CLASSIFICATION OF LARGE SCALE OUTDOOR POINT CLOUDS USING CONVOLUTIONAL NEURAL NETWORKS

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Tom Hemmes

April 2018

The work in this thesis was made in the:

Geomatics
Faculty of Architecture & the Built Environment
Delft University of Technology

Intelligent Imaging
TNO Research Institute

Supervisors:	Dr.ir. Mathias Lemmens
	Prof.dr.ir. Peter van Oosterom
TNO Supervisor:	Dr.ir. Maarten Kruithof
Co-reader:	Kaixuan Zhou, MSc.

ABSTRACT

There is a paradigm shift from two- to three-dimensional data, from maps to information dense models. Self-driving cars, digitization of historic buildings or maintenance of highway infrastructure are a small selection of many applications that use laser scanning to acquire three-dimensional data of our physical surroundings. Most of these applications require more than the shape of their surroundings. For example, a self-driving car needs to identify pedestrians, road signs and traffic lights in order to navigate safely. Therefore the point cloud acquired by laser scanning needs to be enriched with additional information. Automatic assignment of the object type a point belongs to is called *classification*.

This research focuses on *deep learning* for point cloud classification, because it revolutionized classification of imagery. *PointNet* is used to enable deep learning directly on point cloud data sets. To date PointNet is proven for indoor point clouds, this research explores the application of PointNet on an outdoor highway scene. The methodology creates point cloud training data efficiently by reusing known 2D object locations. Different spatial representations and sampling methods for the points are tested.

On average the method classifies 50% of the points correctly in four object classes. In combination with clustering of the point-wise predictions, the method predicts 60% of the 2D object locations successfully. The performance is comparable to the 47% average class accuracy PointNet achieves for 13 classes on the indoor data set.

ACKNOWLEDGEMENTS

This thesis is the result of a year-long graduation project. Graduation is an individual project, however I was supported by supervisors, colleagues and friends throughout this project.

First of all I would like to thank Mathias Lemmens (Tjeu), my first supervisor. Tjeu taught me a great deal of geospatial research throughout my master Geomatics. Tjeu is a long distance cyclist, whereas I personally prefer riding fast. He takes the time to interpret the surroundings and uses all available information for the task at hand. Our debates on the balance between scene knowledge and universal applicability of the method will inspire me for times to come. Then I would like to thank Peter van Oosterom for being my second supervisor and Kaixuan Zhou for proof-reading my thesis.

The graduation project is an opportunity to pursue academic research in a new environment. For me this environment was the department *Intelligent Imaging* at *TNO*. Maarten Kruithof leads the *Deep Learning* research group at Intelligent Imaging and I was lucky to have Maarten as supervisor. Especially his expertise in deep learning was essential for this project. I would like to thank Maarten for all his guidance and patience during this research.

TNO has a culture of supporting graduation research. Sometimes, in my skepticism of slow legacy corporations, I refer to this as a graduation factory. Unfairly, because I learned just as much from the other students at the department as from my supervisors. Specifically, I would like to thank Arthur van Rooijen, Arjan van Ramshorst and Ward Heij for their support and feedback at almost any hour of the day.

Finally, I would like to thank Wikke for her support and enabling me to put everything into perspective. After all, this is only the beginning.

CONTENTS

GLOSSARY

accuracy From the domain of ML. The percentage of correctly classified samples during training or validation.. 1

augmentation Manipulation of the training data in order to create more data, e.g. rotation of images in image classification.. 1

epoch From the domain of ML. One iteration of the entire training and validation data set.. 1, 19

feature A measurable property or characteristic of a phenomenon being observed. *Note the definition used is from the ML domain. In the geo-spatial domain a feature refers to an entire physical object*.. 1, 6

generalization From the domain of ML. The capability of a model to perform on different data.. 1, 36

ground truth Information from direct observation, as opposed to information from inference.. 1, 17, 21

loss From the domain of ML. A summation of the errors made during training or validation.. 1

object A real-world physical three-dimensional object, e.g. a road sign or traffic light.. 1

real-world The existing state of things, as opposed to a simulation or imagination. 1, 5, 20, 33, 35

sample A subset of the data set. For this research; a selection of points within one spatial partition.. 1, 18

voxel A portmanteau of volume and pixel, indicates a volume cell in three-dimensional space. 1, 8, 19, 22, 33

ACRONYMS

2D two-dimensional. 1–3, 7–9, 11, 16, 31, 32
3D three-dimensional. 1, 2, 5, 8

BIM Building Information Model. 1, 2, 5

CAD Computer Aided Drawing. 1, 2, 11, 16
CNN Convolutional Neural Network. 1, 2, 7, 8
CPU Central Processing Unit. 1

DEM Digital Elevation Model. 1, 5
DL Deep Learning. 1, 2, 7–9, 11, 16, 21, 33, 35, 37, 38

GCNN Graph Convolutional Neural Network. 1, 9, 36, 38
GIS Geospatial Information System. 1
GNSS Global Navigation Satellite System. 1, 6, 11
GPS Global Positioning System. 1
GPU Graphics Processing Unit. 1, 16, 19, 20

IMU Inertial Measurement Unit. 1, 6, 11
IOU Intersection over Union. 1, 27

MIOU Mean Intersection over Union. 1, 9, 23, 27–29, 38
ML Machine Learning. vii, 1, 6, 7
MLS Mobile Laser Scanning. 1, 3, 5, 6, 11, 33, 35, 37, 39

NN Neural Network. 1, 7

PCA Principal Component Analysis. 1

SFC Space-filling curve. 1

TLS Terrestrial Laser Scanning. 1, 5, 39

1 INTRODUCTION

1.1 TOPIC

From self-driving cars that navigate on busy crossroads, to archaeologists who archive historical sites in high detail, or an infrastructure planner who maps a highway scene before maintenance. Many applications rely on three-dimensional (3D) data of our physical surroundings. A popular method to acquire this data is laser scanning, that acquire the data in the form of *point clouds*.

Consider the example of the infrastructure planner. It is important that every road sign, traffic light or lamppost removed during maintenance of the highway returns to it's original location. The infrastructure planner can refer to a 3D scan of the area made before the maintenance. He identifies the different objects in the scan and creates a two-dimensional (2D) map with the object's type and location. After maintenance the map is used to return objects to their original locations.

In general laser scanning point clouds represent the outer surface of objects in the scene. Points in the point cloud lack information on the object they belong to. The process to add this information to each point is called *classification*. Classification of point clouds can be automated using a computer algorithm that determines the object type based on *features*. Properties of a point, statistics of a point's neighbourhood or scene knowledge are all features. For example, the height or Z value of a point can be used to discriminate points that are part of the ground surface. Or the normal vector on neighbouring points of a point, which is similar for points in the same plane.

The classification algorithm consists of a model and point cloud processing functionality. The model determines the class label for a point. This model can be rule-based or learned from example data. Traditional models are often rule-based, they are based on manually engineered rules that discriminate points based on manually engineered features [Friedl and Brodley, 1997]. The first learning algorithms were also based on manually engineered features. Both manually engineered rules and features involve domain and scene knowledge and the performance of the resulting models is application dependent. More recently, Deep Learning (DL) models are used for classification problems. Especially in image classification these models achieve state-of-the-art accuracy. Instead of manually engineering features DL models learn *abstract features* directly from examples, called training data.

The success of DL on imagery is owed to Convolutional Neural Network (CNN). Images are often stored as two-dimensional matrices of pixel values. CNN networks use convolutions to incorporate spatial information of these two-dimensional matrices. The resulting abstract features of a pixel contain information of neighbouring pixels. To apply the same principle to point clouds is non-trivial. Point clouds are generally stored as an array of point records. However, these point records are in no particular order. This poses a challenge to incorporate local spatial information into the abstract features and create DL models that perform as well as CNN on imagery.

This research is part of a project from the Dutch research institute *TNO*. The project develops methodologies for automatic classification of highway scenes. The highway scenes consist of roads with road markings, guard rails, gantries, road side objects and vegetation and is captured by MLS. Currently, mapping of road side objects is performed manually. An operator inspects the point clouds and pins

object locations on a 2D CAD drawing. This research proposes a methodology to create training data, perform classification and create the 2D CAD mapping automatically. The accuracy of the CAD drawings is important, because the location of road signs is critical to road safety. After automatic classification the operator has to perform a manual check and add missing or change incorrect object locations. Still, the automatic classification could reduce manual labor.

1.2 RELEVANCE

There is a paradigm shift from 2D to 3D geo-information. 3D information models, like Building Information Model (BIM), replace 2D CAD information resources. It is important to reuse existing 2D information to classify new 3D data. The methodology in this research reuses the known object locations to create point cloud training data for the DL algorithm.

As stated, the application of DL directly to point clouds is non-trivial. Many researchers apply CNN by transforming the point cloud to a raster representation. Depending on the application these methods also require back-transformation of the classification results. Transformation back and forth between different representations introduces computational overhead. With growing quantities of point cloud data transformation becomes a bottleneck. This can be avoided by classification on the original representation of these point clouds. To apply DL to a "raw" point cloud data set Qi et al. [2016] introduce *PointNet*.

In the original PointNet paper Qi et al. [2016] perform classification of an indoor scene. The indoor scene consists of office rooms that partition the data set and provide spatial structure. The methodology by Qi et al. [2016] relies on this structure, while the highway scene in this research and most other outdoor scenes lack this structure. Because most geo-spatial applications consider outdoor scenes, it is important to explore methods to apply PointNet on these scenes.

1.3 RESEARCH QUESTION

The research question of this research states:

To what extent is PointNet suitable for classification of raw point clouds of a highway scene?

The suitability of PointNet for outdoor highway scenes depends on multiple aspects. The feasibility of a methodology with training data from two separate input data sets of an outdoor scene is not yet proven. The representation of points may be different from indoor scenes and is not yet defined. The sampling of points may be different for outdoor scenes. Finally, the applicability of the model on different locations should be tested. The following questions are stated to collectively provide an answer to the main research question.

- *To what extent can training data automatically be generated from point clouds and known 2D object locations?*
- *What is the best way to represent a 3D point for deep learning?*
- *What is the optimal sampling of points for classification of road side objects?*
- *Does the model generalize so it can be used at other locations?*

1.4 SCOPE

The methodology in this research performs the following steps. First, it creates training data from the MLS point cloud and 2D locations of road side objects. With the training data PointNet is trained to perform point-wise classification. Finally, point-wise predictions are mapped to 2D object locations. The classification accuracy of both point-wise prediction and predicted 2D object locations are evaluated.

The goal is not to achieve state-of-the-art accuracy on point cloud classification. The goal is to discover whether this methodology shows enough potential to pursue this direction of research.

The research makes two main contributions to the field of research for point cloud classification. First, the method to create training data from point cloud and object locations. Second, the exploration of the suitability of PointNet for outdoor scenes. To limit the scope it is important to state what is not part of the research.

The PointNet model was not adapted or optimized, but used as-is. Optimization of the model architecture or parameters for the training process, *hyper-parameters*, could improve classification accuracy [Probst et al., 2018]. However, this is beyond the scope of the research.

The point cloud is considered static, the time dimension is excluded. This means the methodology will not be applicable to dynamic data and classification of changing or moving objects. The data set contains X, Y, Z and intensity attributes for every point. The original PointNet paper also uses *RGB color* attributes for each point, these attributes are only available if the point cloud is combined with (panoramic) photographs. This is not the case for the data set in this research, therefore RGB attributes are excluded. Most laser scanning point clouds contain at least X, Y, Z and intensity values. Because the methodology only depends on these values it is applicable to most laser scanning point clouds.

2 THEORETICAL BACKGROUND AND RELATED WORK

2.1 POINT CLOUDS

A point cloud is a set of data points in a spatial reference system. In practice the term is used to refer to 3D points representing the shape, size and location of real-world objects. Point cloud data sets always contain at least X, Y and Z attributes for every point. Other common attributes are *RGB* for color or *intensity* of the laser reflection. The intensity value represents the reflective properties of the object's surface. These attributes provide information on the properties of the scanned object and can be used in the classification process.

Point cloud data is the basis for many spatial information products ranging from Digital Elevation Models (DEMs) to 3D Building Information Model (BIM). To create these products extensive processing of the point cloud is needed. A typical processing pipeline consist of partitioning, indexing, filtering, segmentation, classification, clustering, mapping, reconstruction and visualization.

Methods to acquire point cloud data are *laser scanning*, *radar* or *dense-image-matching* techniques.

2.2 MOBILE LASER SCANNING

A laser scanner consists of a laser, sensor and rotating mirror. The system measures time between emitting a laser pulse and sensing its reflection (Figure 2.1). This setup acquires many range measurements in 360 degrees around the device. Laser scanners typically emit and detect millions of laser pulses per minute.

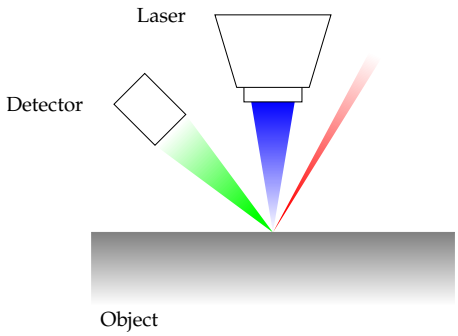

Figure 2.1: The principle working of a laser scanner. A laser emits laser pulses, that refract on the object's surface and are detected by a detector.

