FACULTY OF DIGITAL ENGINEERING
Computer Graphics Systems Group

Analysis and Visualization of Transport Infrastructure Based on Large-Scale Geospatial Mobile Mapping Data

Dissertation
in partial fulfillment for the academic degree
"doctor rerum naturalium"
(Dr. rer. nat.)
in IT Systems Engineering

Hasso Plattner Institute // Faculty of Digital Engineering // University of Potsdam

submitted by
Johannes Wolf

Supervision:
Prof. Dr. Jürgen Döllner

Potsdam,
June 29, 2021

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-53612
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-536129

Abstract

3D point clouds are a universal and discrete digital representation of three-dimensional objects and environments. For geospatial applications, 3D point clouds have become a fundamental type of raw data acquired and generated using various methods and techniques. In particular, 3D point clouds serve as raw data for creating digital twins of the built environment.

This thesis concentrates on the research and development of concepts, methods, and techniques for preprocessing, semantically enriching, analyzing, and visualizing 3D point clouds for applications around transport infrastructure. It introduces a collection of preprocessing techniques that aim to harmonize raw 3D point cloud data, such as point density reduction and scan profile detection. Metrics such as, e. g., local density, verticality, and planarity are calculated for later use. One of the key contributions tackles the problem of analyzing and deriving semantic information in 3D point clouds. Three different approaches are investigated: a geometric analysis, a machine learning approach operating on synthetically generated 2D images, and a machine learning approach operating on 3D point clouds without intermediate representation.

In the first application case, 2D image classification is applied and evaluated for mobile mapping data focusing on road networks to derive road marking vector data. The second application case investigates how 3D point clouds can be merged with ground-penetrating radar data for a combined visualization and to automatically identify atypical areas in the data. For example, the approach detects pavement regions with developing potholes. The third application case explores the combination of a 3D environment based on 3D point clouds with panoramic imagery to improve visual representation and the detection of 3D objects such as traffic signs.

The presented methods were implemented and tested based on software frameworks for 3D point clouds and 3D visualization. In particular, modules for metric computation, classification procedures, and visualization techniques were integrated into a modular pipeline-based C++ research framework for geospatial data processing, extended by Python machine learning scripts. All visualization and analysis techniques scale to large real-world datasets such as road networks of entire cities or railroad networks.

The thesis shows that some use cases allow taking advantage of established image vision methods to analyze images rendered from mobile mapping data efficiently. The two presented semantic classification methods working directly on 3D point clouds are use case independent and show similar overall accuracy when compared to each other. While the geometry-based method requires less computation time, the machine learning-based method supports arbitrary semantic classes but requires training the network with ground truth data. Both methods can be used in combination to gradually build this ground truth with manual corrections via a respective annotation tool.

This thesis contributes results for IT system engineering of applications, systems, and services that require spatial digital twins of transport infrastructure such as road networks and railroad networks based on 3D point clouds as raw data. It demonstrates the feasibility of fully automated data flows that map captured 3D point clouds to semantically classified models. This provides a key component for seamlessly integrated spatial digital twins in IT solutions that require up-to-date, object-based, and semantically enriched information about the built environment.

Zusammenfassung

3D-Punktwolken sind eine universelle und diskrete digitale Darstellung von dreidimensionalen Objekten und Umgebungen. Für raumbezogene Anwendungen sind 3D-Punktwolken zu einer grundlegenden Form von Rohdaten geworden, die mit verschiedenen Methoden und Techniken erfasst und erzeugt werden. Insbesondere dienen 3D-Punktwolken als Rohdaten für die Erstellung digitaler Zwillinge der bebauten Umwelt.

Diese Arbeit konzentriert sich auf die Erforschung und Entwicklung von Konzepten, Methoden und Techniken zur Vorverarbeitung, semantischen Anreicherung, Analyse und Visualisierung von 3D-Punktwolken für Anwendungen im Bereich der Verkehrsinfrastruktur. Es wird eine Sammlung von Vorverarbeitungstechniken vorgestellt, die auf die Harmonisierung von 3D-Punktwolken-Rohdaten abzielen, so z. B. die Reduzierung der Punktdichte und die Erkennung von Scanprofilen. Metriken wie bspw. die lokale Dichte, Vertikalität und Planarität werden zur späteren Verwendung berechnet. Einer der Hauptbeiträge befasst sich mit dem Problem der Analyse und Ableitung semantischer Informationen in 3D-Punktwolken. Es werden drei verschiedene Ansätze untersucht: Eine geometrische Analyse sowie zwei maschinelle Lernansätze, die auf synthetisch erzeugten 2D-Bildern, bzw. auf 3D-Punktwolken ohne Zwischenrepräsentation arbeiten.

Im ersten Anwendungsfall wird die 2D-Bildklassifikation für Mobile-Mapping-Daten mit Fokus auf Straßennetze angewendet und evaluiert, um Vektordaten für Straßenmarkierungen abzuleiten. Im zweiten Anwendungsfall wird untersucht, wie 3D-Punktwolken mit Bodenradardaten für eine kombinierte Visualisierung und automatische Identifikation atypischer Bereiche in den Daten zusammengeführt werden können. Der Ansatz erkennt zum Beispiel Fahrbahnbereiche mit entstehenden Schlaglöchern. Der dritte Anwendungsfall untersucht die Kombination einer 3D-Umgebung auf Basis von 3D-Punktwolken mit Panoramabildern, um die visuelle Darstellung und die Erkennung von 3D-Objekten wie Verkehrszeichen zu verbessern.

Die vorgestellten Methoden wurden auf Basis von Software-Frameworks für 3D-Punktwolken und 3D-Visualisierung implementiert und getestet. Insbesondere wurden Module für Metrikberechnungen, Klassifikationsverfahren und Visualisierungstechniken in ein modulares, pipelinebasiertes C++-Forschungsframework für die Geodatenverarbeitung integriert, das durch Python-Skripte für maschinelles Lernen erweitert wurde. Alle Visualisierungs- und Analysetechniken skalieren auf große reale Datensätze wie Straßennetze ganzer Städte oder Eisenbahnnetze.

Die Arbeit zeigt, dass es in einigen Anwendungsfällen möglich ist, die Vorteile etablierter Bildverarbeitungsmethoden zu nutzen, um aus Mobile-Mapping-Daten gerenderte Bilder effizient zu analysieren. Die beiden vorgestellten semantischen Klassifikationsverfahren, die direkt auf 3D-Punktwolken arbeiten, sind anwendungsfallunabhängig und zeigen im Vergleich zueinander eine ähnliche Gesamtgenauigkeit. Während die geometriebasierte Methode weniger Rechenzeit benötigt, unterstützt die auf maschinellem Lernen basierende Methode beliebige semantische Klassen, erfordert aber das Trainieren des Netzwerks mit Ground-Truth-Daten. Beide Methoden können in Kombination verwendet werden, um diese Ground Truth mit manuellen Korrekturen über ein entsprechendes Annotationstool schrittweise aufzubauen.

Diese Arbeit liefert Ergebnisse für das IT-System-Engineering von Anwendungen, Systemen und Diensten, die räumliche digitale Zwillinge von Verkehrsinfrastruktur wie Straßen- und Schienennetzen auf der Basis von 3D-Punktwolken als Rohdaten benötigen. Sie demonstriert die Machbarkeit von vollautomatisierten Datenflüssen, die erfasste 3D-Punktwolken auf semantisch klassifizierte Modelle abbilden. Dies stellt eine Schlüsselkomponente für nahtlos integrierte räumliche digitale Zwillinge in IT-Lösungen dar, die aktuelle, objektbasierte und semantisch angereicherte Informationen über die bebaute Umwelt benötigen.

Contents

Abstract ii

Zusammenfassung iii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Structure of this Thesis . . . 4
- 1.3 Fundamentals . . . 5
- 1.4 Data Sources . . . 8

2 Preparing Raw 3D Point Cloud Data 11
- 2.1 Related Work . . . 11
- 2.2 Harmonization . . . 14
  - 2.2.1 Data Formats . . . 14
  - 2.2.2 Coordinate Systems and Shift . . . 15
  - 2.2.3 Outliers . . . 16
  - 2.2.4 Point Density Reduction . . . 17
  - 2.2.5 Scan Profile Detection . . . 19
- 2.3 Normal Vectors and Segmentation . . . 20
  - 2.3.1 Normal Vectors . . . 20
  - 2.3.2 Segmentation . . . 21
- 2.4 Metrics Computation . . . 22
  - 2.4.1 Local Point Density . . . 22
  - 2.4.2 Verticality . . . 24
  - 2.4.3 Normal Vector Diversity . . . 25
  - 2.4.4 Horizontal Planarity . . . 26
  - 2.4.5 Post-Likeness . . . 27
  - 2.4.6 Segment Density . . . 28
- 2.5 Manual Creation of Training Data . . . 28

3 Semantic Classification of Infrastructure Elements in 3D Point Clouds 33
- 3.1 Related Work . . . 34
- 3.2 Geometric Approaches . . . 37
- 3.3 Machine Learning on Rendered Images . . . 39
  - 3.3.1 Ground Image Analysis . . . 40

3.3.2 Scan Profile Analysis . . . 45
3.4 Machine Learning on Spatial Data . . . 49

4 Applications of Semantically Classified Mobile Mapping Data 55
4.1 Data Export . . . 55
4.1.1 Vector Data . . . 56
4.1.2 Graph Data . . . 58
4.1.3 Statistics Computation and Further Analyses . . . 58
4.2 Data Verification . . . 61

5 Ground-Penetrating Radar Data 63
5.1 Related Work . . . 66
5.2 Visualization Concepts . . . 67
5.2.1 3D Scene View . . . 67
5.2.2 2D User Interface . . . 68
5.3 Analysis Techniques . . . 68
5.4 Usability and Performance . . . 71

6 Panoramic Images 73
6.1 Related Work . . . 75
6.2 Visualization Concepts . . . 75
6.3 Analysis Techniques . . . 78

7 Software Architecture and Implementation 81
7.1 Overview . . . 82
7.2 Basic System Components . . . 82
7.2.1 Core Tools . . . 83
7.2.2 PCLib . . . 85
7.3 Python ML Implementations . . . 87
7.3.1 PCNN . . . 87
7.3.2 Traffic Sign Classification . . . 88
7.3.3 Road Marking Classification . . . 89

8 Evaluation of Application Scenarios 91
8.1 Results for Semantic Classification of 3D Point Clouds . . . 91
8.2 Road Marking Detection and Vector Data Derivation . . . 98

9 Conclusions and Future Work 101

Bibliography 105

Chapter 1

Introduction

1.1 Motivation

Our environment is constituted by natural and human-made objects and structures such as soil, rivers, trees, mountains, buildings, streets, railways, etc. The scale at which these objects and structures are digitally represented depends on the requirements of the applied analysis, modeling, simulation, or visualization operations, ranging from single objects, countries, and continents to the global scale of the Earth. "Consider biodiversity, climate change, cultural heritage, debt, energy, water, natural hazards, health, poverty, or security—spatial information at global, regional, and local scales is essential for addressing each of these challenges. They all require a better understanding of, and better decisions about, the location and interaction of things in space and time" (Kuhn, 2012).

A growing number of applications demand for mapping of selected features of the environment to digital representations applying high-resolution acquisition technology. For example, navigation systems require up-to-date map data, administrations demand for virtual 3D city models, and railway companies rely on precise infrastructure maps to operate the railway system (Vosselman, 2003; Wirth, 2008). For cities in flood-prone regions, digital flood simulations are conducted based on 3D models of the environment to analyze potential risks and manage emergency plans (Kia et al., 2012). In general, for all these applications, geospatial and georeferenced data and models of the real world must be generated, collected, prepared, integrated, and processed to be used for their corresponding purposes.

