        eig = linalg.eig(cov) #Eigenvalues and eigenvectors
        #The plane normal n is the eigenvector of S with the smallest
eigenvalue >>>> min(eig[0])
        normal = np.array(eig[1][np.where(eig[0] == min(eig[0]))[0][0]])
        if np.dot(normal, VP - kdtree.data[i]) < 0: #Check consistent
global orientation
            normal = (-1)* normal #Flip normal
        l = sorted(eig[0].real, reverse=True) #Sorted eigenvalues
(ascending)
        C = characteristics(l) #Eigenvalue-based 3D features
        #curvature_list.append(C)
        xyz = kdtree.data[i]
        all_points.append(Point(i, xyz, dic[tuple(xyz)][0], C, normal,
nn_indices[i], dic[tuple(xyz)][1]))
    regions = region_growing(all_points)
    THRES = 5000
    regionsss = []
    numbersss = Set() #All regionnumbers smaller than threshold
    for region in regions:
        if region.l < THRES:
            regionsss.append(region)
            numbersss.add(region.number)
    deleted_numbersss = []

    while len(regionsss) != 0:
        x = Set() #Set with all the regionnumbers of the neigbours of all
points that is not own region number
        region = regionsss.pop()
        numbersss.remove(region.number)
        deleted_numbersss.append(region.number)
        for point in region.points:
            for i in point.nn_indices:
                rn = all_points[i].region_number
                if rn != region.number:
                    x.add(rn)
        xd = x.difference(deleted_numbersss)
        if len(xd) != 0:
```

```
            distances = []
            for each in xd:
                dis =
dist(region.lab_avg[0],region.lab_avg[1],regions[each].lab_avg[0],regions[e
ach].lab_avg[1])
                distances.append((dis,each))
            rnmax = min(distances)[1]
            regions[rnmax].points.extend(region.points)
            regions[rnmax].l += region.l
            if rnmax in numbersss and regions[rnmax].l >= THRES: #If region
that is merged to is > threshold remove from this list
                regionsss.remove(regions[rnmax])
                numbersss.remove(regions[rnmax].number)
                deleted_numbersss.append(regions[rnmax].number) ###!!!!
        regions[region.number].points = None

    for region in regions:
        if region.points != None:
            f = open("output_" + str(region.number) + ".xyz", 'w')
            for point in region.points:
                f.write(('%s %s %s %s %s %s\n' %
(point.xyz[0],point.xyz[1],point.xyz[2],point.rgb[0],point.rgb[1],point.rgb
[2])))
            f.close()

    timer2 = timeit.default_timer()

    print 'Done! (in', (timer2 - timer1)/60.0, 'minutes)'
```

Appendix C - Media Outreach

C.1. Strategy

An outreach to the media is in benefit for this Geomatics Synthesis Project, it's purpose is to inform society on the context of the project and the outcome in order to improve the visibility of the TU Delft's Geomatics program. This Geomatics Synthesis Project with it's outcome should be most interesting for fellow Geomatics students and other students or organisations in the geomatics field with interest in point clouds. To inform this select few different kinds of media can be used. At first a presentation in which the whole project flow and conclusions are explained and visualised with the opportunity to ask questions to the researchers (The Geomatics Synthesis Symposium) is useful to reach the main target group related to the University. Secondly publishing an article or a paper within a in the geomatics field notifiable magazine like GIM International Young Geo in Focus or GIS magazine will reach a larger group interested in Geo-information. Publishing in such magazines concern certain guidelines which have to be met. Publication in GIM International is for sure. GIS magazine is approached for possible publication.

With this media outreach the idea is to communicate a certain message. This message, related to the results of the Geomatics Synthesis Project, is defined as to inform a wide variety of people that it is possible to semantically enrich large point clouds very accurate with the use of colour. Beside this main message the actual goal is to recall benefits of working with point clouds when they are semantically enriched.

One of our major resources are the other groups, we have the same interest in reaching out with the results. Therefore, we decided to cooperate with the other groups in creating an outreach for the symposium, and together look for possible external speakers for the symposium.