A scanner can be mounted onto any platform. To measure the shape of a physical object, e.g. a building or industrial site, laser scanners are often placed on a tripod. This type of scan is called Terrestrial Laser Scanning (TLS) and in practice also refers to a static scan. Given the static position of the scanner the range measurements show the position of the points of reflection relative to the scanner.

To scan large urban areas the scanner is mounted on a vehicle called Mobile Laser Scanning (MLS). For even larger areas, entire cities or countries *Airborn lidar* exists. Most MLS systems have two scanners oriented to scan a line almost perpendicular

to the direction of the vehicle. The system scans continuously while the vehicle moves. Almost perpendicular, because two scan lines crossing diagonally cause less occlusion sideways.

MLS requires additional sensors to determine the position and attitude of the scanner over time. Often a Global Navigation Satellite System (GNSS) determines the global position of the vehicle. The measurement frequency of a GNSS is about once every two seconds. An Inertial Measurement Unit (IMU) increases the measurement frequency of the position and measures the attitude of the scanner over time.

2.3 OBJECT CLASSIFICATION

Object classification is the process of adding the label of the object class to a cluster of points. There are two challenges; which points belong to the same object and what is the class of this object. There are two ways to approach both challenges.

The first approach is to perform segmentation of the point cloud into clusters of points that are likely to belong to the same object. After segmentation the clusters can be classified to determine the class of object (*object classification*). In this approach both segmentation and object classification are non-trivial. However, the top object classification algorithms are all above 90% accuracy on data sets with perfectly segmented objects such as *ModelNet40*. This shows segmentation is the most complex in this approach [Grilli et al., 2017].

The second approach assigns a class label to each individual point (*point-wise classification* or *semantic segmentation*), and clusters nearby points with the same label afterwards. Classification of individual points is also non-trivial [Hackel et al., 2016], [Weinmann et al., 2015a].

Classification algorithms work well for objects, because data from points of the entire object cluster is used. For segmentation or point-wise classification data of a specific point can be used, but this is often not sufficient for accurate prediction. Additional information of neighbouring points could improve point-wise classification. However, the definition of a neighbourhood and the method to select neighbouring points efficiently are non-trivial.

2.4 MACHINE LEARNING

Point cloud classification algorithms rely on features. A feature in this context is any bit of information that informs the classification problem. The development of these features started based on the point, it's neighbourhood and domain knowledge [Vosselman, 2013], [Weinmann et al., 2015b]. Features that are based on *domain knowledge* are often referred to as *manually engineered features*. Multiple aggregations of neighbouring points improve these features [Hackel et al., 2016], [Yang et al., 2017]. Also Yang et al. [2017] implement contextual features from *scene knowledge*. Combining techniques results in extensive sets of features [Weinmann et al., 2015a].

Traditional point cloud classification models are rule-based and rely on manually engineered features. There exist many types of features, e.g. point density histograms, local shape descriptors and many other statistical descriptions. Because of the required domain knowledge and scene knowledge, most traditional models are developed for specific scenes or applications.

More recently Machine Learning (ML) algorithms became very popular. ML models learn to predict output with manually engineered features as input. Learning the model instead of manually engineering rules reduces the amount of domain or scene knowledge required.

In their paper [Qi et al., 2016] state that creation or selection of manually engineered features is non-trivial and its success very application dependent. Learning methods based on those features are not limited to the original data, but are limited to the capabilities of those features to describe the original data. This introduces an unnecessary limitation for learning algorithms.

Deep Learning (DL) algorithms are a subcategory of ML and are large layered networks of *weights* and *biases*. The values for these weights and biases are retrieved by training the network. Training data propagates through the network in a specific direction, this is the *forward pass*. With the results of the forward pass, the weights and biases are updated in a *backward pass*. The network is capable of deriving *abstract features*.

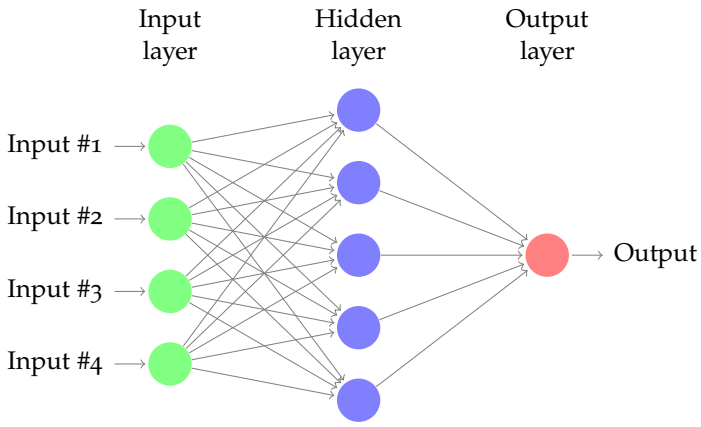

Figure 2.2: A fully connected neural network with one hidden layer. The data propagates from the input (left) through the hidden layers to the output (right).

Learning algorithms require training data. Training a model with this data is called *supervised* learning. During the training phase the model learns to predict output for the input data. If successful, the model is able to predict output for input data that has no annotations, this phase is called *inference*.

In general DL models improve with more training data. With each new training sample the *abstract features* become more general. Training a DL model with too little data may result in *over-fitting*. To over-fit means the model's *abstract features* fit tightly with the training data, but will fail to fit other data including the validation data.

The *abstract features* learned by DL in combination with large quantities of available training data have proven very successful in image classification [Krizhevsky et al., 2012].

2.5 CONVOLUTIONAL NEURAL NETWORK

For classification of images Convolutional Neural Networks (CNNs) have dominated the field of DL algorithms. CNN are similar to Neural Network (NN) (Figure 2.2), but use *convolutions* and *pooling* to reduce the number of trainable parameters. Reducing parameters reduces computation cost and improves the ability to generalize.

The convolutional layer is the reason for the success of CNNs. Convolution is the product of two functions which results in a third function. For the case of image classification the input image can be described as $i(x, y)$, a pixel value for every position of x and y. The second function is a kernel $k(x, y)$, this kernel is a 2D matrix too. In a 2D convolution the kernel slides over every pixel and computes the product of the pixel with neighbouring pixels and the weights of the kernel. This

method incorporates spatial information from the image into the third function $f(x, y$, the feature map (see Figure 2.3).

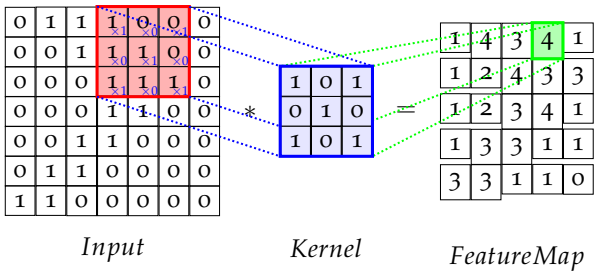

Figure 2.3: A convolution with a 3x3 kernel. The kernel slides over the input image and their product is the feature map.

Pooling is used to downsample the input or feature maps. Besides reducing trainable parameters, the pooling layer also increases the area affected by the kernel in the next convolution. The combination of convolutions and pooling creates a pyramid structure. At the bottom of the pyramid (close to the input) small scale features are detected, such as edges and textures. Near the top of the pyramid higher scale features are detected, such as parts and objects [Olah et al., 2017]. Figure 2.4 shows an example of *max-pooling*. Max- and average-pooling are the most common pooling variations. With max-pooling the highest value input within the kernel is selected, with average-pooling the average value.

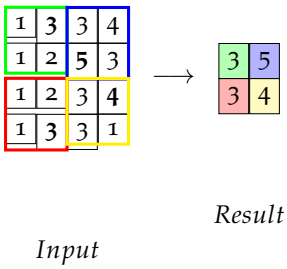

Figure 2.4: Max-pooling with a 2x2 kernel.

To apply CNN to point clouds similar to image classification the representation can be transformed. Point clouds can be transformed to a voxel representation, voxels are structured in a 3D raster [Zhou and Tuzel, 2017]. Alternatively, multiple 2D projections can be made from the point cloud. These projections can be classified with a CNN and the results back-transformed to a point cloud representation. Recent papers focus on the application of Deep Learning (DL) algorithms directly on raw point cloud data. This seems to be a promising method for semantic segmentation [Qi et al., 2016], [Qi et al., 2017].

2.6 LEARNING ON UNSTRUCTURED POINT SETS

Learning directly from raw point clouds is no trivial task. As stated before, DL gained popularity for its success in image classification. Images are a structured data type, a gray-scale image is a collection of values stored in a 2D raster. Every value in this raster is spatially related to its neighbouring values. A point cloud is an unstructured collection of points. There is no specific order of points and no spatial relationship between a point or the next in the collection.

To make DL networks invariant to the order of the points, *permutation invariant*, Ravanbakhsh et al. [2016] introduce an additional layer to transform the input. With

this layer the resulting abstract features are identical despite variations in the order of the input. Qi et al. [2016] implement this layer in *PointNet* and are the first to perform classification directly on point clouds. PointNet achieves 48% average class accuracy, or MIOU, on the indoor benchmark dataset *S3DIS* with 13 classes.

PointNet combines the learned features of all points in an area into a local feature. Engelmann et al. [2017] suggest to learn features on multiple scales of neighbourhoods. Using multi-scale neighbourhoods is similar to the methods by Hackel et al. [2016] and Yang et al. [2017]. PointNet++ is an implementation of this improvement by Qi et al. [2017]. The effect of multi-scale neighbourhoods in DL networks for point clouds is similar to the spatial filters in networks for image classification. However, to ensure permutation invariance the multiple scales remain very local.

An alternative approach comes from the field of *graph theory*. The representation of point cloud data as a graph is common. An example of this is the *mesh* data type, a connected point cloud to represent surface reconstruction. For DL classification of graph structures Kipf and Welling [2016] and Simonovsky and Komodakis [2017] propose spatial filters on the edges in a graph. Landrieu and Simonovsky [2017] proves the scalability of this approach and the application on point cloud data sets. In a recent paper Wang et al. [2018] claims state-of-the-art over PointNet, with a Graph Convolutional Neural Network (GCNN) with dynamic neighbourhood definitions. The GCNN achieves 56% MIOU on the *S3DIS* data set.

As stated in the introduction (Chapter 1) point cloud classification is an active field of research. The PointNet network was released at the start of this research. Therefore PointNet is part of the methodology in this research (Chapter 4). However, during the period of this research PointNet++ and the dynamic GCNN were introduced by Qi et al. [2017] and Wang et al. [2018] respectively. In the conclusion (Chapter 7) these progressions in classification of point clouds are included.

2.7 POINTNET

For point-wise classification (or *semantic segmentation*) PointNet takes input samples of a point cloud and returns class labels for each point of the input. Besides point-wise classification the PointNet paper introduces similar architectures to perform *Object classification* and *Object-part segmentation*. Figure 2.5 shows the architecture of PointNet, the *Semantic segmentation* part of this network is used in this research.

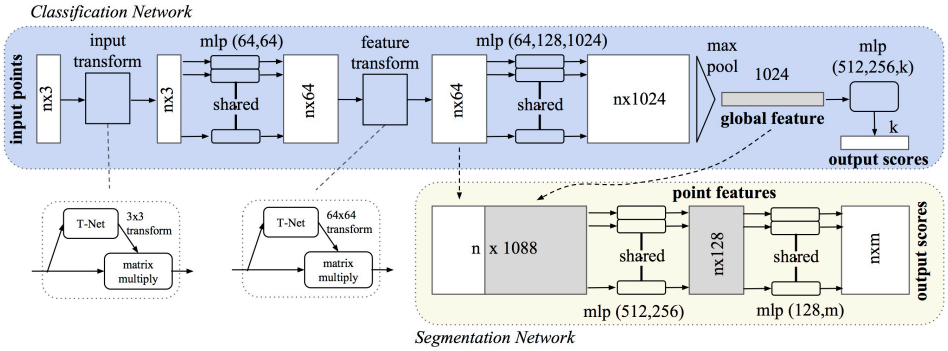

Figure 2.5: PointNet architecture consisting of a classification and segmentation network [Qi et al., 2016].

The input for PointNet is a 2D matrix *attributes x number of points* (see Figure 2.6). Qi et al. [2016] partition the point cloud data set into 1 by 1 meter grid cells. The points in a grid cell are a training sample. To keep the input dimensions the same the *number of points* in a training sample is fixed. The *number of points* is a free parameter, for the indoor data set *S3DIS* Qi et al. [2016] uses 4096 points per 1 by 1 meter sample.

The kernel in PointNet's convolutions is one-dimensional. One-dimensional kernels create features for each point instead of including multiple (unrelated) points into the feature map. Multiple convolutions with these one-dimensional kernels result in *Point Features*.