A large number of **application domains** deal with specialized or dedicated geospatial models of the real-world environment. For construction projects, property boundaries, terrain shape, and building measures must be known or defined precisely; the position of trees and power or water lines is just as relevant as roof orientations for assessing solar panel suitability (Babahajiani et al., 2017; Nadia et al., 2018). Many stakeholders pay attention to how construction projects fit into the surroundings. In large cities, details such as air currents between high-rise buildings are of interest during construction planning, and they can be simulated in advance (Koss, 2006). In disaster management, highly up-to-date data on access roads, water levels, fallen trees, and slope slides are of the utmost importance (M. Yu et al., 2018). For example, after storm damage, insurance companies must verify what the condition was before and after the storm to use this

information to calculate compensation. Railroad tracks must be kept free of vegetation to ensure the track environment is always clear and the required clearance can be maintained. For bulky and oversized transports, it must be determined in advance how routes can be used, the exact clearance height of a bridge, and the full turning circle at an intersection in the case of a full road closure.

Regardless of the capturing technology and the scale at which geospatial data is handled, IT systems are confronted with vast amounts of data that cannot be processed manually. For most application domains, there is a steadily growing need to automatically capture, collect, and process geospatial data, build corresponding digital models, and maintain data and models in increasingly shorter time intervals.

At the same time, there is enormous progress in **sensor technology**. The variety of sensors, their cost efficiency, availability, and compatibility is increasing. For that reason, future geospatial models, systems, and applications will be based on heterogeneous geospatial data sources that reflect various aspects and dimensions of the environment (Dold and Groopman, 2017). "Massive supplies of sensor data, obtained remotely and *in situ*, cover now large parts of the world at multiple granularities. Increasingly, they are becoming available in open access form" (Kuhn, 2012). The city of Essen, for example, uses its own vehicle for mobile mapping of streets, recording data with different sensors; they are capturing data on a yearly basis and on demand for construction sites (Stadt Essen, 2017). Self-driving cars continuously capture their current surroundings to enable safe participation in road traffic. Potentially, these data could also be used to update future geospatial models. Zourlidou and Sester (2019) give an overview of how crowdsourced GPS data can be used to identify traffic regulators, e. g., traffic lights and stop signs.

Figure 1.1: *Visualization of an airborne 3D point cloud depicting from left to right: Meadows around a river, agricultural land, train tracks, residential area, sports areas, allotment gardens, a highway, and forest area.*

3D point clouds are a universal and discrete digital representation of three-dimensional objects and environments; in particular, they have been established themselves as a key data format for the large-scale acquisition of real-world environments (Figure 1.1). For geospatial applications, 3D point clouds have become a fundamental type of raw data (Q. Wang and Kim, 2019) acquired and generated using various methods and techniques. They represent a specific captured environment by an unstructured set of measured 3D points in three-dimensional space; additional attributes such as RGB colors can be assigned to individual points (Richter, Behrens, et al., 2013). 3D point clouds provide a discrete spatial representation without any abstraction or semantics. All single measurements are contained individually and are not approximated or simplified.

Generally, capturing 3D point clouds results in large amounts of data—automated and efficient processing approaches are therefore required.

To capture and reflect real-world environments, 3D point clouds can be complemented by almost all kinds of **geospatial and image data** such as vector data describing road boundaries or property areas or 3D CAD models of buildings. In many cases, 3D point clouds can be used as a data basis to derive abstracted or generalized geospatial information (Vosselman, 2003; D. Chen et al., 2017). Depending on the capturing technique, additional panoramic images are required for coloring a 3D point cloud with real-world color information. Meshes can be derived as a surface approximation from 3D point clouds and have the advantage of a surface for texturing compared to 3D point clouds while accepting a loss in accuracy.

Frequently, **generalized data** cannot be effectively or reliably used because they do not correspond to reality in detail. For example, a cuboid does not show the details of a building's facade, a tree model lacks information about the exact position of individual branches, road markings may have worn over time and lost their generalized form, and traffic signs may be soiled or pasted over and thus no longer be legible. As a representation, 3D point clouds are close to reality and, unlike abstract models, can provide precise, detailed information about the as-is state.

For 3D point clouds, one of the **core challenges** arises from their volume and simplicity because 3D point clouds do not provide structure, hierarchy, semantics, or object-based information (Discher et al., 2018). To extract added value from 3D point clouds, segmentation, classification, and semantics enrichment play an essential role—they are the prerequisites for any downstream processing tasks (Hackel, Savinov, et al., 2017). In this way, questions such as "How many trees are in a given area?", "Where have trees been planted or felled since the last survey?", "Where does vegetation protrude into the light room?", "Where exactly are which road signs?", "Which road markings have worn off?", "Where are utility hole covers?" can be answered. In many places, preprocessing is still conducted by semi-automated analysis procedures, which generally require a considerable amount of time and personnel. This approach of analysis does not scale for vast amounts of data and permanent recording. Automatic approaches for classification and filtering are therefore inevitably required (Xie et al., 2020).

Digital twins denote virtual counterparts of physical objects. They are used increasingly frequently for decision making, optimization, and maintenance of captured objects (Negri et al., 2017). Digital twins represent not only the objects' shapes but can also hold metadata such as information about their internal states. For that reason, digitalization demands for a kind of "real-time" digital twin of environments. 3D point clouds serve as raw data for creating such digital twins of the built environment. "There is virtually no limitation for the specific types of 3D objects, structures, or phenomena that can be identified and extracted by ML-based 3D point cloud interpretation" (Döllner, 2019). Semantic annotation of the 3D point cloud data, for example by methods described in this thesis, is an essential prerequisite for creating a meaningful representation.

1.2 Structure of this Thesis

This thesis concentrates on the research and development of concepts, methods, and techniques for preprocessing, semantically enriching, analyzing, and visualizing 3D point clouds for applications around transport infrastructure, typically captured by mobile mapping. The remainder gives an overview of the topics covered in the thesis and explains important terms and data sources.

Chapter 2 describes required preparation steps for 3D point cloud data. A typical step is converting the 3D point clouds into a single reference coordinate system, e. g., converting latitude/longitude values into a local Cartesian coordinate system. In addition, surface normal vectors are computed as well as specific metrics such as density values, verticality, planarity, normal vector diversity, and post-likeness. The chapter explains which of these metrics are required for a semantic classification and which information they can contribute. Furthermore, the segmentation of a 3D point cloud into groups of points forming objects' surfaces is presented. The chapter further describes how training data is created for machine learning-based approaches, both from existing ground truth data and manually using an annotation tool.

Chapter 3 forms the core topic of this thesis and explains how individual approaches for semantic classification have been implemented. Classification uses geometric approaches based on the precalculated metrics, machine learning via previously trained Convolutional Neural Networks (CNNs) on images and 3D data, and hybrid approaches combining those mentioned before. Semantic classes analyzed by the different approaches are the base classes *ground*, *building*, *vehicle*, and *vegetation*, which can be extended and divided into various subclasses such as, e. g., *road*, *rail*, *pedestrian*, *traffic sign*, and *road marking* depending on the respective technique. The semantic classification forms the basis for all further data processing of 3D point clouds.

In **Chapter 4**, applications and required export steps are presented in detail. Lines and polygons are derived for usage in GIS tools, individual objects such as specific road markings are identified, and statistics about the datasets are computed, e. g., the number of trees in a particular area or an aggregated value representing a road's or railroad track's "health". The chapter also presents a concept for the automated verification of existing data by the example of a traffic sign cadastre.

Data acquisition using mobile mapping systems allows for capturing additional datasets collected in parallel to 3D point clouds. Measurement vehicles often combine various sensors to scan the surrounding environment. The following two chapters highlight the use of additional data sources and how they can be combined with 3D point clouds. Possibilities for using Ground-Penetrating Radar (GPR) data are discussed in **Chapter 5**. Below-ground information is used to detect potholes or cables, and a combined visualization with 3D point clouds for individual exploration is presented. **Chapter 6** explains the usage of panoramic images to gain color information, e. g., for the automated identification of traffic signs as well as visualization techniques for the combined exploration together with 3D point cloud data.

The Point Cloud Research framework, which was used to implement the approaches

described in this thesis, and its dependencies are presented in **Chapter 7**. It gives an overview of the software architecture, including external libraries, and presents the three main applications used for analyzing, visualizing, and modifying 3D point cloud data. It further describes the Python scripts implemented for machine learning-based geodata analysis.

Chapter 8 summarizes evaluations of the results' quality and processing speeds of individual processing steps. Firstly, it presents a comparison between a geometry-based and a machine learning-based semantic classification. Secondly, it investigates the quality of automatic road marking vector data derivation from 3D point clouds.

Finally, **Chapter 9** draws conclusions and presents ideas for future work in this area.

1.3 Fundamentals

This section explains the terms and techniques that are the main concepts for this thesis's domain.

3D Point Clouds represent geospatial information in the form of individual measurement points within three-dimensional space. They are technically stored as an unordered collection of points, each featuring three-dimensional coordinates and optionally additional attributes (Richter, Behrens, et al., 2013). The coordinates represent the points' spatial location, usually in a given → *coordinate system*. Depending on the capturing technique, 3D point clouds can store attributes that directly result from the measurement and attributes that are computed based on additional analyses (Habib et al., 2005). For example, when using → *LiDAR* for capturing the 3D point cloud information, all points have individual intensity values, representing how strong the emitted laser's reflection was. When using → *image matching* for creating a 3D point cloud, RGB color information is available for each point. 3D point clouds are the primary data basis for the analyses within this thesis.

Airborne Capturing describes the process of measuring geospatial data from aircraft. Measuring devices can be attached to aircraft and Unmanned Aerial Vehicles (UAVs) flying over a specific area to obtain the desired coverage. Regular scan flights are typically performed by municipalities or states that want to collect geospatial data about their territory (Stadt Berlin, 2020). Commonly, the data from airborne capturing is used to derive information about, e. g., terrain height, land use, building sizes, and tree locations. Because transport infrastructure analysis is usually based on ground-level data, this thesis focuses on → *mobile mapping* data.

B-Scans are a form of → *GPR* data representation. They represent the amplitudes measured by a GPR antenna in relation to the position of the measurement on a → *trajectory* (Ozkaya and Seyfi, 2018). B-Scans are often visualized with the height of measured amplitudes mapped to color intensity and the amplitude direction mapped on hue, e. g., red and blue. In this thesis, B-scans are used to automatically

detect pavement regions with developing potholes and for a combined visualization with 3D point clouds.

Convolutional Neural Networks (CNNs) are a particular form of artificial neural networks inspired by biological processes in the human brain (Goodfellow et al., 2016). Within these networks, convolutions are performed in some of the internal layers. The respective neighborhoods for individual sections of the data play a significant role during this process, making the method particularly suitable for processing image information. The result is usually the output of a probability for a specific object class based on the input data, such as what kind of object was detected in an image. The methods described in this thesis partially use CNNs for the semantic classification of geospatial data.

Coordinate Systems describe the context in which location values (coordinates) are interpreted. → *3D point clouds* are often handled in zoned coordinate systems from the UTM Reference System, also known as Military Grid Reference System. Such locally used coordinate systems have the advantage that the approximation of a rectangular Mercator projection of the Earth's surface in a small, enclosed area deviates only slightly from the actual conditions so that a Cartesian coordinate system can be used within each of these areas. x, y, and z coordinates of a 3D point cloud can here be interpreted in meters, enabling more comfortable handling and intuitive perception of the coordinate values in contrast to latitude and longitude values.

Global Navigation Satellite System (GNSS) is a satellite-based system that provides geolocation information for devices with respective receivers. Several systems are operated by different nations: GPS (USA), GLONASS (Russia), BeiDou (China), and Galileo (European Union). A GNSS serves as a data source for registering geodata, i. e., for determining the position of a → *mobile mapping* vehicle and therefore the coordinates of its measurements (R. Li, 1997).

Ground-Penetrating Radar (GPR) delivers below-ground structural information by sending radiation into the ground and capturing the returning signal via dedicated antennas (Davis and Annan, 1989). The scanning device can measure material properties several meters below ground, creating insights about the non-visible foundation of roads and pathways. The strength of the returned signal is used to conclude the condition of the pavement and underground structures such as pipes and developing potholes (Huston et al., 2000). This thesis uses GPR as an additional data source for geospatial analysis and a combined visualization with → *3D point cloud* data.

Image Matching describes the "reconstruction of 3D surface representations from large sets of overlapping imagery" (Haala and Cavegn, 2016). "Space reconstruction starts with identifying features of interest in overlapping images. Conjugate features and the exterior orientation parameters of the involved images are then used in an

intersection procedure yielding corresponding object features" (Habib et al., 2005). From this image data, → *3D point clouds* can be generated with high resolution without the need for a → *LiDAR* scanner (Wenzel et al., 2012). Because → *mobile mapping* primarily uses LiDAR for 3D point cloud capturing, this thesis does not go into image matching methods in detail.

LiDAR (Light Detection and Ranging) is a capturing technique using a laser to measure objects' distances. It "stands out as one of the most valuable sources of geospatial information, providing enormous benefits in a wide range of scientific and professional fields" (Deibe et al., 2020). Traditional LiDAR is "operating in that part of the spectrum comprising ultraviolet to near-infrared regions. It consists of a laser which emits radiation in pulse or continuous mode through a collimating system, and a second optical system which collects the radiation returned and focuses it on to a detector" (Barrett and Curtis, 1999). By measuring the time until the reflection returns to the device, the precise distance is determined. → *Mobile mapping* systems often use motorized optomechanical scanners with rotating mirrors, which are as by now the most common type of LiDAR scanners in the industry (D. Wang et al., 2020). Two or more scanners with different orientations can be used to minimize the area that is not scanned between two revolutions and avoid missing areas vertically oriented to the vehicle. In recent years, Flash LiDAR sensors were introduced for handheld consumer devices such as tablets and smartphones. "Flash LiDAR uses all solid-state components, which has the advantages of no moving parts, being resistant to vibrations, a compact size, and low price" (D. Wang et al., 2020).

Mobile Mapping techniques can be used in contrast to → *airborne capturing* to gain information on the ground level. Mobile mapping systems can be installed on moving vehicles like cars or trains, functioning as moving carrier platforms. Navigation sensors such as → *Global Navigation Satellite System* (GNSS) receivers and Inertial Navigation Systems (INS) provide precise information about the current location; mapping sensors such as → *LiDAR* are used to measure distances to the environment (R. Li, 1997). A combination of these data sources enables the creation of mobile mapping → *3D point clouds*. For example, these can be used in applications analyzing road or railroad environments, as described in Section 1.1.

Panoramic Images are taken with a dedicated camera setup capable of capturing a 360 degrees environment. Usually, the setup consists of at least five circular placed cameras for a surround-view and an optional camera pointing to the top. The individual images are transformed and merged into a panoramic image via a process called "stitching" (Szeliski and Shum, 1997). Panoramic images can be used for, e. g., coloring → *3D point clouds* or applying digital image analysis for object detection.

Trajectory is the name of the path a → *mobile mapping* scanning vehicle took during the measurement. The combination of a → *GNSS* and an Inertial Measurement

Unit (IMU) provides accurate location data (Gressin et al., 2012). The trajectory consists of a series of coordinates with timestamps that enable a reconstruction of the measuring path and the vehicle's speed at a given position. The information is used, e. g., for projecting a → *GPR* scan into a 3D point cloud.

Transport Infrastructure within this thesis describes infrastructure used for transporting people and goods, primarily focusing on roads and railroads. In these areas, individual assets like traffic signs, road markings, railroad signals, track switches, and many others are of specific interest for planning and maintenance purposes. Data for these use cases are mostly captured via → *mobile mapping*.

Voxel Grids are three-dimensional data containers consisting of usually cubic cells called "voxels" stacked together to fill a particular space. They can be used to analyze neighborhoods in → *3D point clouds* efficiently. After all points have been placed into the particular voxel surrounding their position, all points within the same voxel can be handled as a local neighborhood with the voxel's dimensions. Voxel grids are used, e. g., for the calculation of several metrics described in Section 2.4.

1.4 Data Sources

This section lists the datasets used for the work described in this thesis. The included data formats, the dataset's size, existing ground truth information, and unique characteristics are presented in Tables 1.1 through 1.4.

Name	**Essen**
Type	Mobile mapping from car
Description	Individual streets distributed in the city area, recorded explicitly for individual analysis before/after construction work
Provided by	Amt für Geoinformation, Vermessung und Kataster; City of Essen
Data	• LiDAR – LAZ format – Separate left and right scans – Intensity values – 30.3 GB (218 scans) • Panoramic images – Merged 360 degrees images from 5 sideways cameras and 1 upward camera – 8 000 × 4 000 px resolution – 69.1 GB (13 891 images) • GPR – 8.8 GB (318 scans) • GPS trajectory – 677 GB (94 files)

Table 1.1: *Essen Dataset.*

Name	**Hamburg**
Type	Mobile mapping from car
Description	Streets in the city center; recorded for the acquisition of general road environment data, among other things, for the derivation of a base map for autonomous cars
Provided by	AllTerra Deutschland GmbH
Data	• LiDAR – LAZ format – Left and right scans merged – Intensity values – 126 GB (809 scans) • Panoramic images – Merged 360 degrees images from 5 sideways cameras and 1 upward camera – 8 000 × 4 000 px resolution – 75.6 GB (15 748 images) • Ground Truth – Manually created shape files with location of road markings, traffic signs, traffic lights, trees, etc. – 173 MB (73 files) – Road markings have only length, no width; sign types are only categorized in circular, rectangular, etc.; locations are not necessarily precise; some road markings are recorded but not existent in the LiDAR data

Table 1.2: *Hamburg Dataset.*

Name	**Potsdam**
Type	Mobile mapping from car
Description	Complete city area
Provided by	City of Potsdam
Data	• LiDAR – LAZ format – Separate left and right scans; occlusions by other scanner – Intensity values – RGB values mapped from panoramic images – 335 GB (6004 scans)

Table 1.3: *Potsdam Dataset.*

Name	**Railroad**
Type	Mobile mapping from train
Description	Tunnels and overground sections recorded for structure checks, searching objects in the track environment, and deriving rail networks with tracks and switches
Provided by	Nextrail GmbH
Data	• LiDAR – LAS/LAZ format – Scan profiles – Intensity values – 111.4 GB (303 scans) • Video – RGB values – Precise capturing position per frame – 67.4 GB (2 files) – 1 920 × 1 080 px resolution – 10.7 hours duration

Table 1.4: *Railroad Dataset.*

Chapter 2

Preparing Raw 3D Point Cloud Data

Sensors provide raw data for 3D point clouds; similarly, photogrammetric approaches compute that kind of raw data. However, raw data needs to be preprocessed for several reasons, such as:

- Unification of different coordinate systems or reference systems for different raw data sets
- Harmonization of the average spatial point density
- Removal of outlier points and noisy points
- Removal of incorrectly or incompletely encoded data

This chapter first presents related work in the preprocessing of 3D point clouds in Section 2.1 and then describes the harmonization of coordinates and point densities as well as outlier recognition in Section 2.2. Section 2.3 describes procedures for computing normal vectors and segmenting 3D point clouds into point groups belonging together. Section 2.4 describes the calculation of various metrics that are required for later analysis steps, and Section 2.5 concludes with approaches for generating training data for machine learning.

2.1 Related Work

Outlier detection is a fundamental problem when working with 3D point clouds and has been investigated with different approaches for years. Han et al. (2017) state "the raw point cloud is often noisy and contains outliers. Therefore, it is crucial to remove the noise and outliers from the point cloud while preserving the features, in particular, its fine details". The basis for most approaches is established outlier detection algorithms from the field of general statistics.

K. Zhang et al. (2009) define a metric called "Local Distance-based Outlier Factor" (LDOF), stating the outlier-ness of a point in a scattered two-dimensional dataset. They use a point's relative location to its neighbors to determine the degree to which the object deviates from its neighborhood. Therefore, they define the LDOF being the average distance of a point to its k-nearest neighbors divided by the average distance between all pairs of points in this k-nearest neighborhood.

Nurunnabi et al. (2015) propose a "Maximum Consistency with Minimum Distance" (MCMD) approach, stating that outliers are usually perceived as points not fitting the majority of the surrounding data and, therefore, points not forming consistent homogeneous surfaces are those to be removed.

LDOF and MCMD are combined in the method proposed by Dey et al. (2019). They argue that each method only considers parts of the problem, and a combination of both leads to much better overall results.

With increasing computing performance, 3D point cloud **density reduction**, also known as **3D point cloud thinning**, becomes less critical but can nevertheless be used to speed up processing times and decrease visual clutter. Many different approaches for general uniform data thinning can also be applied to 3D point clouds. Of greater interest are methods tailored to the overall structures in a 3D scan and, therefore, preserving them better.

Dyn et al. (2008) describe 3D point cloud thinning as a "greedy point removal scheme" in which they compute the significance of a point for its local neighborhood and, one by one, remove the least significant point. Significant to them are points with few other points nearby and those in non-uniform areas.

Tazir et al. (2016a) propose a method basing decisions on RGB color information. They place all points into a voxel grid and, in each voxel, cluster points by color using k-means clustering. Each cluster only keeps one representative point, stating that all other cluster points have a similar color in a similar position when using a "small voxel size".

Du and Zhuo (2009) describe a 3D point cloud thinning method based on curvature values. For all points in the 3D point cloud, they determine the curvature using a paraboloid fitting approach. The points are placed into a voxel grid, and all voxels are analyzed one by one. Suppose the average curvature in a voxel is lower than the complete 3D point cloud's average curvature. In that case, they only keep the point with its curvature being closest to the average in this voxel. If a voxel's average curvature is higher than the overall average, they keep all points from this voxel with a curvature higher than the overall average. Many points in regions with high curvature remain in the data, while low curvature regions are drastically thinned by this method.

Points located at sharp edges best describe the structure of scanned objects and are thus most important to preserve during data reduction. "Because of the distinct feature represented by data points located on or near the sharp edges (edge points), these points should always be retained by the simplification process" (Song and Feng, 2009). Song and Feng present a method to explicitly identify edge points and consider tangential discontinuity in addition to the curvature. To determine the importance of single points, they consider only points in the respective neighborhood, which were not before recognized as edge points.

Normal vector computation is a necessary preprocessing step for many of the following metric computations and analyses. Traditional 3D point cloud analysis ap-

proaches often used surface reconstruction to create surface triangle meshes from the 3D point cloud data (Amenta et al., 1998; Funke and Ramos, 2002). From these meshes, surface normals can trivially be determined. However, surface reconstruction is very time-consuming and thus not feasible and consequently less often used for modern large-scale datasets. Also, with high-density point information, a reconstruction of surfaces is no longer necessary for many use cases, and thus the reconstruction was broadly replaced by precise point-based analyses.

C. Wang et al. (2001) compare local plane fitting methods from which normal vectors are derived. Mitra and An Nguyen (2003) in detail present a method based on local least-squares fitting and analyze the effects of point neighborhood sizes, curvature values, and sampling density and consider noise.

The framework used in this work uses the Point Cloud Library (PCL) (Rusu and Cousins, 2011) coming with an implementation for the normal vector computation also based on least-squares plane fit.