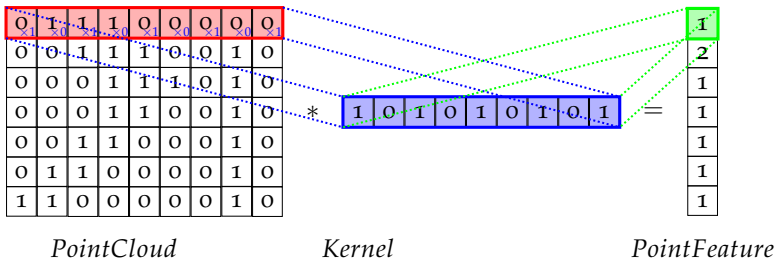

Figure 2.6: Convolution with a $9 * 1$ kernel

PointNet creates a *Global Feature* by *max-pooling* all point features of a sample. Both the *Point Feature* and the *Global Feature* are permutation invariant. The *Global Feature* is concatenated to every individual point feature. The final combined feature used for classification therewith contains information on the specific point and all the points in that specific sample.

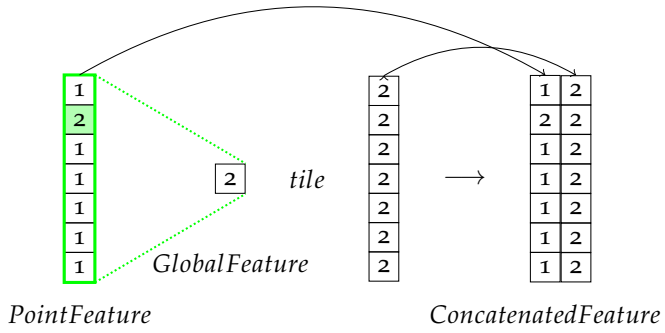

Figure 2.7: Max-pooling of *PointFeatures* into *GlobalFeature* and concatenation of *GlobalFeature* to each *PointFeature*

3 DATA, SOFTWARE & HARDWARE

3.1 DATA

3.1.1 Mobile Laser Scanner point cloud

The first type of input data for this project is Mobile Laser Scanning (MLS) point cloud (see Chapter 2). The point cloud is the main input data set, it contains the geometrical information for classification. The point cloud is combined with the point locations of the objects in the highway scene to create the training data set.

The attributes for each point are; X, Y, Z and Intensity. The intensity attribute is the intensity of the laser reflection on the surface of the scanned object.

3.1.2 CAD 2D map

The second type of input data is a 2D CAD file. The CAD files contain 2D point locations of objects in the highway scene and are combined with the point cloud data to create labels for each point in the point cloud data set.

The CAD files are created manually by an infrastructure planner (see Chapter 1). The object locations are taken from MLS point clouds by visual inspection.

The file contains records in various geometric types *Points*, *MultiPoints* and *LineStrings* in different layers. The object locations are stored both in the Point and MultiPoint geometric type. The different object classes are stored in different layers.

3.1.3 GPS POS dataset

The third data type is a trajectory of Global Navigation Satellite System (GNSS) measurements. The trajectory is the position of the MLS over time. It is used to compute a spatial reference for each point in the point cloud data set. This spatial reference is the point's distance to the trajectory, or assuming the MLS is a car, the road.

The GNSS sensor measures at an interval of 2.5 seconds. The vehicle travels at a speed of circa 80 km/h, therefore this produces a rather coarse trajectory. To interpolate the GNSS measurements an Inertial Measurement Unit (IMU) measures orientation and speed constantly.

3.1.4 Ring Groningen

The first highway scene covers about 12 kilometer of the highway *Ring Groningen* in Groningen, the Netherlands. The scene consists of large stretches of highway, intersections and overpasses (see Figure 3.1).

The scans are acquired on a sunny summer day, the 27th of August 2016. These weather conditions are perfect for laser scanning, because there is little humid in the air or water on object's surfaces to refract laser pulses.

The 2D CAD file of this area contains objects of different classes, four classes are selected because these occur at least a 100 times. Less than 100 objects for a class are expected to be too little to train a DL network (see Chapter 5). The classes are the *Lamppost*, *Road sign*, *Hectometer sign* and *Traffic light*. In total 2811 objects of the

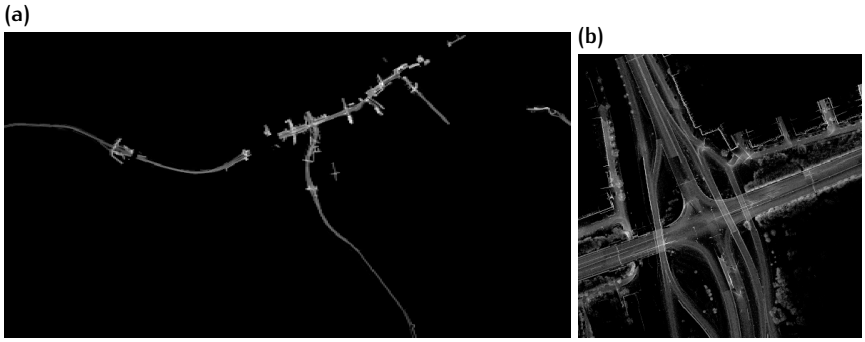

Figure 3.1: Coverage area (a) and close-up (b) of the Ring Groningen MLS point cloud, the points are aggregated and colored by point density.

selected types are mapped in the Ring Groningen data set, the frequencies of objects and points belonging to the objects per class are shown in Figure 3.2.

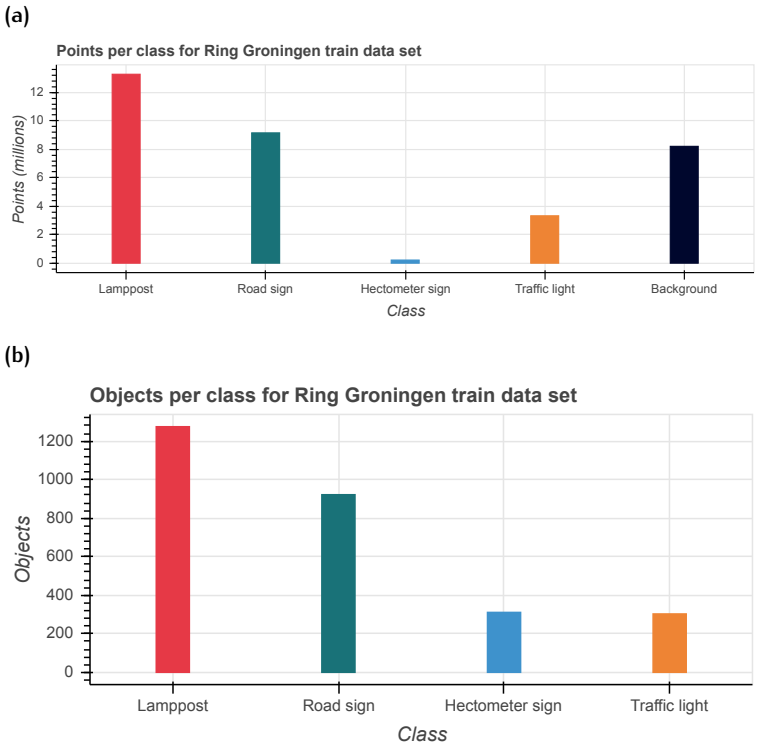

Figure 3.2: Point frequency (a) and object frequency (b) per class for Ring Groningen train data set

3.1.5 Badhoevedorp West

The second highway scene covers 3.4 kilometers of the highway *Badhoevedorp West* just north of Schiphol Airport. The scene consists of an S-shape stretch of highway with an intersection and highway exit (see Figure 3.3).

The scans are acquired on a rainy autumn day, the 9th of November 2017. Compared to the conditions of the *Ring Groningen* data set, these weather conditions are less ideal. Rain causes refraction of the laser and less accurate measurements [Rasshofer et al., 2011]. Another difference between scans acquired in autumn and

summer is the representation of vegetation. In autumn trees and shrubs lose their leafs which makes vegetation less dense causing less occlusion.

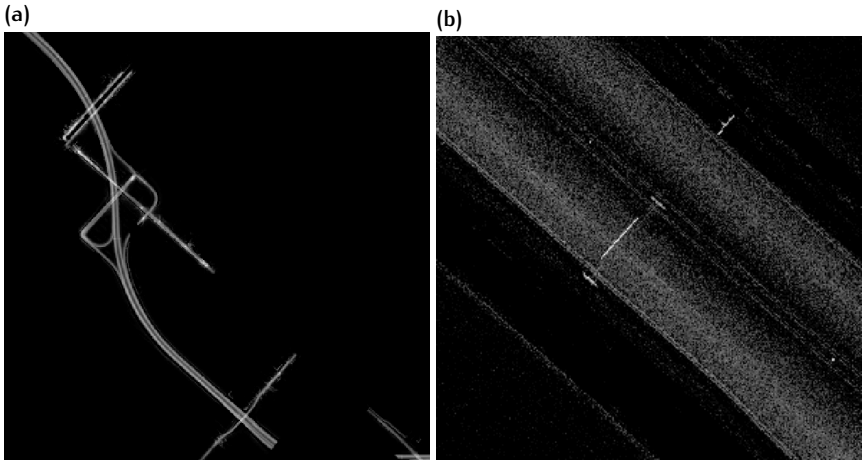

Figure 3.3: Coverage area (a) and close-up (b) of the Badhoevedorp West MLS point cloud, the points are aggregated and colored by point density.

Of the four classes from the *Ring Groningen* scene, three also occur in the *Badhoevedorp West* scene. In total 801 objects of the classes *Lamppost*, *Hectometer sign* and *Traffic light* (see Figure 3.4). The objects are represented by about 899M points, their class frequencies are shown in Figure 3.4.

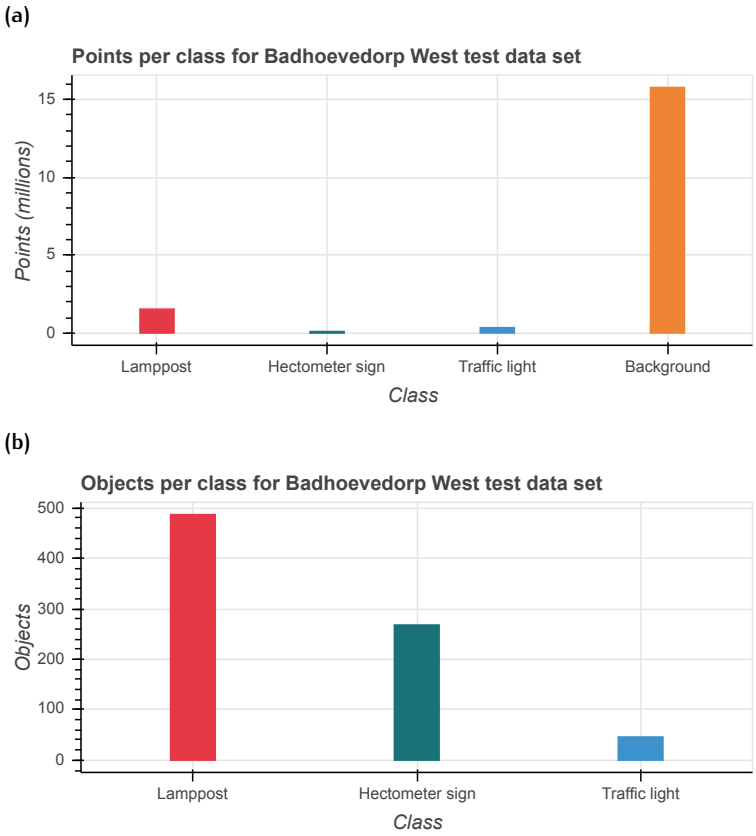

Figure 3.4: Points frequency (a) and object frequency (b) per class for Badhoevedorp West test data set

3.2 SOFTWARE

The entire methodology is implemented in *Python*, using the following libraries. *Numpy* for numerical processes, *Pandas* for dataframe operations and *Dask* for enabling Pandas operations in parallel. Spatial operations rely on *Geopandas* and *Scipy spatial* and graph operations are implemented using *NetworkX*. The deep learning model was implemented in *Tensorflow* by Qi et al. [2016]. Finally, *Bokeh* enables most of the visualizations and is extended with *Holoviews* for visualization of larger amounts of data. *iPyvolume* enables visualization in 3D for point sets and other geometric shapes.

For storage of data in between processing steps *Apache Parquet* is used with a Python interface *fastparquet*.

3.3 HARDWARE

The setup for this project involves a single *Jupyter Notebook* running on a remote server. Specifications for the machine are;

- **CPU** Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz
- **GPU** GeForce GTX 1080 Ti
- **Memory** 64GB
- **OS** Ubuntu 16.04.3 Xenial LTS

4 METHODOLOGY

4.1 INTRODUCTION

This chapter describes a methodology to perform classification of outdoor point clouds and builds on technologies listed in the Related works Chapter 2. Figure 4.1 shows the different processes, files and points of evaluation for this methodology.

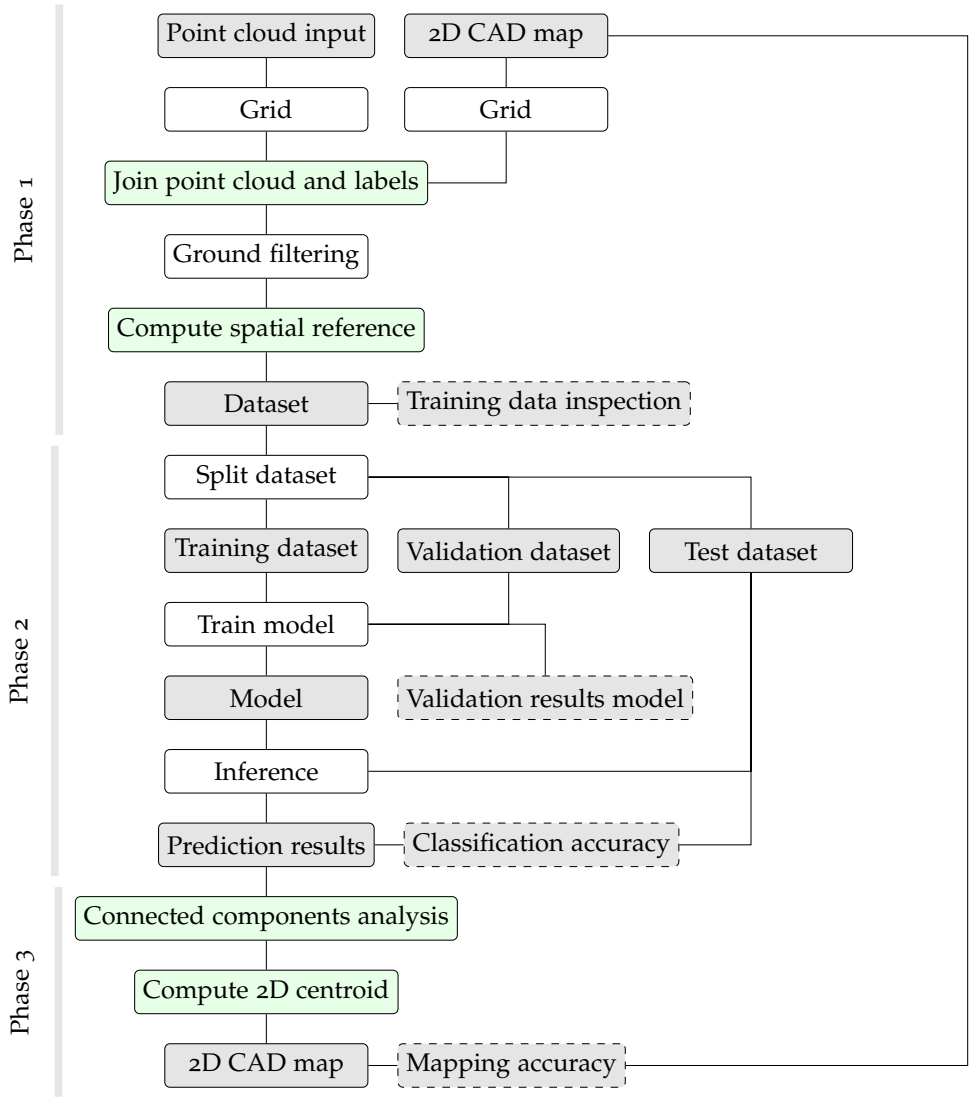

Figure 4.1: Flow diagram of the methodology with processes (white), contributions (green), input/intermediate/output files (gray, solid border) and points of evaluation (gray, dashed border)

The aim is to perform point-wise classification of a point cloud using deep learning. The application of deep learning requires a training data set. To perform point-

wise classification the training data set should have a label for each point. The first phase creates a data set that contains points with corresponding class labels (see Phase 1 in Figure 4.1). To create this data set it integrates the data from a raw 3D point cloud with the 2D CAD map that contains locations of objects. First, the two data sets are partitioned, then a spatial join selects 3D points to assign a specific label from the 2D CAD map. Incorrectly labeled points are filtered by applying ground surface filtering. Finally, several spatial references are computed for each 3D point and values are normalized. Of these steps the process to reuse existing 2D data and the computation of multiple spatial references are novel contributions.

The next phase is training the DL model (see Phase 2 in Figure 4.1). Standard procedure is to divide the data set into three subsets, a training, validation and test data set. The training and validation data subset are used to train the model and validate the training process. The test data subset is used to test the model for final point-wise classification accuracy.

After application of the model to the test set the point-wise predictions are clustered (see Phase 3 in Figure 4.1). The clusters' 2D centroids are the final predicted object locations and are compared with the input CAD file to evaluate mapping performance.

4.2 GRID PARTITIONING

The methodology involves processing of two large input data sets to create a labelled data set. Memory limitations require partitioning of the data set. There is a memory limitation for processing the data before DL, but also the DL network itself can only process batches of data at a time because of limited GPU memory.

A two-dimensional grid is the simplest way to partition space. The grid partitions space into *grid cells* based on a coverage area as described by the lower and upper bound of the X and Y values of all points. A lower XY-pair and the *grid size* describe each of its cells boundaries. This approach is identical to the original PointNet paper, where a 1 meter grid size is used. The grid sizes used in this project are 1, 2, 5 and 10 meters. Larger grid sizes are used because objects in a highway scene are larger and farther apart.

The *Morton code* (or *Z-order*) provides an efficient spatial index to optimize processing. It computes an one-dimensional index value for the X and Y combination of each grid cell by bit-wise interleaving the binary representations of X and Y Morton [1966]. A spatial index for both grid cells and object locations enables filtering grid cells that do not contain an object at all, which drastically reduces the amount of data to process. Also it reduces the computational expensive search for points that are near the object by only considering points within the grid cell instead of all points (see Section 4.3).

4.3 ASSIGNING LABELS

The second phase of the methodology focuses on the creation of one data set with points and labels. The model architecture designed by Qi et al. [2016] requires training with a data set with point-wise labels to perform point-wise classification. To create this data set the point cloud and 2D CAD input are joined.

The join is performed in three steps. The first step is to match grid cells from both data sets by *Morton code*. The next step is to create a 3D cylindrical buffer around each 2D CAD point. The point specifies the center of the cylinders base, its radius and height are related to the objects class. Finally, all points of the point cloud that are within this buffer are labeled the class of the 2D point (see Figure 4.2). The remainder of the points in the grid cell is labeled *background class*.

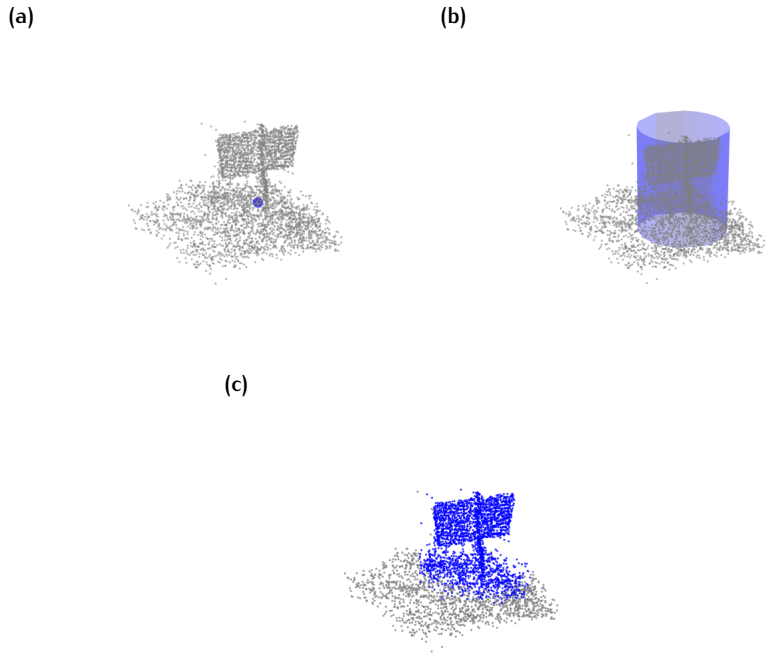

Figure 4.2: Spatial join of raw point cloud and 2D CAD map input. The input of the spatial intersection a point cloud (gray) and 2D object location (blue) (a). A buffer around the 2D object location (blue) intersects with the point cloud (b). The points in the buffer are assigned the object class (c).

However, at the base of each object are also *ground surface* points (see Figure 4.2). These points are within the cylindrical buffer, but are not part of the object. This step of the methodology filters those points and assigns the background class label. The ground surface points are filtered after assigning labels to avoid processing grid cells that do not contain an object at all. Grid cells without objects contain only background class points and therefore no potentially miss-classified object points.

The method to filter the miss-classified ground surface points consists of two steps. First, the points in the grid cell are divided into subsets of points along the Z-axis. A *frequency histogram* shows the amount of points per subset and the largest subsets of points are selected. The Z-range with the ground surface points is expected to be a large subset as the ground surface is a horizontal surface with many points. The second step is to analyze the variance ratio between the X, Y directions and the Z direction. The subset with the least variance in the Z direction is assumed to be the ground surface. The variance ratio indicates "flatness" of the subset and the ground surface is expected to be horizontal. The points in the Z-range that are part of the ground surface are labeled *background class*.

The ground filtering finalizes the process of labeling the data set. The result is an automatically generated ground truth, with the notion that previous processing steps are not perfect and so the ground truth is not 100% accurate.

4.4 SPATIAL REFERENCE

The points in the ground truth data set are XYZ coordinates in the *Amersfoort / RD New* spatial reference system ($EPSG$: 28992). These coordinates represent the point its absolute position in the Netherlands. To provide the model with more information alternative spatial references can be computed. In the original paper

of PointNet the authors use a spatial reference for each points position in a room (the authors apply PointNet to an indoor point cloud) and its position in the grid cell. For this project all points have a position relative to the road and a certain position within the grid cell too. In theory these spatial references provide similar information for each point. From the position in the room arrangement of furniture can be learned. From the position relative to the road a zonal arrangement of road side objects. Expectations are that the distance of an object to the side of the road is standardized and part of road safety regulations.

To represent each point this section introduces three spatial references. The first is based on the original XYZ coordinate, let this be the *global* spatial reference. The second spatial reference is the position of a point within the grid cell, let this be the *local* spatial reference. Lastly, the position of a point relative to the side of the road, let this be the *trajectory* reference.

To compute the global reference the origin of the coordinates is translated to the center of the point cloud data set. The local reference is every points position relative to the center of the grid cell. To compute the trajectory reference the trajectory data set is used (Chapter 3 for more detail). Because the vehicle drives on the road, the trajectory path can be seen as the length-axis of the road. The distance from each point to the nearest point on this trajectory is measured. This distance contains a horizontal and vertical component. These values are the trajectory reference. Finally the values of the global and trajectory are normalized for the entire data set and the values of the local reference per grid cell. Normalization remaps the values to a -1 to 1 scale. Normalization is needed for the learning process of the model, because the initialization of the models weights is by default with small random values. Without normalization the learned parameters may never reach the domain of the data.

The ground truth data set now contains points with three spatial references, a class label and multiple grid cell identifiers.

4.5 SAMPLING

The data set created in the previous sections is split into three parts, a training set, a validation and test set. It is important to keep these sets strictly separated, because a test with training data does not evaluate the model's ability to classify unseen data. Let all the points in a specific grid cell be a sample. Two methods to split the data set are used; *random* and *spatial*. For a random split samples are randomly selected from the entire data set. For a spatial split all the samples in the data set are divided into three parts, e.g. north/mid/south. The different classes of objects are not equally distributed throughout the scene. Therefore a spatial split could result in certain classes of objects missing in the resulting parts of data set. If a class is missing in the train set the model will not learn this class and if it is missing in the validation or test set it cannot be verified that the model has learned this class. Therefore the spatial split is only used in this project for experiments covering two input scenes.

The samples contain varying amounts of points, because the input point cloud is sparse and point density varies. The model requires samples to be of the same size *n number of points*, which is a free parameter. Sampling is applied to select n points from all points within the sample. This is under-sampling if n is larger than the total amount of points in the sample or over-sampling if the sample contains less than n points. The PointNet paper states that the model is robust to missing or duplicate points. If the sample contains less points than a certain threshold the sample is discarded, this threshold was set to 100 points for this research. Three methods for sampling are used; random, grid-wise with density preservation and grid-wise with density flattening. The original PointNet paper uses random sampling, the two additional methods are expected to improve classification performance because they

result in a more consistent point distribution. Random sampling selects n points randomly, all points have the same probability of being selected. The grid-wise methods first bins the sample in three dimensions. To preserve density variations in the sample, probabilities for selecting points are based on the amount of points in each three-dimensional bin, called a voxel. The third alternative forces even point density by lowering probabilities for points in voxels containing many points.

The more data can be used for training the model, the better the model becomes. This research applies two methods to increase the amount of training data. Many samples contain more points than are used after sampling, the first method focuses on reusing these points. By sampling these samples multiple times more training samples are generated (see Figure 4.3). This method might perform better than simply increasing the number of points, because the GPU Memory limits the number of points and more points in one sample might not necessarily improve performance (see Chapter 5). In Section 4.3 the grid cells without objects are excluded from the data set. The second method to enlarge the training data set is to reintroduce a certain amount of these samples.