Several papers emphasize the relevance of an analysis of individual **scan profiles** or scan lines in various applications in the mobile mapping field (X. Chen et al., 2009; Chu et al., 2019). Yan et al. (2016) explain the advantage of a scan profile-based analysis with the relatively small number of points in a single profile that can be managed easily with simple data structures. Scan profile analysis is also very time efficient for large datasets because scan profiles can be processed in parallel.

Anh Nguyen and Le (2013) published a survey about 3D point cloud **segmentation** techniques. They categorize them into edge-based, region-based, attributes-based, model-based, and graph-based methods. Vosselman (2013) presents an approach based on a 3D Hough transform focused on objects in road space, such as trees, posts, and vehicles, aiming to minimize undersegmentation. Vo et al. (2015) describe a method combining advantages of clustering and region growing. While clustering is a robust method without the need for seed points that are required for region growing, it is computationally expensive. They propose a fast but rough octree-based clustering followed by a step merging left-over points to the best-fitting clusters. Similarly, later works such as Ni et al. (2017) base their segmentation on a combination of multiple algorithms, in this case RANSAC region growing, followed by scattered points clustering and small segments merging.

Many 3D point cloud analysis steps require information about surfaces in the data, which provides the basis for many other recognition steps. Oehler et al. (2011) developed an approach for a planar segmentation. Even in sparse data from moving cars, it is possible to detect planes, as W. Wang et al. (2016) show.

In the same way that more and more machine learning-based algorithms are being investigated for the semantic classification of 3D point clouds, segmentation can also be approached with machine learning. RGCNN and VV-NET are two of the recently popular networks (Te et al., 2018; H.-Y. Meng et al., 2019). RGCNN is an approach based on graph construction and convolution, which does not require any preceding

voxelization as many other algorithms. VV-NET, on the other hand, is using voxels. However, the authors emphasize the advantage of storing a binary occupancy value per voxel and a more detailed multi-dimensional latent space.

Various **metrics** have been developed for the geometry-based analysis and semantic classification, focusing on different characteristics.

The usage of a local point density value is, e. g., discussed by Ning et al. (2018). They define sparse outliers as "erroneous measurement points with low local point density" and isolated outliers "have high local point density and are relatively separated from the scanned data". They further define the local point density as the average distance of a point to its k-nearest neighborhood and use this metric for outlier detection. Similarly, Breunig et al. (2000) use a "local reachability density", which is also based on average distances to the nearest neighbors, for outlier detection. Soilán et al. (2016) define a planarity value for a cluster of points based on the eigenvalues from a Principal Component Analysis and a post-detection by region growing, starting at positions where traffic signs have been detected by intensity value and performing several height checks.

2.2 Harmonization

3D point clouds can be acquired with a variety of different scanning systems. Depending on the technology used, such as LiDAR or image matching, and the hardware manufacturer, the raw data are available in various data formats with different characteristics.

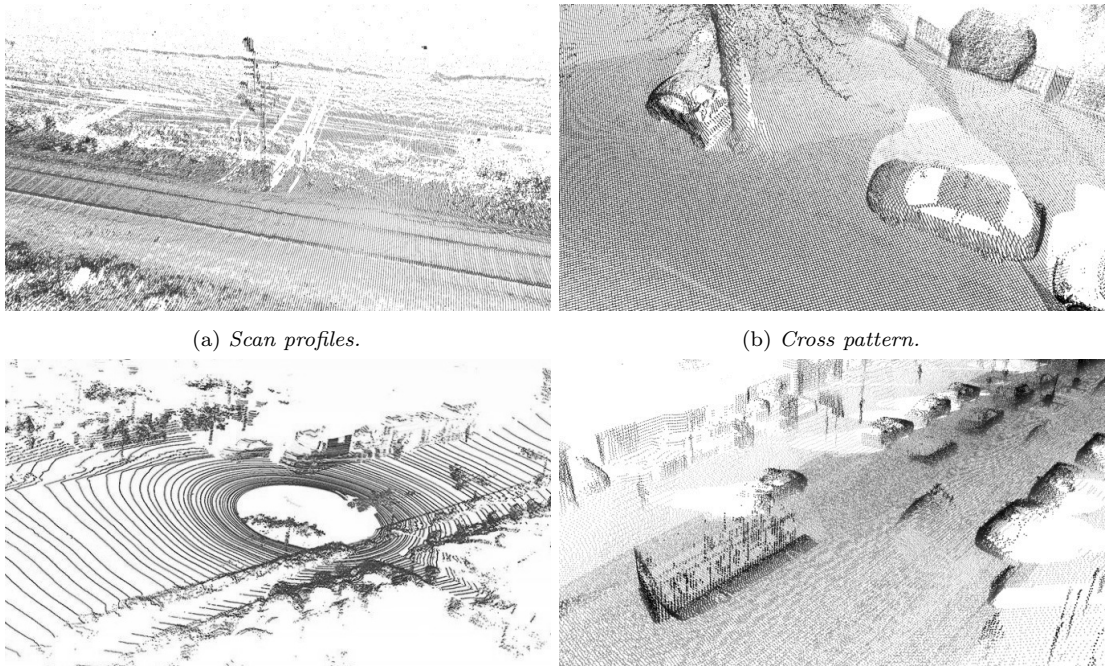
2.2.1 Data Formats

When using image matching, the 3D point clouds are calculated from image data afterward, creating a uniform point density in a regular raster. When using LiDAR, this is not the case because the measuring points are exactly created where the emitted laser beam hit an object and was reflected. Depending on the mounting orientation of the scanner and the number of scanners used, this can result in scan profiles, cross patterns, radial data, or irregular distributions, as shown in Figure 2.1.

Different data formats are used for storing 3D point cloud information. Three commonly used formats are the following:

LAS The format's name is derived from the term "laser" as it is a file format primarily developed for storing and exchanging LiDAR 3D point cloud data. LAS is a widely-used open format developed by the American Society for Photogrammetry and Remote Sensing (ASPRS, 2019). Compressed LAS files usually get the extension LAZ.

XYZ Files with this ending use a non-standardized ASCII representation of 3D point cloud data. The data is usually represented column-based, with the first three columns holding information about the x, y, and z coordinates. The following columns store additional attributes such as color or normal vectors.



(a) *Scan profiles.* (b) *Cross pattern.*

(c) *Radial pattern. Adapted from Simonite (2017).* (d) *Irregular distribution.*

Figure 2.1: *Visualization of 3D point clouds captured with different scanner setups resulting in different point distribution patterns.*

E57 This format was developed by ASTM International and is named after its Committee E57, which is responsible for the development of standards for 3D imaging systems (ASTM International, 2019).

The framework used for the implementations described in this thesis uses its own format XPC and is described in Chapter 7 in more detail. It implements readers for several typical data formats and converts them into a unique internal representation. Thus, later processing steps can assume the always same data format.

2.2.2 Coordinate Systems and Shift

Besides the data format, coordinates of individual points can be stored in various coordinate systems. These can either use latitude/longitude values or metric values in a local coordinate system. Local coordinate systems are often described by their EPSG codes, defining the valid area and its projection on the earth's surface (IOGP Geomatics Committee, 2020).

For easier processing, source data is always converted into a suitable local metric coordinate system. In this local coordinate system, distances can be read directly from the coordinates. For this purpose, each dataset is given the information in which coordinate system it is stored in an EPSG code. Data from different systems can be converted into one another and put into relation. In Germany, for example, "ETRS89 / UTM zone 32N"

and “ETRS89 / UTM zone 33N” with the EPSG codes 25832 and 25833 are widely used as reference systems.

The accuracy of decimal places decreases with larger numbers because of how typical representations of floating-point numbers in computer systems work. The coordinates' numerical values must not become too large to ensure sufficient precision for the measuring points' coordinates. A shift to be applied to all local coordinates is specified in each dataset to ensure this. Local coordinates and shift together result in the actual global coordinates. The individual numerical values of the local coordinates can thus be kept significantly smaller.

2.2.3 Outliers

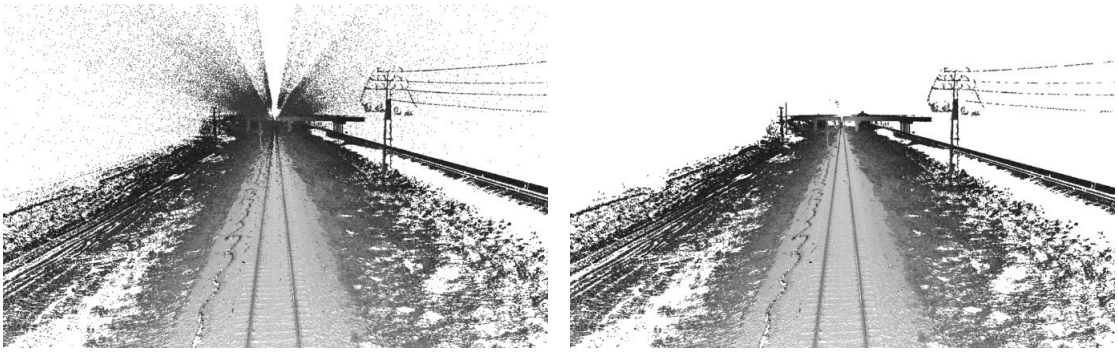
Outlier points are individual measuring points that are considered as disturbance or noise (Han et al., 2017). For example, reflections on wet surfaces, glass, or metal can lead to measuring points at positions with no actual object. As described in the review of related work, there are numerous approaches for outlier removal, each based on specific assumptions about what outliers are and how individual types of datasets should be handled.

For the datasets considered in this thesis, outliers are mostly small groups of loose points without any recognizable affiliation to other objects or points that are located at the edge of the 3D point cloud, where only sporadic measurement values are available. In addition, reflections during scanning can create unwanted points under the ground or behind facades. Outlier filtering removes points that have no or only a few other points in their immediate vicinity.

The outlier detection is not a focus of this thesis because a small number of outliers within millions of points is essentially not hindering the semantic classification. However, especially those outliers visually disturbing the scene are removed by a fast outlier removal algorithm that does not require long processing times.

Precise analysis for the detection would place a virtual sphere around each point and determine the number of neighboring points within this sphere. The exact thresholds depend on the respective dataset and the average point density. For example, in typical applications, it may be required that a sphere with a diameter of one meter around the point contains at least 10 points for this point not to be considered an outlier.

For a more efficient calculation without sphere radii and consideration of all individual points, an approximate solution based on a voxel grid can also be used. All points of the 3D point cloud are placed into a voxel grid with a cell size of, e. g., one cubic meter. The points in all cells that subsequently contain less than 10 points can be removed as outliers. This approximation solution is many times faster than the precise variant because sorting the points into the voxel grid based on their coordinates is not very computationally intensive. Afterward, not all other points have to be searched for a possible neighborhood, but only the points within a voxel are counted. It is also possible to use the voxel grid as acceleration for an exact determination. A voxel grid is filled with the points first. For each point, the sphere neighborhood analysis is performed only



(a) *Complete data with outliers.* (b) *Data after outlier points have been removed.*

Figure 2.2: *Visualization of a 3D point cloud from the Railroad dataset before and after outlier points removal.*

with points, which are possible neighbors due to their assigned cell in the voxel grid.

An additional step in outlier removal can remove points below the ground. After a ground detection has been performed, underlying points can be removed, which are usually caused by reflections on roads or rails or by the holes in utility hole covers. Figure 2.2 shows a visualization of a 3D point cloud before and after outlier points removal.