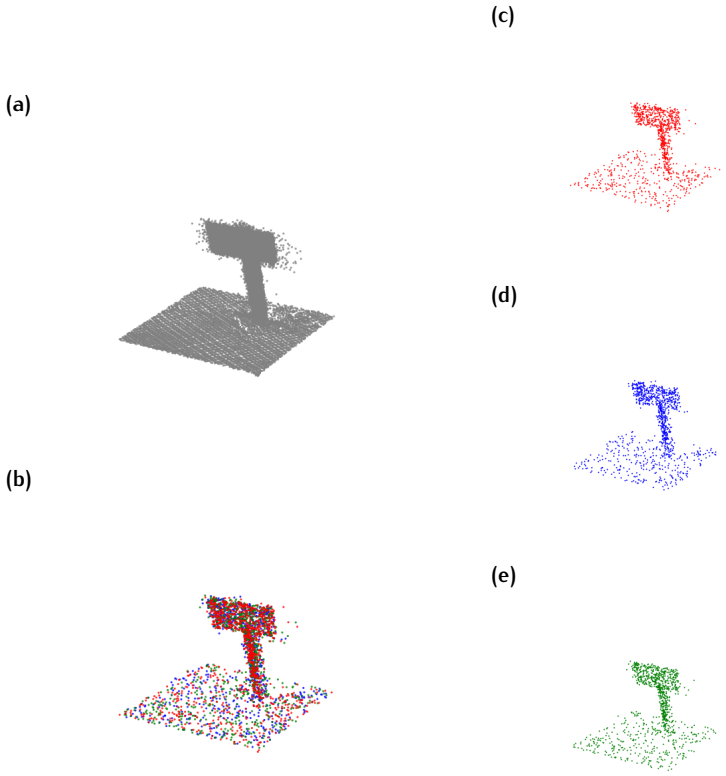

Figure 4.3: A grid cell with circa 12.000 points (a) subsampled to 3000 points (b), can generate 3 training samples (c,d,e) to create more training data.

4.6 DEEP LEARNING

The model is trained by feeding batches of training data, each batch contains multiple samples. The model learns by going through all batches and updates its weights after each batch. After all training data is processed the model is applied to the validation data to show what it has learned. This process of training and validation is repeated multiple times, every iteration is called an epoch. Training the model

requires the specification of model specific parameters, as discussed above (spatial references, grid size, number of points, sampling method and data split method). In addition the model requires specification of *hyper-parameters* related to the model structure (weight initialization, drop out, activation function and loss function) and related to training (batch size, learning rate, learning rate decay and number of epochs). The model is used as described in the original PointNet paper with similar hyper-parameters, except for batch size. The batch size is lowered to allow for experimentation with higher amounts of points per sample, the total amount of batch size times samples times number of points is limited by the GPU Memory.

After processing all epochs the validation results and the models weights are stored. The validation results provide information on how the model has evolved and its expected performance. The model can now be applied to new data just as it was applied to the validation data set, given this new data is prepared in similar fashion. The application of the model to the test data set results in point-wise label predictions for data that has not been part of the training process.

4.7 OBJECT DETECTION

The final goal of this methodology is to produce a 2D mapping of the predictions. The prediction results for the test data set are still unrelated 3D points, be it with a class label. This step in the methodology aims to create the relation between points that belong to the same real-world object. This means to cluster points with the same predicted label that are nearby.

Two methods to cluster points are used. For both methods the background class points are excluded. The first method simply groups all points with the same predicted label that are in the same grid cell. This method assumes that there is only one object of this predicted label in that grid cell.

The second method is the *Connected Components* method. This method involves three steps. First the points are represented as nodes in a graph. Then nodes are connected with an edge if the points are less than 0,2 meters apart and are predicted the same class label. If there are objects in the sample this results in multiple sub graphs separated by distance or class difference. Finally, for each sub graph the points are assigned an object identifier.

Comparable to the isolates in the previous steps, very small components could occur. To prevent mistaking a single object for multiple objects components of an insignificant size are discarded. The amount of points for a connected component to be insignificant depends on the resolution (total amount of points per hash) used in classification.

For all of the remaining components a bounding box can be fitted around all points. To map the objects to a two-dimensional CAD file (similar to the input file) the centroid is projected on the ground surface.

5 EXPERIMENTS AND RESULTS

5.1 EXPERIMENTS

5.1.1 Introduction

The previous chapter describes the methodology, it depends on multiple parameters. This chapter introduces experiments to test the methodology. Every experiment varies a parameter to test an assumption. The results (Chapter 5) show the evaluations of these assumptions. The evaluations answer the research questions and the main research question *To what extent is PointNet suitable for classification of raw point clouds of a highway scene?* indirectly.

5.1.2 Training data

The first research sub-question states *To what extent can training data be created from point clouds and known object locations?*. As mentioned before, DL algorithms require large amounts of training data. To answer the question it is important to assess the quality and quantity of training data.

The quality of the training data is evaluated by visual inspection. The assumption is that training data can be created by integrating raw point cloud data and the provided CAD files containing point locations for the objects. The requirement for training data is that each point of the point cloud is assigned a class label. These labels are ground truth for further analysis. By inspection of multiple training samples and estimation of miss-classified points an overall accuracy of the ground truth is estimated.

The quantity of training data is evaluated by evaluating the classification accuracy of different classes. There are different quantities of objects for each object class, which may result in different classification accuracy (see Figure 3.2 in Chapter 3).

5.1.3 Spatial reference

The next research sub-question is *What is the best way to present spatial information for deep learning?*. The methodology introduces three spatial references (Chapter 4). The position of a point within the Netherlands, or the *global* reference. The points coordinates relative to the center of the grid cell, or the *local* reference. And the distance to the trajectory, or the *trajectory* reference.

The first assumption relates to the global and local spatial reference. The assumption is that if the model learns the local reference, the global reference will be redundant. To test the contributions to classification accuracy of three spatial references different models are trained and tested, with the local reference, with the global reference and with the local and global reference combined.

The *trajectory* reference is assumed to increase the models performance. Because the distance between a point and the road is expected to be a discriminating property. To test this hypothesis the model is trained and tested with the local and trajectory reference and compared with the results from the model trained only with local reference.

5.1.4 Point sampling

The next research sub-question is *What is the optimal sampling of points for classification of road side objects?*. The point density of the input point cloud is not homogeneous. Variations in point density are inherent to laser scanning. The training samples are of fixed number of points, this is a model requirement. The point sampling to create a training sample, depends on the grid size, number of points and the sampling method.

With larger grid sizes the chance a sample contains multiple objects of different classes increases. The global feature that is computed for the entire sample will contain point features from different classes. The assumption is that this could decrease the classification. The experiment involves training and testing four different models with data samples with grid sizes 1, 2, 5 and 10 meters.

The number of points in each training sample is a model parameter. However, every grid cell contains a maximum number of unique points. Grid cells with that contain less points than the parameter specifies are randomly over-sampled by duplicating points. The assumption is that more points improve the accuracy, until more points are duplicated than unique points are added. The experiment to test this assumption increases the number of points in four steps, with 1000, 2000, 4000 and 8000 points. The grid size will be the best performing grid size from the previous experiment.

The variation in point density within the sample depends on the point sampling method. Point sampling is used to create training samples of a fixed number of points. The methodology describes three different methods for point sampling. The expected difference is that random sampling introduces gaps in point coverage. Whereas voxel sampling ensures consistent point density. Therefore voxel sampling is expected to perform better. For the experiment three models are trained with the different sampling methods, the same sampling method is also applied to the testing data on inference.

5.1.5 Generalization

The last research question is *How does the model generalize to another location?*. The previous section elaborated on the performance of different spatial references. The suitability of the methodology depends on the re-usability of the model for different locations. The assumption is that the local spatial reference does generalize to a different location, but the global reference does not. To test whether the spatial references generalize to other areas all the three models (trained with local, global and the combination of local and global) are applied to an area on a different location.

5.2 RESULTS

5.2.1 Introduction

Each phase in the flow diagram (Figure 4.1 in Chapter 4) ends with a point of evaluation. First, the inspection of the generated training data (Phase 1 in Figure 4.1). Second, the accuracy of point-wise prediction (Phase 2 in Figure 4.1) and last the accuracy of mapping the clustered predictions (Phase 3 in Figure 4.1). The prediction accuracy of the model is the most critical assessment of these three. Classification accuracy variations are directly related to variations of the parameters in the training data generation process.

Conducting experiments multiple times shows variation in results. This variation is caused by the random order of batches in a train session. The variation is not significant to evaluation of some assumptions, for others the Discussion (Chapter 6) elaborates on the variation.

5.2.2 Accuracy measure

The evaluation of point-wise classification accuracy requires an accuracy measure. The Mean Intersection over Union (MIOU) accuracy is often used in point-wise classification problems. The MIOU shows the mean of the point-wise classification accuracy per class. For this research the MIOU excluding *background class* is used. This measure is representative because background is the dominant class by a large margin and causes incomparably high accuracy results.

5.2.3 Training data quality

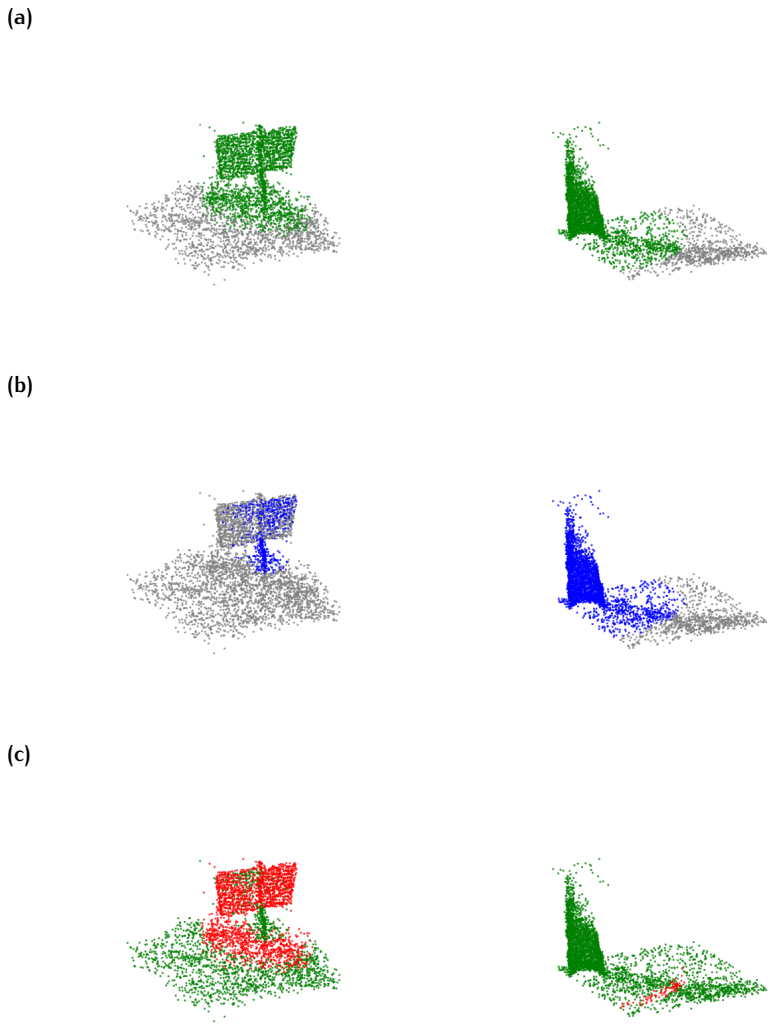

Figure 5.1: The automatically generated ground truth for a Hectometer sign (left) and Lamp-post (right) (a). Followed by the prediction of the model (b) and the difference between ground truth and prediction (c).

Figure 5.1 shows the ground truth and prediction results for two training samples. The estimated percentage of incorrect labels in the ground truth is 12-14%. This estimation is based on visual inspection of multiple samples. Common causes for inaccuracy are grass, overhanging vegetation and guard rails. These types of objects were not filtered by the ground filtering method and are within the intersection

buffer. The figure also shows how the model handles ground points at the base of an object. In some cases the ground points are included, in others excluded. Note that the ground points at the base of each object already cause significant decrease in point-wise classification accuracy. Other inaccuracies are from object variations within a class, several experiments showed that not all objects within a class were equal. For example the *road sign* class contained different types of road signs, e.g. small triangular warning signs and larger rectangular signs with route directions. Also the *traffic light* class contains both road side traffic lights on a pole and traffic lights hanging above the road.

Overall the training data is inaccurate, the error sources mentioned above cause points with an incorrect label, but following experiments prove that the model can learn from this training data.

5.2.4 Spatial reference

The experiment tests the assumption that if the model learns the global spatial reference the local reference would be redundant. Figure 5.2 shows the results for various combinations of spatial references. The model trained with only local spatial reference achieves 32% accuracy. The model trained with global spatial reference achieves 21% accuracy. 21% is the same performance as random class assignment, because there are 5 classes. The combination of the two spatial references also achieves 32% accuracy. This means the model does not learn the global spatial reference and performance is solely based on information from the local spatial reference.

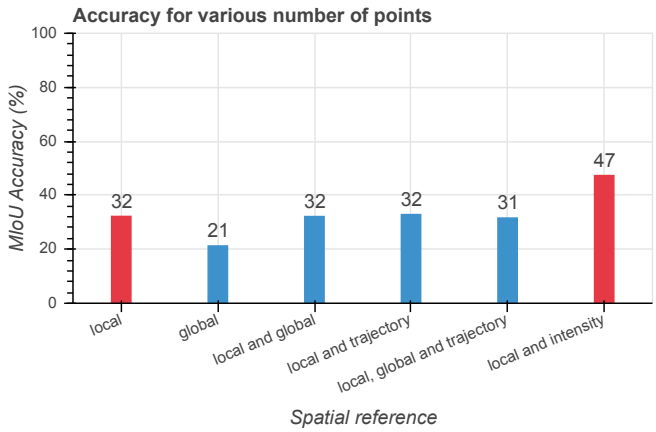

Figure 5.2: Average class accuracy for various spatial references. The red bars indicate contributing attributes. (grid size 5m, 4k points, random sampling)

Another assumption was that the trajectory reference would increase performance. Figure 5.2 shows that the trajectory reference does not contribute to the performance at all. The assumption was based on the idea of a zonal arrangement of objects on the road side. Figure 5.3 shows the frequencies and their distance to the trajectory for different object classes. The figure shows two peaks, one peak around 3 meters and another around 7 meters distance from the road. The 4 meters in between are exactly the width of one lane, which indicates the car changes lanes.

The last bar in Figure 5.2 shows the accuracy of the model trained with local reference and the intensity value. The major increase in classification accuracy for this model shows the importance of this intensity value.

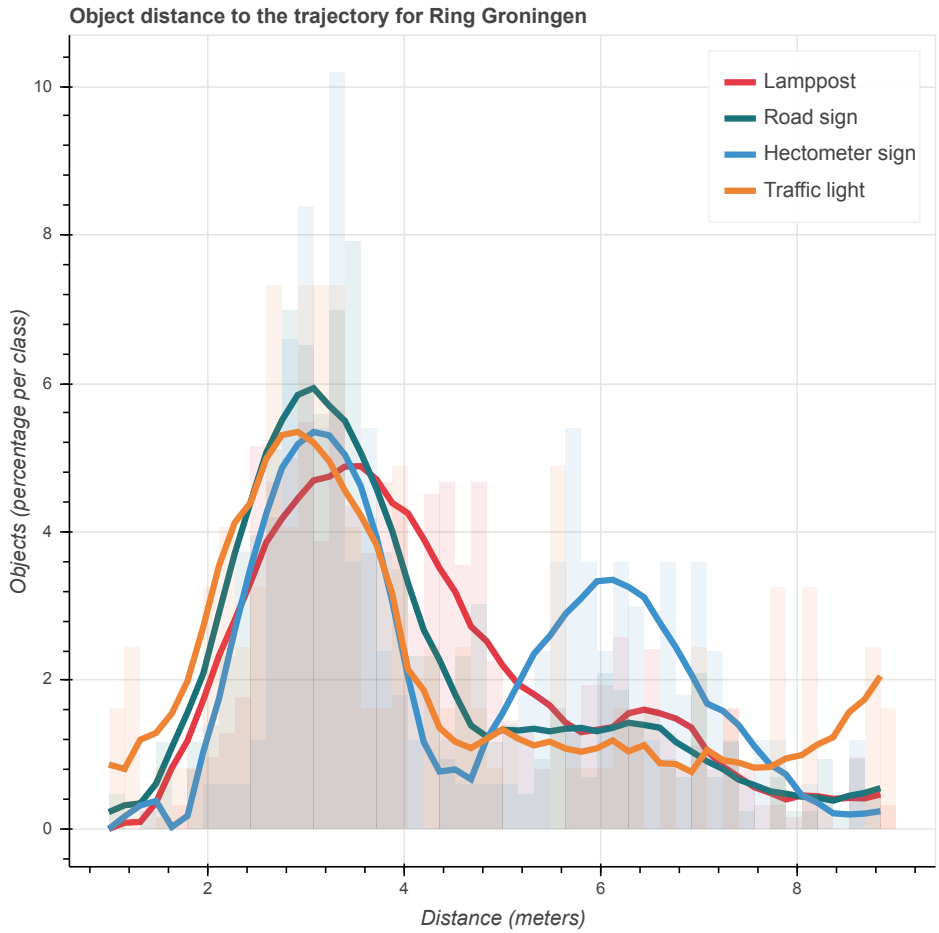

Figure 5.3: The object frequency and distance to the trajectory for different object classes in the Ring Groningen data set.

5.2.5 Point sampling

There are three experiments that provide insight in the aspect of point sampling. The first experiment focuses on the grid size parameter. The assumption is that a larger grid size is bad for classification performance, because of a higher chance that multiple objects of different classes occur within one sample. Figure 5.4 shows that this is not the case. Larger grid sizes achieve better performance, but performance stagnates at 47% for the 5m grid size.

The second experiment tests the *number of points* parameter. The assumption is that training the model with more points per sample results in higher classification accuracy. Figure 5.5 shows that more points indeed increases performance. However, more than 4000 points has negative influence on the classification accuracy. On average a 5 by 5 meter grid cell contains 8000 unique points. In theory a grid cell containing 8000 unique points can be subsampled 8 times to create 8 training samples with 1000 points. This *multi-sampling* generates unique training samples in the sense that the samples contain unique points, while the object is the same. This augmentation method did not contribute to classification accuracy.

The third experiment examines three different sampling methods. The assumption is that random sampling introduces gaps, whereas a voxel (grid) sampling would provide a more consistent representation. The expectation is that voxel sampling performs better than random sampling. Figure 5.6 shows the differences between different sampling methods with 4000 points on 5x5m samples. There is no significant difference between the different methods.

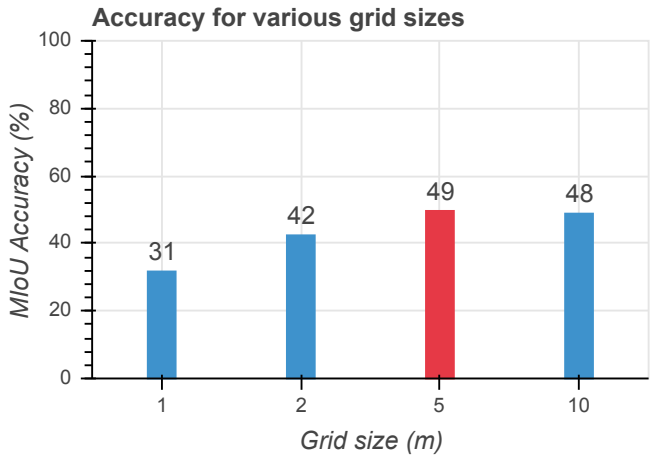

Figure 5.4: Average class accuracy for different grid sizes. The red bar indicates the best performing grid size (local XYZ and intensity, 4k points, random sampling)

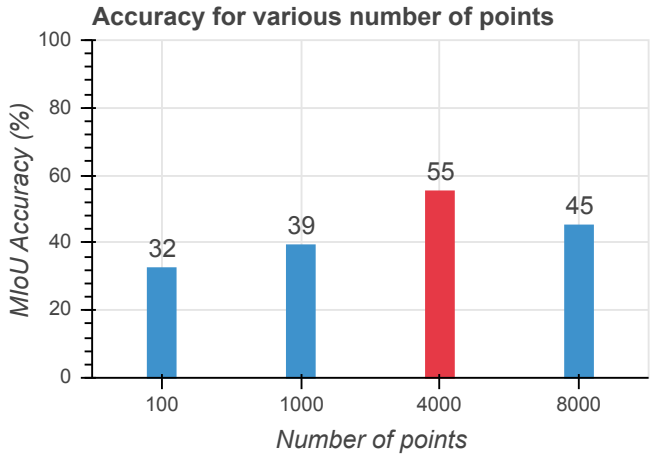

Figure 5.5: Average class accuracy for different number of points per sample. The red bar indicates the best performing number of points. (local XYZ and intensity, grid size 5m, random sampling)

A follow-up assumption is that if there are so many points, the model learns well regardless of the sampling method. To test this hypothesis the same experiment is performed with only 100 points. Figure 5.7 shows that with only 100 points the sampling method does actually matter and the training with consistent point density performs better than the random and per-voxel-random method.

5.2.6 Generalization

The previous experiment proves that the model learns from the local reference, but not from the global reference. The assumption regarding generalization is that the model with local reference does generalize to other areas, whereas the model with the global reference does not. Figure 5.8 shows the average class accuracy for different combinations of train and test scene. The first experiment tests the local, global and combination of references for the model trained and tested with data from the Ring Groningen dataset. Obviously, the results are similar to the results in Figure 5.2. The second experiment tests these models trained on Ring Groningen with test data from Badhoevedorp West another location. The results prove that the

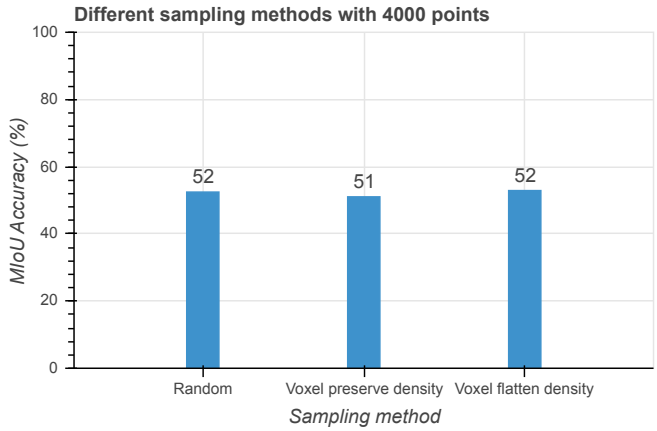

Figure 5.6: Average class accuracy for different sampling methods. All bars are blue, because there is no significant difference in classification accuracy. (local XYZ and intensity, grid size 5m, 4k points)

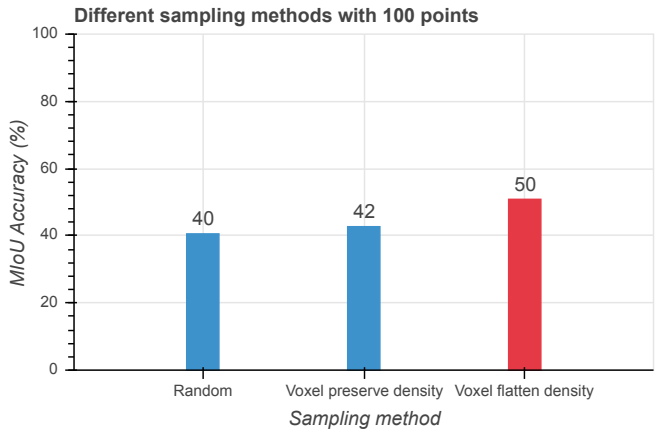

Figure 5.7: Average class accuracy for different sampling methods. The red bar indicates the *voxel flatten density* method performs best with this number of points. (local XYZ and intensity, grid size 5m, 100 points)

assumption is true. The model trained with global or the combination with global reference achieves nearly 0% average class accuracy on the new location.

5.2.7 Overall performance

The individual experiments optimize specific parameters. The combination of the optimal settings should result in a overall best performance. The following results evaluate the implicit assumption of the main research question, the overall suitability of PointNet for classification of outdoor point cloud data. Figure 5.9 shows the classification performance, Intersection over Union (IOU), per class excluding background class. Interesting to note is the similarity between the frequency of points in the data set (Figure 3.2) and the accuracy per class (Figure 5.9)). This similarity suggest that more points for a class result in higher classification accuracy, which confirms earlier stated assumption that more training data increases performance. The average performance of the model is 50% MIOU for the four object classes.