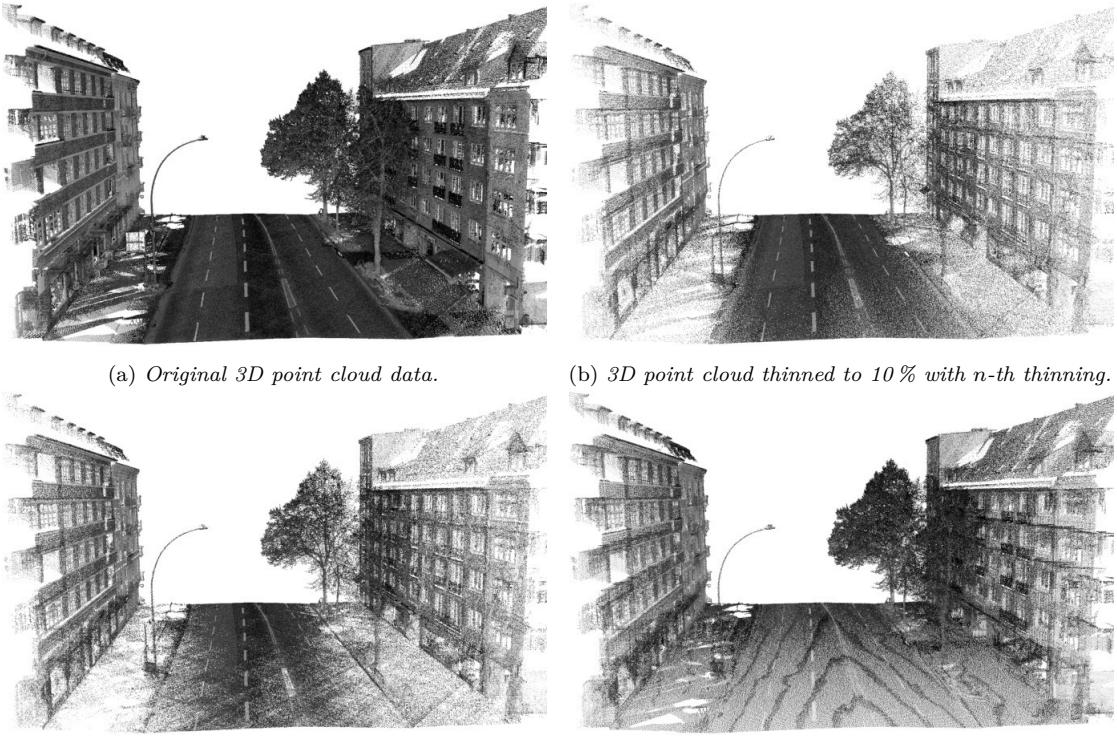
2.2.4 Point Density Reduction

In 3D point clouds captured by LiDAR, there is no regular point density as in image matching 3D point clouds. With LiDAR, the point density is significantly higher close to the scanner than with increasing distance (Boulch, Saux, et al., 2017). The measuring vehicle's speed also influences the point density since the scanners usually work at a fixed measuring speed, and the number of points captured per second thus remains constant. Depending on the application, it can be advantageous to adjust the point density within a dataset and across datasets (Seufert et al., 2020; Tazir et al., 2016b). Areas close to the scanner often have a very high point density, which is not required for further evaluation and only increases the computing effort. Besides, strongly varying point densities can have an unintended influence on other metrics' calculations and negatively impact machine learning approaches (Hackel, Wegner, et al., 2016).

Therefore, 3D point clouds can be thinned out and contain fewer points afterward. The local point density (see Section 2.4.1) depends on the point distribution characteristics, e. g., when estimating distances from the measuring vehicle based on this value.

Thinning can be performed by various approaches as shown in Figure 2.3. A trivial approach called *n-th thinning* keeps only every n-th point on iterating over the 3D point cloud. All other points are discarded, resulting in a 3D point cloud with a density of $1/n$ compared to the input data.

To better preserve the 3D point cloud structures, one way is to thin out the points semantically based on curvature values. The curvature indicates how much the



(a) *Original 3D point cloud data.*

(b) *3D point cloud thinned to 10 % with n-th thinning.*

(c) *3D point cloud thinned to 10 % with curvature thinning.*

(d) *3D point cloud thinned to 10 % with voxel grid thinning.*

Figure 2.3: *Visualization of a 3D point cloud from the Hamburg dataset and different thinning methods.*

approximate surface is curved at this point in the 3D point cloud. A plane has no curvature, and a cylindrical post has a high curvature. With this *curvature thinning* approach, flat surfaces can be thinned out more than areas of the 3D point cloud rich in structures.

Usually, the ground near the measuring vehicle was very densely captured. However, the high number of measuring points offers only limited value because the ground's condition and any markings or other objects can also be detected with fewer points. Therefore, significantly more points can be removed in this area than, for example, at traffic lights, signals, or edges of building facades, representing essential structures in the environment to be analyzed in detail.

An efficient implementation performs randomized removal considering the curvature values. Points with lower curvature have a higher probability of being removed. By setting the desired percentage of total thinning, appropriate probability thresholds can be defined to perform the thinning.

Another approach is *voxel grid thinning* in which all points are placed into a voxel grid with a small voxel size. In each voxel, only one representative point will be kept. This point can either be arbitrarily chosen or the point closest to the voxel's center. In

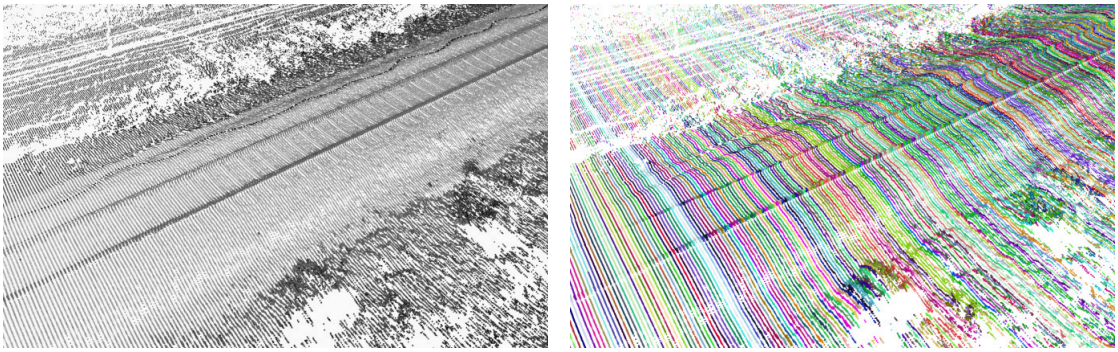
contrast to the other approaches, the thinning percentage cannot be precisely configured because the voxel size must be guessed. Also, the local density is not a meaningful metric anymore in 3D point clouds thinned by this approach.

Although curvature thinning takes more time than n-th thinning, it usually allows removing more points without degrading subsequent analyses, whose processing time is reduced by the lower number of points. Therefore, the time required to determine the curvature can be worthwhile. Voxel grid thinning is fast but has a significant impact on further analysis steps.

2.2.5 Scan Profile Detection

In some scanning setups, the 3D point cloud data is recorded in the form of individual scan profiles. Especially in the railroad environment, the LiDAR scanners are often mounted at the train's front and measure perpendicular to the train. Due to the scanner's rotation, the laser beam rotates in all directions in a circle around the train—the additional movement of the train results in a spiral scan of the environment. The paths of the spiral have a distance of a few centimeters from each other (Yan et al., 2016). Each revolution within the spiral is called a *scan profile* and represents a *cross-section* of the environment. The individual scan profiles are rarely closed all around. For example, there are only a few measuring points above the train on overhead lines or signal bridges outside of tunnels. In most cases, scan profiles contain only points near the ground and lateral objects such as vegetation or signals. Figure 2.4 shows a 3D point cloud with highlighted individual scan profiles.

In some acquisition datasets with scan profiles, the respective scan profile ID is part of the scanner's raw data. If this information is not provided, it can be calculated by various methods to use it for subsequent analyses (Chu et al., 2019). For example, the image-based classification of track environments described in Section 3.3.2 relies on individual scan profiles.



(a) *Points colored by intensity values. Lighter colors represent higher intensity.*

(b) *Points colored by scan profile ID. Each scan profile has a distinct color, all points in the same scan profile have the same color.*

Figure 2.4: *Visualization of a 3D point cloud from the Railroad dataset and individual scan profiles within.*

If available, the scanner's angle at the time of scanning can be used to determine individual scan profiles. Frequently, the measuring points are stored in the order of their acquisition. A scan profile comprises all points within a 360 degrees scanner rotation before the next rotation begins. If the angle is unavailable as an attribute, it can be approximated via the trajectory if the points are ordered by acquisition time.

If the measuring points are not ordered, the determination of the affiliation to scan profiles is more complex and time-consuming but still possible as an approximate solution. From the trajectory, the direction of travel can be determined for the local environment. The searched scan profiles are oriented perpendicular to it. Thus, points whose distance to the neighboring point in the expected direction of the profile is smaller than a defined limit value and significantly smaller than the distance to a neighboring scan profile can be assigned to the same scan profile. It is considered that the resulting scan profile must approximately result in a plane and that points cannot shear out too far laterally because otherwise, they would have to be assigned to a different scan profile.

The above-mentioned methods assign the associated scan profile ID as an attribute to each point in a 3D point cloud. Subsequent processing steps can access the information.

2.3 Normal Vectors and Segmentation

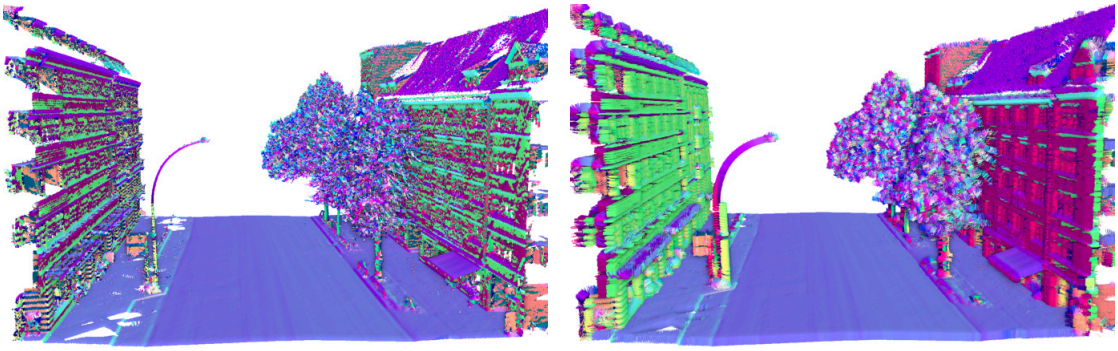
To analyze a 3D point cloud, information about its structure is required. The two most basic preprocessing steps for almost all applications are calculating the surface normal vectors and a subsequent division into segments. Both are described in detail below.

2.3.1 Normal Vectors

Mathematically, points have no orientation and, therefore, no normal vector. For a 3D point cloud, the orientation of an individual point is understood as the local neighborhood's orientation, for which an imaginary surface can be constructed. Based on that idea, the normal vectors in a 3D point cloud indicate the surface orientation of the points.

To compute the points' normal vectors, the local neighborhood is regarded for each point, and the best-fitting plane is determined. This plane's normal vector is recorded as normal vector for this point of the 3D point cloud. With the help of the normal vectors computed in this way, statements can be made about connected surfaces, a precondition for the subsequent segmentation, among other analysis steps.