The confusion matrices (see Figure 5.10) show how the classes are confused by the model. Both matrices state the absolute number of total points and objects, followed by the percentages of confused points and objects. In general the classes that are confused most are very similar in shape, e.g. the confusion between hectometer

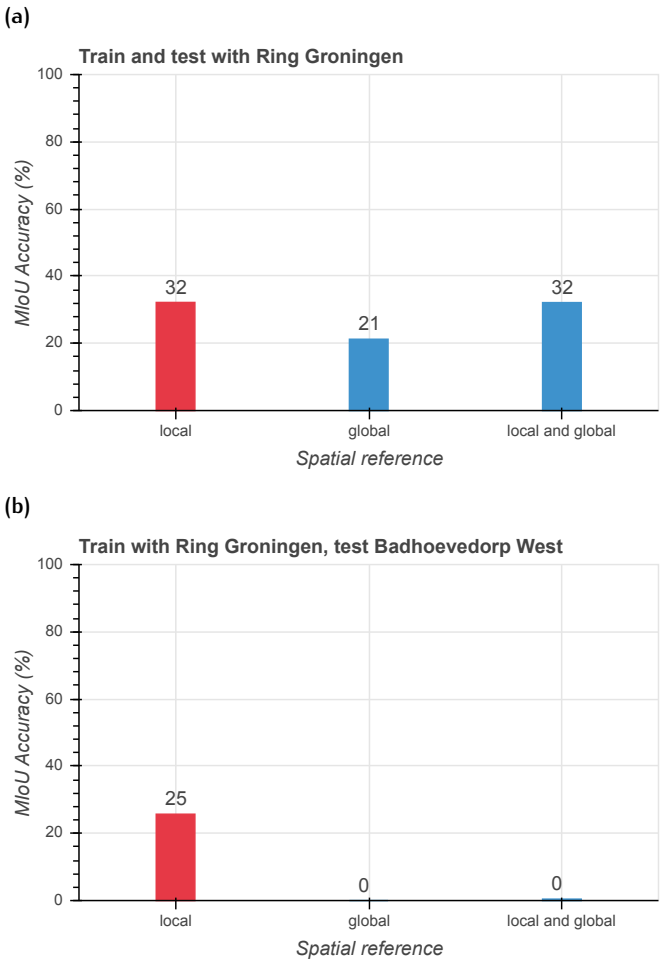

Figure 5.8: Average class accuracy for train and test the model with data of the same area Ring Groningen (a) and with different areas, respectively Ring Groningen and Badhoevedorp West (b). The red bar indicates that the local spatial reference generalizes to other locations. (local XYZ and intensity, grid size 5m, 4k points, random sampling)

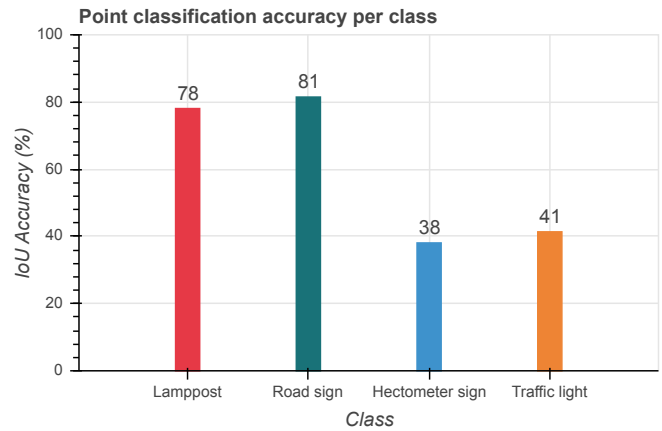

Figure 5.9: Point-wise classification accuracy per class (local XYZ and intensity, grid size 5m, 4k points, random sampling)

signs and road signs. The point-wise classification performance is 50% MIOU. The

final object location accuracy is higher than the point-wise classification accuracy, this is 60% MIOU.

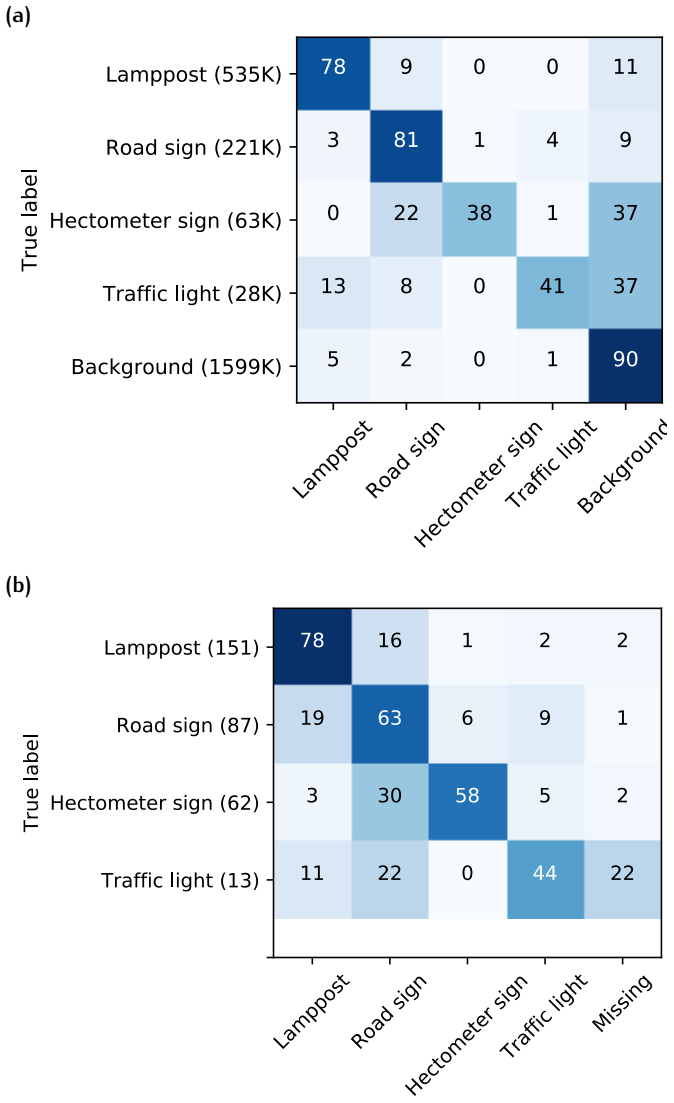

Figure 5.10: The confusion matrix for the point-wise class prediction (a) and the final mappings of each object (b). Each class shows an absolute count between parentheses and a percentage of this count for confusion. The cells are colored by percentage.

6 DISCUSSION

6.1 TRAINING DATA

The results show two samples from the training data set (see Figure 5.1). These samples reveal an important downside of a grid. The boundaries of the grid cells cut through the objects. Objects that are distributed over multiple grid cells are called *edge cases* (see Figure 6.1). Edge cases cause two problems in this methodology. First, samples that contain only half of the object might be harder to classify. Even though classification is done point-wise, the global feature is less accurate if the sample contains only half of the object.

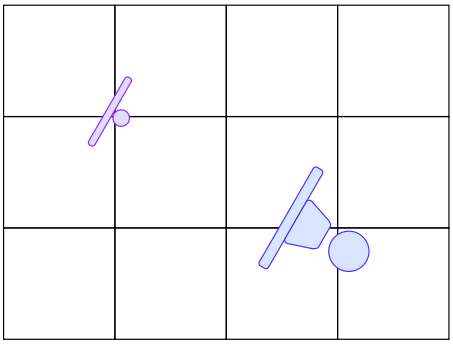

Figure 6.1: A sketch top-view of hectometer sign and traffic light edge cases with a small grid size

The second problem is caused by the spatial intersection. The intersection with a buffer is a simple spatial operation, but computationally expensive. Therefore objects are first matched based on the grid cell identifier. The consequence is that the intersection method only considers point cloud points within the same grid cell. Parts of an edge case object that are not in the same grid cell as the 2D object location are not labelled.

The method uses a cylindrical buffer with class dependent sizes. The cylinder is efficient because most objects in the highway scene are pole-like structures and the signs or lights attached to the structure can be oriented in any direction. Determining the size of the cylinder, however, is not trivial. In this research the size of the cylinders is based on the object class, though not all objects within a class appear to be identical. This results in object points outside of the buffer and therefore incorrect class labels.

The ground filtering method relies on various assumptions. The most critical of these assumptions is that all ground points are within a single bin along the Z-axis (see Section 4.3). Sloped ground surface and areas with low vegetation (high grass) are exceptions to this assumption, however appear relatively often. This causes ground surface points at the base of objects to end up with an object class label in training data. Throughout the project it appears that this noise is not disastrous for the end result, however does likely decrease performance.

Despite the inaccuracies the model is able to generalize. The estimated training data 12-14% error, however, decreases the maximum achievable accuracy to around 86-88%.

6.2 SPATIAL REFERENCE

The assumption is that if the model learns the local spatial reference, the global reference will be redundant (see Section 5.1.3). This assumption is correct, the model is not able to learn the global reference, it only learns the local spatial reference. If the two spatial references are combined the performance is similar to performance achieved with the local reference only. The model does not learn the global reference, because it does not contain any information to classify other points of the scene. The position of every point in a global reference system is unique, therefore it does not generalize to other points in the scene. The local spatial reference in PointNet is similar to the 2D position of objects in an image for image classification. When the model learns training samples with the object on different positions and rotations within the sample the resulting features will be robust to these variances. With this insight augmentation techniques can be borrowed from image classification for learning on point cloud data sets (see Section 7.2.6).

The second assumption is that the trajectory reference improves classification accuracy (Section 5.1.3). Results prove that the trajectory reference does not contribute to performance. The model performance is similar to the local reference without the trajectory reference. The distance to the road, as implemented in this research, is not a discriminating property to classify points (Section 5.2.4). The reason for this could be that the error is larger than the information. A possible cause for a systematic error in this value is the scanning vehicle changing lanes every once in a while. Because the trajectory of the car changes the distance of points to the trajectory changes. The width of one lane is just as wide as the road side area in which the objects are placed. Figure 5.3 shows the object frequency per distance to the trajectory and rules out this reasoning. It proves the car changing lanes is actually recognizable in the trajectory reference. This systematic error is not larger than the information and two peaks are clearly visible for every object class. The problem is these peaks are aligned, at the same distance to the road. Still, the zonal arrangement of objects at the side of the road could be at sub-meter level. If this is true, differences of scanning vehicle's position within lane could cause the object's distance peaks to spread. This error can be compensated by first detecting the true side of the road or guardrails. However, this does complicates the methodology significantly.

6.3 POINT SAMPLING

Grid size, number of points and sampling method determine what points are selected for each training sample. The assumption is that larger grid sizes result in lower classification performance (Section 5.1.4). On the contrary, the experiment shows that training samples from larger grid sizes result in higher classification accuracy (Section 5.2.5). The performance stagnates around 5 by 5 meter grid size. If objects are further apart than 5 meters, grid cells still contain points of background class and just one type of object. In this case global features (the feature for the entire sample) describes only one object class. Also, smaller grid sizes cause more edge cases, because there are more edges. Edge cases cause inaccurate class labels in the ground truth, this introduces conflicting information in the training data set. These findings suggest that the best grid size is as large as possible to reduce edge cases, limited by the distance between objects.

The second assumption states that classification accuracy improves with more points per sample (Section 5.1.4). The experiments show that below 4000 points the assumption is true and the performance increases with more points. The model performs best with 4000 points and above 4000 points the performance decreases (Section 5.2.5). The accuracy increases because the point density on real-world objects is higher than point density of the ground surface (see Figure 6.2). With the increase of points per sample the amount of points relative to background increases. This creates a more equal balance between object classes and background classes. DL algorithms are known to perform better with more equal class distributions. The classification accuracy decreases after 4000 points, because on average a 5 meter grid cell contains 8000 points. Therefore to reach 8000 points half of the samples will include duplicate points.

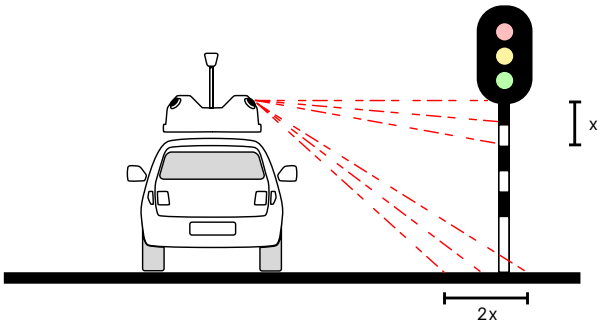

Figure 6.2: A sketch of the MLS scanning a traffic light. The angle with the ground surface results in a lower point density for ground surface then for surfaces of the object.

The third assumption is that voxel sampling methods outperform random sampling. The results show that this assumption is only correct in cases with very little points. The experiment shows that with 4000 points per sample the amount of information is enough to compensate irregularities in sampling distribution for the random sampling method. With 100 points the distribution of those points becomes important. This makes a case for the voxel sampling method that flattens density in circumstances with restricted computation power or time. Because the samples with 100 points contain less data, the model trains significantly faster.

6.4 GENERALIZATION

The results confirm the assumption that the model trained with the local reference does generalize to other areas, but the global reference does not. Section 6.3 on the experiments with spatial references also confirms this assumption. If the model does not learn the global reference, there is nothing to generalize.

The results show that the performance of the model trained with local reference also decreases. There are multiple possible reasons for the performance to decrease. It could be due to the time of acquisition. The first scan was made on a summer day, whereas the second data set was acquired on an autumn day with clouds and a little rain. The shapes and intensity of reflections differ for vegetation in different seasons. Also can wet surface produce difference in reflections of the laser.

To train the model for a certain number of classes it is important that each class is represented in the training data set. If one class is dominantly present in the training data set the model could predict all points to be this specific class and its overall accuracy would still be high. To balance the total number of points for each class in the training data set the model can be trained only with data samples that actually contain an object. This reduces the points belonging to the *background*

class (the dominant class in the scene) However, it also reduces the total amount of training samples.

7 CONCLUSION AND RECOMMENDATIONS

7.1 CONCLUSIONS

7.1.1 Introduction

The motivation for this research was the challenge of automatic classification of objects in a highway scene from MLS point clouds. The main research question is *To what extent is PointNet suitable for classification of raw highway point clouds?* The four research sub-questions contribute to answering this question. The results regarding the first question show that it is possible to automatically create training data with the provided input data sets. The next set of experiments explain how to present the spatial information of this data and how the model generalizes to different locations. Final experiments showed how many points and how to select these points.

7.1.2 Training data

The first research sub-question states *To what extent can training data be created from point clouds and known object locations?* The results show that the method to create usable training data works.

The results and discussion elaborate on many inaccuracies in the training data and possible causes. The largest source of error is the ground filtering method. The ground filtering method is not robust to the variations in real-world data. It is not the correct way to filter ground surface points and many better methods exist (see Section 7.2.5). The spatial intersection method is the correct way to start the labelling process, but requires more refinement.

Even though the highway scene is a real-world data set it is important to note that it is a man-made environment. For safety regulations many road sides are free of vegetation and most objects have standard shapes, sizes, materials and position relative to the highway. This significantly reduces noise in the data set.

To conclude the training data can be made from the point clouds and known object locations. The accuracy of the method to create training data is circa 86%. This means about 12 to 14% of the points are assigned an incorrect label. This inaccuracy is largely compensated by the amount of data and regularization capabilities of DL. Therefore the method is a prove of concept, but requires improvement to eliminate inaccuracies.

7.1.3 Spatial reference

The second research sub-question is *What is the best way to present a 3D point for deep learning?* The research assessed three spatial references. The discussion shows that the global reference has no value in the task of classification. The trajectory reference is a contextual reference that, in this context, does not contribute to the classification. The conclusion is therefore that the best way to present spatial information is the local spatial reference.

7.1.4 Point sampling

The answer to *What is the optimal sampling of points for classification of road side objects?* depends on three experiments. The *grid size*, *number of points*, and *sampling methods* provide insight in how many points and how to sample those points.

The results show that the model performs best with a grid size of 5 by 5 meters and 4000 points. These results are closely related to the object classes, specifically their size and shape. More object points generally improve the classification accuracy, however than 4000 points overall do not improve classification accuracy. This is due to the balance between object points and background points. The background class is dominant in the scene and adding more points decreases balance between classes. For the average accuracy of all object classes 4000 points performs best.

The method to sample these points is irrelevant for 4000 points. The assumption that random sampling could introduce gaps in point coverage is only relevant for amounts of points close to 100. With 4000 points the model has enough training data to compensate any gaps in random sampling. For cases with little computational resources grid sampling is a solution, because it ensures similar distribution of points with less data to process.

7.1.5 Generalization

The last research sub-question states *How does the model generalize to another location?*. generalization in this context means that the model is applicable to other data. The experiments show that only the local reference generalizes to point clouds at other locations. However, even the model with local reference shows a decrease in performance on the data of another location. This decrease can be explained by a difference in time of acquisition. The conclusion is that the model with local reference and intensity does generalize to other locations and differences in summer and winter.

7.1.6 Summary

As stated, the main research question is *To what extent PointNet suitable for classification of raw highway point clouds?* The methodology presented in this research is successful and achieves 50% point-wise classification accuracy and about 60% object detection accuracy. These results are lower than the current state-of-the-art of 56% point-wise classification accuracy by Wang et al. [2018]. However, the goal of this research was not to achieve state-of-the-art performance (see Section 1.4). This research is a successful exploration of PointNet on outdoor highway scene. In general can be concluded that the methodology in this research is suitable for classification of a raw point cloud of a highway scene. However, considering recent developments (Chapter 2) of GCNN for point clouds it is important to look beyond PointNet for a classifier of raw point clouds.

7.2 RECOMMENDATIONS

7.2.1 Introduction

The conclusion from the previous chapter states that this methodology is suitable for classification of raw point clouds of highway environments. However, the results show many inaccuracies. Besides the current methodology many technologies surfaced during this research that could improve the methodology and inspire future research.

7.2.2 Subdivision of classes

The objects in the data set used for this research are annotated with several classes. It appears however, that the objects within a class are not identical. For example, different types of road signs in the class *road sign* or hanging traffic lights and traffic lights on a pole in the class *traffic light*. To improve classification accuracy these classes can be subdivided.

Subdivision of the current classes increases the total amount of classes. More classes generally decrease DL performance, simply because the classification problem becomes more complex. However, alternative hierarchical class structures exist. In hierarchical class structures classes are like a decision tree. This limits the number of top-level classes.

7.2.3 Incremental space partitioning

The laser scanning point clouds vary highly in density. However, a grid does not account for density variations. The grid cells are the same size for the entire cover area. This results in grid cells with varying amounts of points that need over- or sub-sampling to create training samples of similar size.

Alternative space partitioning methods that do adapt to point density are for example *KD-tree* or *Octree*. Both methods partition space in iterations until resulting cells contain less than a certain number of points. Kuhn and Mayer [2015] shows these methods are also suitable for very large point clouds. The resulting training samples only require over-sampling and all unique points are used.

7.2.4 Additional attributes

The results in this research show the importance of spatial reference and the intensity value to represent each point (see Section 5.2.4). The MLS point clouds for this research do not contain the RGB color values for each point. However, often MLS point clouds are combined with panoramic photographs to add RGB color data. The expectation is that RGB color values will contribute to classification performance similar to the intensity value.

7.2.5 Ground filtering

To create more accurate training data the methodology in this research uses a ground filtering method to ensure all ground points are assigned the background class label. The ground filtering method assumes all ground points are within a fixed range along the Z-axis. This assumption is often not true, which causes inaccurate ground filtering. The inaccuracies in ground filtering are the main cause for inaccuracy of training data in this research.

Meng et al. [2010] list many alternatives for ground filtering. Best methods are based on iterative point-wise filtering using *clustering*, *morphological filters*. This type of algorithms requires more computation, but is also more robust.

7.2.6 Augmentation

This research introduces the concept of multi-sampling to increase the amount of training data. Multi-sampling is similar to *augmentation* because both methods aim to increase the amount of training data. However, there is an important difference. Multi-sampling creates multiple training samples per sample that contain measured data. Augmentation creates more training samples by manipulating measured data. Manipulation can be small rotation, translation, scaling or intensity changes. Suitable manipulation methods depend on the application. For example, in the case

of classification of road side objects scaling should be avoided as every object type has standard dimensions. For further research augmentation can be implemented to generate even more training data and is not limited by the amount of available unique points.

7.2.7 Deep learning hyper-parameters

The training of the model requires specification of several so-called hyper-parameters. The model in this research is used as-is. This includes most of the hyper-parameters. Generally the performance of a DL could increase by optimizing the hyper-parameters [Probst et al., 2018].

7.2.8 Alternative deep learning models

The related work (Chapter 2) elaborates on developments inspired by PointNet. PointNet combines all point features into a global feature for the entire sample. The global feature is then concatenated to every point feature. This research and recent other work critique the global feature. The combination of all point features does not incorporate local geometrical properties of a point's direct neighbourhood. A PointNet successor, PointNet++ Qi et al. [2017], and many other models introduce new ways to aggregate multiple scale neighbourhoods to capture local geometric properties. Wang et al. [2018] introduces Graph Convolutional Neural Network (GCNN) with dynamic definitions of a point's neighbourhood.

The methodology of this research could also work with a different classification model. Future work could involve the implementation of a GCNN. Wang et al. [2018] proves this network achieves higher MIOU accuracy. The assumption is that this model will also be able to differentiate between different types of objects more accurately. This can be useful for classification of specific types of road signs.

7.2.9 Semi-supervised learning

The methodology in this research is build around a model for supervised point-wise classification. This involves the creation of training data with a label for every individual point. In their work on GCNN Kipf and Welling [2016] and Wang et al. [2018] propose the alternative of *semi-supervised* classification. In semi-supervised classification only 5-20% of the points in the training data set are assigned a class label. The semi-supervised method would solve two important downsides of the methodology in this research. The first is a reduction in computation time for the spatial intersection between object locations and the point cloud. If only a small number of points around the object location are to be considered for spatial intersection. The second improvement is in training data accuracy. Currently many points are incorrect classified for the training data set, which is caused by the buffer intersection. With a smaller buffer less points are labeled, but also less errors are introduced. However, it will require a new method to actively label background class points, instead of the current method where all remaining points are background.

7.2.10 Clustering

After application of the point-wise classification model the points are clustered. The *connected components* algorithm connects points that have the same predicted label. However, this requires the construction of a graph representation. In this research the edges in the graph are based on a fixed distance (0,2 meters) between points. There may be points that do not have connections to others at all, these are so called isolates. For simplicity isolates are discarded. This can be solved by either querying with a dynamic distance or for a fixed *k Nearest Neighbours*.

The algorithm used for *connected components* requires the graph to be in memory. Because the entire test data set is too large to keep in memory the graph is constructed per sample. This limits the graph to be within one grid cell, for edge cases this means multiple clusters are created per object. The additional clusters can be corrected in post-processing.

7.2.11 Post-processing

After clustering of the points with predicted labels for each cluster a centroid is mapped. This mapping is the 2D CAD deliverable. There are two types of inaccuracies that occur in the process of clustering that can be corrected in post-processing. The first type is edge cases that cause multiple clusters for a single object. If an object is represented by multiple clusters this will also result in multiple mapped centroids for one object. The second type is when the model predicts different classes to the points of the same object. One of the resulting mappings should overrule the others. For future iterations of the mapping method a simple buffer operation could check for objects within a certain radius. Assuming all objects are at least a certain distance apart other points that occur within this buffer are probably incorrect and can be removed.

7.2.12 Open benchmark data sets

The data set in this research is a proprietary MLS scan. To provide results that are reproducible for others the methodology should also be applied to an open data set. Recently many high quality MLS and TLS bench mark data sets have become available. The best options currently are *Paris-Lille-3D* [Roynard et al., 2017], *IQmulus* [Brédif, M. and Vallet, B. and Serna, A. and Marcotegui, B. and Paparoditis, 2013], *Paris-Rue-Madame* [Serna et al., 2014],*Semantic3D* [Hackel et al., 2014].

BIBLIOGRAPHY

Brédif, M. and Vallet, B. and Serna, A. and Marcotegui, B. and Paparoditis, N. (2013). Terramobilita / Iqmulus Urban Point Cloud Analysis Benchmark. (id):2–7.

Engelmann, F., Kontogianni, T., Hermans, A., and Leibe, B. (2017). Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds. *Iccv*.

Friedl, M. and Brodley, C. (1997). Decision tree classification of land cover from remotely sensed data. *Remote Sensing of Environment*, 61(3):399–409.

Grilli, E., Menna, F., and Remondino, F. (2017). a Review of Point Clouds Segmentation and Classification Algorithms. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W3(March):339–344.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., and Pollefeys, M. (2014). Semantic3D.Net: a New Large-Scale Point Cloud Classification Benchmark.

Hackel, T., Wegner, J. D., and Schindler, K. (2016). Fast Semantic Segmentation of 3D Point Clouds With Strongly Varying Density. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:177–184.

Kipf, T. N. and Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. pages 1–14.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9.

Kuhn, A. and Mayer, H. (2015). Incremental Division of Very Large Point Clouds for Scalable 3D Surface Reconstruction. *Proceedings of the IEEE International Conference on Computer Vision*, 2015-Febru:157–165.

Landrieu, L. and Simonovsky, M. (2017). Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs.

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

Morton, G. (1966). A computer oriented geodetic data base, and a new technique in file sequencing. Technical report, International Business Machines, Ottowa.

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature Visualization. *Distill*, 2(11).

Probst, P., Bischl, B., and Boulesteix, A.-L. (2018). Tunability: Importance of Hyperparameters of Machine Learning Algorithms. pages 1–27.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.

Rasshofer, R. H., Spies, M., and Spies, H. (2011). Influences of weather phenomena on automotive laser radar systems. *Advances in Radio Science*, 9:49–60.

Ravanbakhsh, S., Schneider, J., and Poczos, B. (2016). Deep Learning with Sets and Point Clouds. pages 1–12.

Roynard, X., Deschaud, J.-E., and Goulette, F. (2017). Paris-Lille-3D: a large and high-quality ground truth urban point cloud dataset for automatic segmentation and classification. pages 1–20.

Serna, A., Marcotegui, B., Goulette, F., and Deschaud, J.-E. (2014). Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. *4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014*.

Simonovsky, M. and Komodakis, N. (2017). Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs.

Vosselman, G. (2013). Point cloud segmentation for urban scene classification. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 40(7W2):257–262.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2018). Dynamic Graph CNN for Learning on Point Clouds.

Weinmann, M., Jutzi, B., Hinz, S., and Mallet, C. (2015a). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105.

Weinmann, M., Schmidt, A., Mallet, C., Hinz, S., Rottensteiner, F., and Jutzi, B. (2015b). Contextual Classification of Point Cloud Data By Exploiting Individual 3D Neigbourhoods. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W4:271–278.

Yang, B., Dong, Z., Liu, Y., Liang, F., and Wang, Y. (2017). Computing multiple aggregation levels and contextual features for road facilities recognition using mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 126:180–194.

Zhou, Y. and Tuzel, O. (2017). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection.

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.