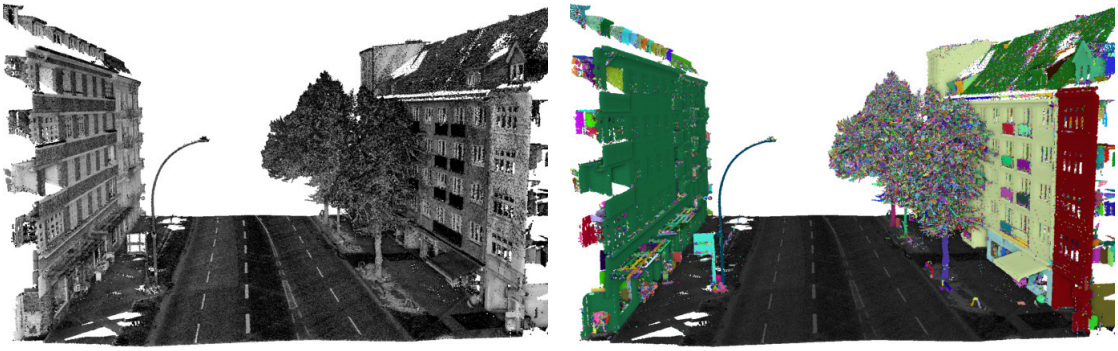
Different approaches can be used to determine the neighborhood of a point. For example, the k-nearest neighbors could be considered the neighborhood or neighboring points are searched for in a sphere radius around the point. For the implementation of the normal vector computation, the implementation from the PCL (Point Cloud Library) is used in the underlying framework of this thesis (see also Chapter 7), which was adapted for the processing of vast and dense 3D point clouds. Figure 2.5 shows visualizations of point normal vectors.



(a) *Points colored by normal vector orientation. RGB channels map to XYZ coordinates.*

(b) *Points with attached lines representing the orientation of the normal vectors.*

Figure 2.5: *Visualization of the points' normal vectors in a 3D point cloud from the Hamburg dataset.*



(a) *Points colored by intensity values. Lighter colors represent higher intensity.*

(b) *Points colored by segment ID. Each segment has a distinct color, all points in the same segment have the same color.*

Figure 2.6: *Visualization of a 3D point cloud from the Hamburg dataset and detected segments within.*

2.3.2 Segmentation

"*3D point cloud segmentation* is the process of classifying point clouds into multiple homogeneous regions" (Anh Nguyen and Le, 2013). The homogeneity can be based on different values. Often segmentation is used to group all points of an object, such as a car or traffic sign (Vosselman, 2013). Depending on the application, the desired segmentation can be larger or smaller. For example, in one case, it might be required to get one segment per car on the road, and in another, a subdivision into the hood, the roof, the tires, and other parts might be of interest. Buildings could either be segmented as a whole or broken down into individual parts such as facades, roofs, windows, and doors. Figure 2.6 shows a visualization of a segmented 3D point cloud.

The surface normal plays a significant role in the semantic segmentation of objects. In many use cases, the segmentation is performed based on surface orientation (Oehler

et al., 2011). Local point clusters with the same orientation are combined into one segment. Planes and rounded edges can be recognized as belonging together. Starting from points with low curvature, neighboring points are added to the segment if their orientation matches the previous points. A substantial change of the normal vector is usually handled as an abort criterion to interrupt segments at hard edges. Likewise, a limit value could be set that segments should not exceed a particular area or number of points if smaller segments are desired.

Alternatively, segmentation can be based on other attributes such as color or intensity. If, for example, similar intensity values are used as a basis, it is possible to detect contiguous road markings which are just as planar as the surrounding road but stand out from it by their intensity.

In the results of segmentation methods, a distinction is made between *oversegmentation* and *undersegmentation*. Oversegmentation creates too many segments, so areas belonging together fall apart into several segments harming individual objects' recognition. For example, if cars are detected by filtering segments of a specific size, individual cars cannot be detected if split into too many individual segments. Undersegmentation occurs if too few segments are created, for example, if the detection does not break off at narrow or rounded edges, although this would be desirable. For example, smaller structures on large surfaces could not be viewed individually, such as windows in a building facade or stones on the ground.

The correct degree of segmentation depends on the individual application and cannot be determined universally.

2.4 Metrics Computation

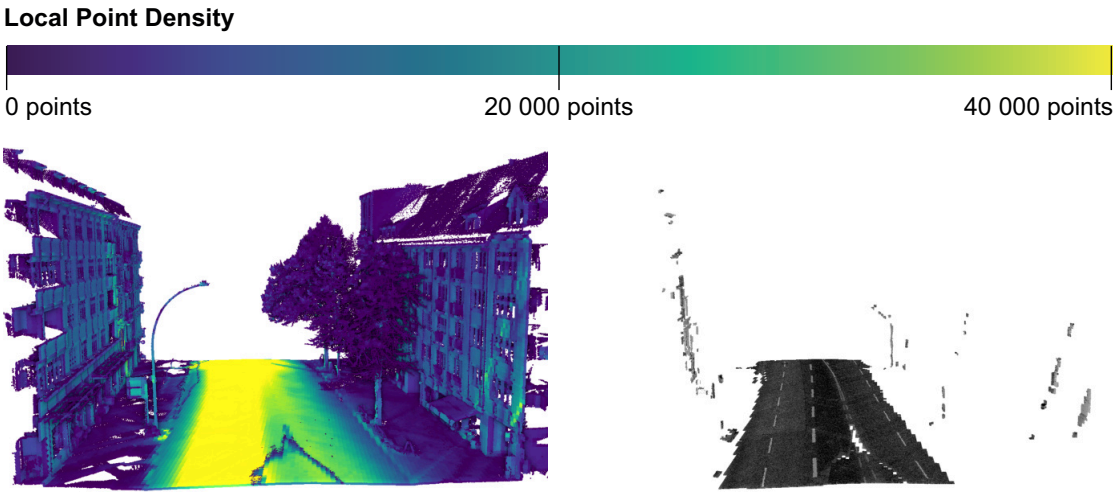
For the geometric analysis of 3D point clouds for semantic classification, many analysis results of specific characteristics, so-called metrics, are required. Metrics provide information about specific properties of points, point groups, and their environments, thus enabling statements to be made about displayed structures and objects.

Additional pre-calculated metrics can also help machine learning-based procedures to improve the classification with CNNs or accelerate their training.

The following sections describe important metrics that have been developed for the analysis of 3D point clouds and have proven to be meaningful for semantic classification. Many of the metrics require normal vectors and segment information to be already available.

2.4.1 Local Point Density

The *local point density* (*LDensity*) is especially relevant in mobile mapping 3D point clouds because here, the point density hugely varies throughout a dataset (Boulch, Saux, et al., 2017). As the distance from the scanner increases, the point density decreases significantly. Several thousand points per square meter can be available in the scanner's immediate vicinity. However, at a distance of 50 meters, for example, only less than 10



(a) *Points colored by local point density.*

(b) *Points filtered by high local point density. Only points with LDensity* $> 20\,000$ *points are shown.*

Figure 2.7: *Visualization of the local point density metric in a 3D point cloud from the Hamburg dataset.*

points are captured in the same area. Figure 2.7 shows a visualization of the local point density metric.

As described in Section 2.2.4, a uniform point density is advantageous for several processing steps.

The local density considers the point density in a small local area in contrast to the general point density, which can be specified as a dataset's total value. The local point density is calculated for each point in the 3D point cloud as an attribute. As a theoretical concept, the number of neighboring points in a sphere with a defined radius of, e. g., half a meter with the considered point in the center is counted and divided by this sphere's volume. The result is a local density value for the considered point.

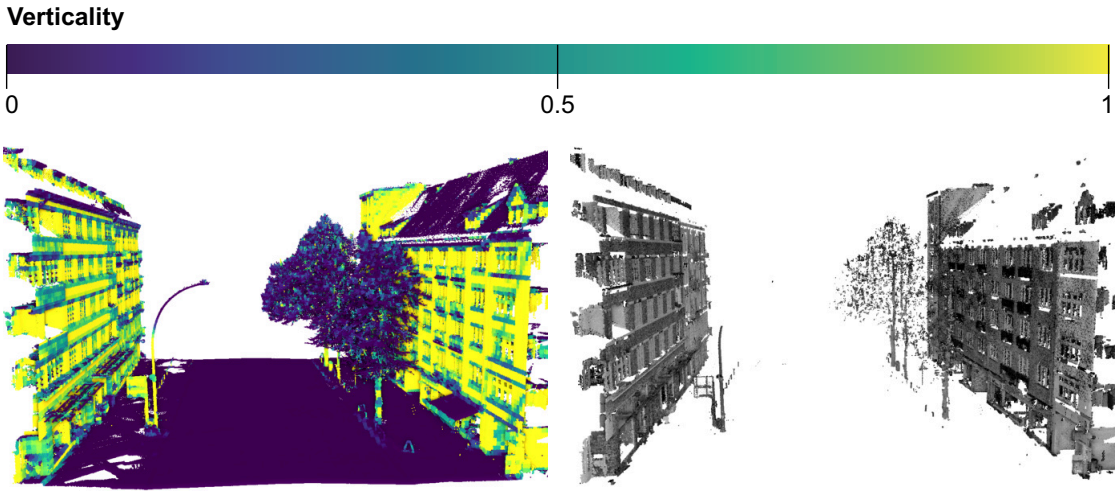
Neighborhood analyses in large amounts of data are very time-consuming, and the local point density is usually not required with per-point accuracy. This allows for an approximate solution performed as follows instead: All points of a dataset are placed in a voxel grid with an edge length of, for example, 30 centimeters. For each cell in the voxel grid, the number of points within (n_C) is determined and divided by the cell volume (V_C). The resulting value corresponds to the cell's local point density ($LDensity = n_C/V_C$) and is assigned to all points within the cell as an attribute value for their local density.

If not yet otherwise available, in many cases, the local density can be used to determine an approximate trajectory of a measuring vehicle, i. e., the path traveled because the point density along the travel path is much higher.

Areas with low local density within a dataset can often be excluded from subsequent analyses because not enough information is available for classification. This is especially the case in border areas of the collected data. Areas with high local density can be thinned out to save memory and processing time when the level of detail is not required.

2.4.2 Verticality

The *verticality* ($Vert$) of a point indicates how clearly its immediate surroundings belong to a vertically oriented surface. Examples of surfaces separated from their surroundings by this metric are building facades or posts whose surfaces stand vertically on the ground (Demantké et al., 2012). Figure 2.8 shows a visualization of the verticality metric.



(a) *Points colored by verticality.*

(b) *Points filtered by high verticality. Only points with* $Vert > 0.5$ *are shown.*

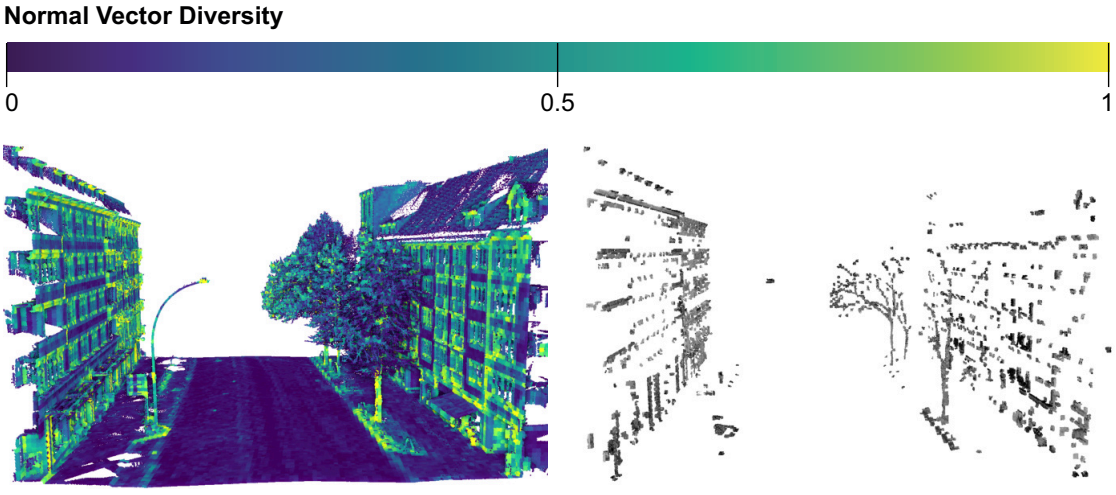
Figure 2.8: *Visualization of the verticality density metric in a 3D point cloud from the Hamburg dataset.*

There are two different approaches to determine the verticality value. One possibility is to apply the procedure for calculating the local density analogously and to use a narrow, infinitely high cylinder instead of a sphere as the measuring volume. This way, regions with many points positioned on top of each other can be recognized, as is the case with building facades, lamp posts, or signal posts.

Since this calculation is not well suited for large amounts of data due to complex neighborhood analyses, an alternative approach was developed. Here, the value is calculated by determining the fraction of normal vectors of all points in a defined neighborhood that are vertically oriented. The verticality is thus defined as the number of vertically oriented points (n_V) divided by the total number of points (n): $Vert = n_V/n$. As already described for the local point density, a voxel grid into which all points have been sorted can be used to look up local neighborhoods. All points within the same voxel receive the same verticality value. Again, a cell size with a side length of about 30 centimeters is suitable for most applications. Vertically oriented are considered all normal vectors whose angle to an up-vector is between 80 and 100 degrees. Since the normal vectors must usually be calculated for several analyses in any case, this approach is very efficient. As soon as verticality values are available, they can be used in connection with observations of the size of segments, e. g., for the efficient semantic classification of building facades.

2.4.3 Normal Vector Diversity

Normal vector diversity ($NDiversity$) makes it possible to locate vegetation areas in particular. The metric describes how strongly the normal vectors scatter in a local neighborhood. Trees and bushes do not have clearly structured surfaces, so the normal vectors of the points of such objects point in many different directions. For example, most buildings and the ground have a much smaller variety in their orientation in relation to the size of the surface. More complex facades might still show a high diversity, however. Thus, depending on the dataset, this metric might not be significant. Figure 2.9 shows a visualization of the normal vector diversity metric.



(a) *Points colored by normal vector diversity.*

(b) *Points filtered by high verticality. Only points with* $NDiversity > 0.8$ *are shown.*

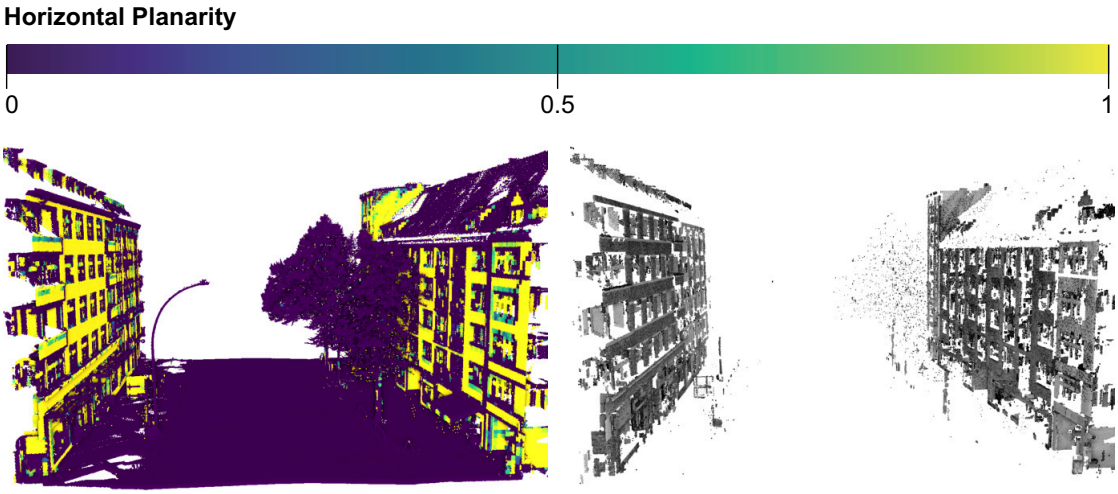
Figure 2.9: *Visualization of the normal vector diversity metric in a 3D point cloud from the Hamburg dataset.*

For the calculation, a voxel grid can be used as a local neighborhood as described above, or all points of a segment can be considered a group. For the group of points, the distribution of the normal vectors is determined. For each normal vector, the angles to the up-vector (which corresponds to the z-axis in the framework used) and the x-axis are determined and considered modulo 18 degrees. This results in 10 possible directions each. The combination of both angles results in a total of 100 possible orientations of the normal vectors.

The normal vector diversity is defined as the proportion of the occurring directions ($n_{directions}$) of the possible directions: $NDiversity = n_{directions}/100$. If all normal vectors point in the same direction, a value of 1/100 results. If all possible orientations are represented, a value of 1 results.

2.4.4 Horizontal Planarity

The *horizontal planarity* ($HPlan$) expresses whether the normal vectors of a particular group of points have a similar horizontal orientation, i. e., without considering the height dimension. Thus, this metric is suitable for the detection of, for example, building facades (Anagnostopoulos et al., 2016). Its advantage over verticality is that curved surfaces, such as those on advertising pillars or tree trunks, do not have high horizontal planarity and are, therefore, distinct from facades. Figure 2.10 shows a visualization of the horizontal planarity metric.



(a) *Points colored by planarity.*

(b) *Points filtered by high planarity. Only points with* $HPlan > 0.5$ *are shown.*

Figure 2.10: *Visualization of the horizontal planarity metric in a 3D point cloud from the Hamburg dataset.*

The points are placed into a voxel grid for the calculation, as already described for other metrics. The points within a voxel form the neighborhood to be considered. For these points, the normal vectors are projected into the x-y-plane, and if the x-value is negative, the vector is rotated by 180 degrees. All normal vectors are then located within a semicircle. This means that the direction of two precisely opposite normal vectors describing the same plane is no longer critical.

A mean value is calculated from these normal vectors, and the ratio of the normal vectors that deviate by more than 15 degrees from the mean is determined. This angle has proven to be particularly suitable in practice. The horizontal planarity is defined as $HPlan = 1 - (n_{15}/n)$ with n_{15} being the number of points with normal vectors deviating more than 15 degrees from the mean and n being the total number of points in the considered neighborhood. This value is assigned as an additional attribute to all points within the voxel under consideration.

Points with high horizontal planarity thus form a flat surface from a top-down view. Curved elements can still occur, e. g., an arched roof surface. These do not influence the metric. Nevertheless, horizontal curves such as on columns provide a low horizontal

planarity value as intended. However, narrow structures such as lamp posts or traffic light posts can still have a high horizontal planarity value in datasets with lower point density. They consist of only a single column of points, all having similar normal vectors. This could be avoided by requiring a minimum number of neighborhood points to calculate a high horizontal planarity value.

2.4.5 Post-Likeness

Many relevant "street furniture objects either contain a pole or are entirely shaped like a pole. For instance, lampposts are often pole-like objects (i. e., shaped like a pole), traffic lights are usually placed in a pole-like structure and street trees often have a pole-like trunk" (Cabo et al., 2014). *Post-Likeness* (PL) is used to detect such structures. The characteristic features searched for are points with many neighboring points at the top and bottom, but only a few at the sides. Laterally, the structure should only spread out to a limited extent. Figure 2.11 shows a visualization of the post-likeness metric.

(a) *Points colored by post-likeness.*

(b) *Points filtered by post-likeness. Only points that are considered post-like are shown.*

Figure 2.11: *Visualization of the post-likeness metric in a 3D point cloud from the Hamburg dataset.*

The metric can be calculated either segment-based or via a voxel grid. In segmental analysis, segments with a high verticality value (see Section 2.4.2) are examined. For this purpose, these are divided into sub-segments of fixed height, e. g., 50 centimeters. For each of these sub-segments, the diameter of an enclosing cylinder is calculated. If the diameter is smaller than a fixed limit for the maximum diameter of a typical lamp post or similar object, the sub-segment is considered to be post-like. In most datasets, a value of about 30 centimeters can be selected as the maximum diameter. Either all points of the sub-segments which fit into this diameter are attributed as post-like, and those of the other sub-segments as non-post-like, or all points of the entire segment receive the value of the share of the post-like sub-segments in all sub-segments as post-likeness.

The analysis based on a voxel grid initially places all points into a voxel grid as described before. The individual voxels are considered filled if at least four measuring points are located in the voxel or empty otherwise. In an analysis starting at ground level, all filled voxels are considered potential posts. If together with horizontally adjacent filled voxels, they form at most a 2x2 neighborhood and all voxels with a distance of 2 are empty, they remain candidates because the contained points thus form a locally contiguous structure that is not part of a more extensive object. If more neighboring voxels are filled, all these voxels are discarded as potential posts. The check is continued with the next layer of voxels further up, where the same neighborhood criteria are applied. Once all voxel grid layers have been checked, all groups of voxels previously identified as potential posts can be marked as actual posts that include at least three voxels directly above each other that meet the criteria. All points in the corresponding voxels are assigned a post-like attribute.

The post-likeness can be used to detect traffic signs, for example. For this purpose, a post structure on the ground must be identified, which may only deviate from its post shape beginning at a defined height. Elements placed there can serve as candidates for a search for traffic signs.

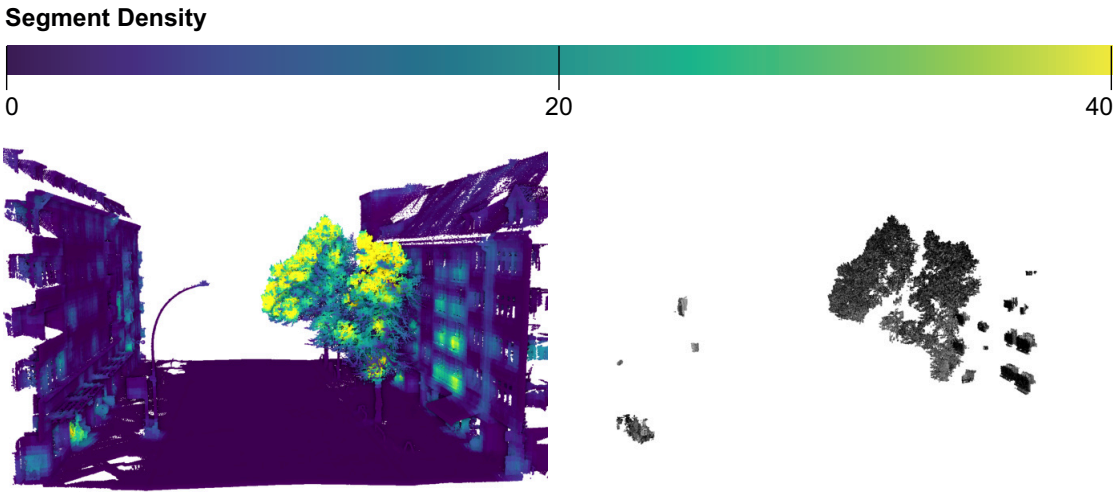
2.4.6 Segment Density

The *segment density* ($SDensity$) indicates how many segments, which were previously determined, are located in a certain area. Like the normal vector diversity, this metric can be used for the detection of vegetation. Since, as already described, the surfaces in vegetation do not usually form smooth surfaces, the normal vectors point in many different directions. The segmentation uses this normal vector information to detect connected segments. It results in many small individual segments in vegetation point groups, which are not combined into larger segments due to their structure. Figure 2.12 shows a visualization of the segment density metric.

The number of segments in a limited area, such as within a voxel in a voxel grid, thus provides information about the probability of vegetation. The number of different segment IDs of each voxel's points (n_S) is counted to determine this value. Typically, a constant value per dataset is used for normalization. For example, a median of the segment counts ($median_{n_S}$) can be used. The segment density thus results as $SDensity = n_S / median_{n_S}$.

2.5 Manual Creation of Training Data

A significant disadvantage of machine learning-based algorithms for semantic classification is that they require large amounts of data to train artificial neural networks. "While recording is nowadays straight-forward, the main bottleneck is to generate enough manually labeled training data, needed for contemporary (deep) machine learning to learn good models, which generalize well across new, unseen scenes" (Hackel, Savinov, et al., 2017). This section presents approaches to generate such annotated training data.



(a) *Points colored by segment density.*

(b) *Points filtered by high segment density. Only points with $SDensity > 20$ are shown.*

Figure 2.12: *Visualization of the segment density metric in a 3D point cloud from the Hamburg dataset.*

In contrast to other research areas like handwriting recognition or face recognition, not much publicly available and already classified training data exists for large mobile mapping 3D point clouds. Classified datasets can be found, but in most cases, these are single objects like tables and chairs or only small scans. Available extensive mobile mapping and aerial datasets are usually not semantically classified.

For a few years, more data for comparative research is gradually being made available. Hackel, Savinov, et al. (2017) also describe the problem of missing training data and provide a dataset called "semantic3D.net" with eight semantic classes, containing more than four billion labeled points. However, this dataset consists of scans taken by a stationary scanner and has different characteristics than mobile mapping scans. Better suited is the "Paris-Lille-3D" dataset from Roynard et al. (2018) which provides over 140 million points with 50 semantic classes on almost 2 kilometers of road. However, the scanners' arrangement and the resulting structures differ from the large datasets used in this work.

Due to the many distinct characteristics of the data depending on the acquisition techniques, it is often necessary to generate new training data from an existing dataset, which can be used for other acquisitions with the same hardware.

It is possible to build on already classified data, to avoid creating all training data manually. This can be obtained, for example, through a geometric analysis as presented in more detail in Section 3.2. Similarly, less sophisticated artificial neural networks can be used for automatic preprocessing, the results of which are subsequently refined. It is also possible to project manually created map data into the 3D point cloud, such as information about the position of road markings. If this data is precisely fitted into the 3D point cloud position, all points within the defined polygons can receive the corresponding

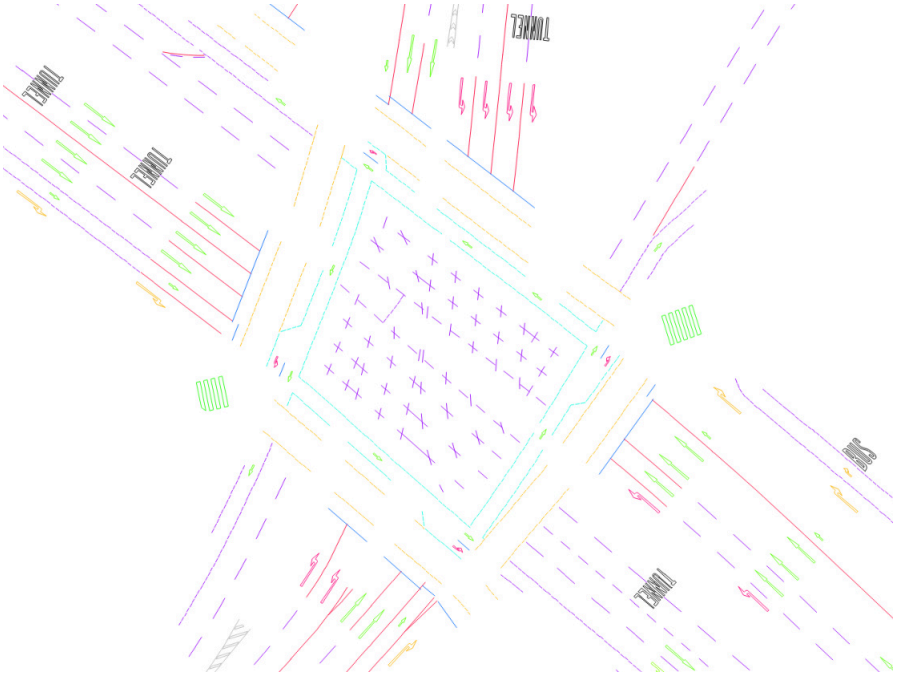


Figure 2.13: *Visualization of road marking map data from the Hamburg dataset. Individual road markings have positional and semantic class data. Each semantic class is represented by a unique color.*

type of road marking as a label. Figure 2.13 shows a map of road marking data that can be used for automated classification.

With these approaches, datasets can first be classified automatically and then serve as new training data after manual post-processing to ensure the correctness of the annotations.

For this manual post-processing or even a completely manual classification, a tool has been developed that enables users to annotate 3D point clouds and provides various interaction techniques for an efficient workflow.

First, the tool, shown in Figure 7.4 in the architecture chapter, enables a basic inspection of the 3D space with the 3D point cloud. The data can be navigated by using the computer mouse and keyboard. The parallel representation of the scene once with a freely movable camera and once with a top-down view enables efficient work. It ensures a good understanding of the scene, especially when annotating objects close to the ground, such as rails or road markings.

Semantic classes can be displayed or hidden individually. Write protection can be activated for each semantic class, for example, if only points are to be annotated that have not yet been assigned a semantic class.

The semantic classes' assignment to points is realized by selecting the desired class and then selecting the respective points. Different selection modes are available:

For example, a height filter selects only points in a defined height range. This serves,

e. g., to annotate rails, which are separated from the surrounding ground by their height. A 3D lasso tool allows users to select arbitrary areas of the 3D point cloud by drawing a polygon that, moving away from the viewer, continues as a frustum into the image space and marks all points contained in it. This method is handy for marking trees or masts. There is also a semi-automatic object detection, which marks, e. g., all connected points only separated by the ground from the rest of the 3D point cloud, starting from a starting point selected by a mouse click. Thus, for example, vehicles on the road can be marked efficiently. The semi-automatic analysis can also use other values than ground points as abort criteria for connected point groups, such as the intensity value, to quickly select all points of a road marking by a single click.

This tool enables the efficient generation of semantically classified data to be used as training data for machine learning-based approaches.

Research is also done on synthetically creating semantically classified 3D point clouds based on virtual 3D models (Danhof et al., 2015; Bechtold and Höfle, 2016). For this purpose, virtual 3D city models can be used whose objects already possess semantic information. One or more virtual LiDAR scanners travel along the modeled city streets and virtually emit beams like real scanners. If these rays hit an object, the three-dimensional coordinate is stored together with the object's semantic class as a point of a 3D point cloud. This allows creating "perfectly" classified 3D point clouds—within the correctness of the underlying semantics defined by the virtual 3D city model. However, the artificial datasets generated in this way differ in their characteristics from real-world data. For example, noise and outliers would have to be artificially added to obtain a similar appearance. In addition, the intensity attribute, for example, is difficult to map because information on surface materials is missing. The problem of few available datasets is shifted to 3D city models, of which there are also only a few freely available datasets whose level of detail is suitable.

Chapter 3

Semantic Classification of Infrastructure Elements in 3D Point Clouds

Semantic classification of 3D point clouds denotes the process of assigning each point an attribute describing one or more semantic classes to which it belongs. Basic semantic classes that are often used in the geospatial domain include *ground*, *building* and *vegetation*. However, depending on application cases, the classification can also be more specific. For example, for infrastructure-related applications the categories could include *vehicle*, *rail*, *road marking*, *traffic sign*, *traffic light* and *person* or even such detailed classes as *straight arrow*, *priority road sign* and *stop line*. Figure 3.1 provides an overview of the most important semantic classes in the environment of roads and train tracks that have been investigated in this thesis.

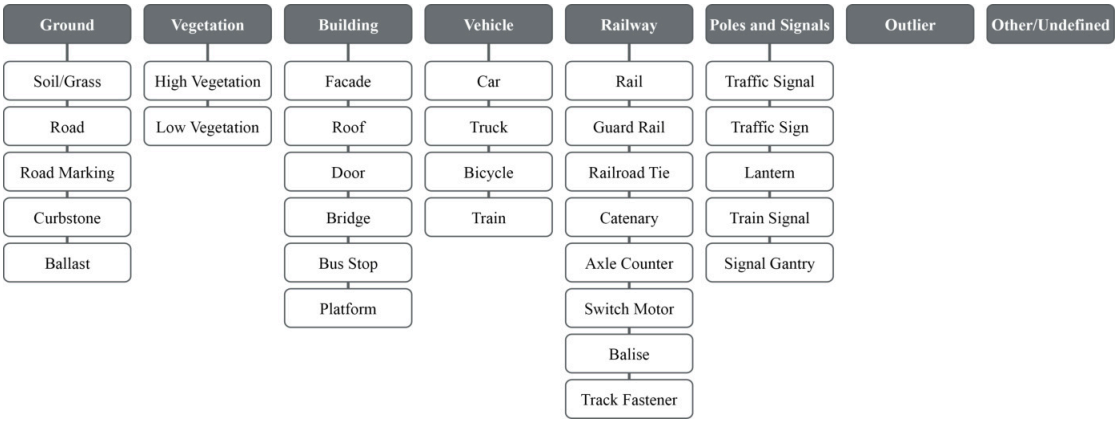


Figure 3.1: *Hierarchical visualization of the most important semantic classes in transport infrastructure mobile mapping scans.*

Semantic information associated with individual points of a 3D point cloud is required for most applications, particularly for analysis tasks, to detect, examine, and evaluate objects in these scans. For example, an application might want to determine how far vegetation is from roads and railways and whether it extends into areas that need to be kept clear. It can also be determined to what percentage an area is covered by forest or

buildings and how far a traffic sign is placed from the curb. Furthermore, basic semantic information is the basis for object recognition, for example, identifying individual trees within the vegetation or examining all detected road markings for their exact type.

This chapter explains semantic classification approaches in detail. First, Section 3.1 presents related work on the topic. Section 3.2 explains geometric approaches for the classification. Sections 3.3 and 3.4 describe semantic classification via machine learning techniques using rendered images and spatial data, respectively.

3.1 Related Work

Semantic classification is a key aspect in the analysis of 3D point clouds. As such, many approaches have been published, focusing on various specific aspects, semantic classes, or object types.

While the semantic classes of interest differ based on the use case, a common distinction into the most basic semantic classes usually separates the 3D point cloud into at least "building/structure", "vegetation", and "ground". The classification approaches can be based on geometric analyses (D. Chen et al., 2017), machine learning techniques (Boulch, Saux, et al., 2017) or a combination of both. Geometric approaches (see Section 3.2) base decisions about the respective semantic class of points or groups of points on calculated metrics within predefined thresholds for specific object types. Machine learning techniques as described in Sections 3.3 and 3.4 use annotated training data for the training of an artificial neural network, which afterward makes decisions about the semantic class based on the "recognition" of similar structures or environments compared to the training data.

Explicit rules can be defined to distinguish semantic classes by geometric attributes (Grilli et al., 2017), e. g., for separating nature from human-made structures (Yao and Fan, 2013). Usually, well-defined semantic classes are used for the classification. For example, Niemeyer et al. (2012) distinguish the five object classes "building", "low vegetation", "tree", "natural ground", and "asphalt ground" as shown in Figure 3.2 and use Conditional Random Fields (CRFs) to separate the individual classes.

The separation between vegetation and non-vegetation is discussed by Rutzinger, Höfle, et al. (2008), who use full-waveform (FWF) data for the analysis. Many analysis approaches have been developed for aerial 3D point clouds, but they cannot be applied directly to mobile terrestrial scans. Nüchter et al. (2006) classify points that an indoor rescue robot has captured. Rusu, Marton, et al. (2009) also focus on indoor data such as kitchen areas and describe an efficient semantic object labeling method. Rutzinger, Pratihast, et al. (2011) analyze outdoor mobile mapping scans for parameters like crown diameter and stem height for individual trees.

Post-like structures such as lanterns, traffic signs, and tree stems are often found in mobile mapping datasets. Researchers seek to detect and classify these objects correctly by their shape (Pu et al., 2011). A dedicated detection of lanterns and traffic signs is described by Lehtomäki et al. (2010). It is also possible for post-like structures to voxelize

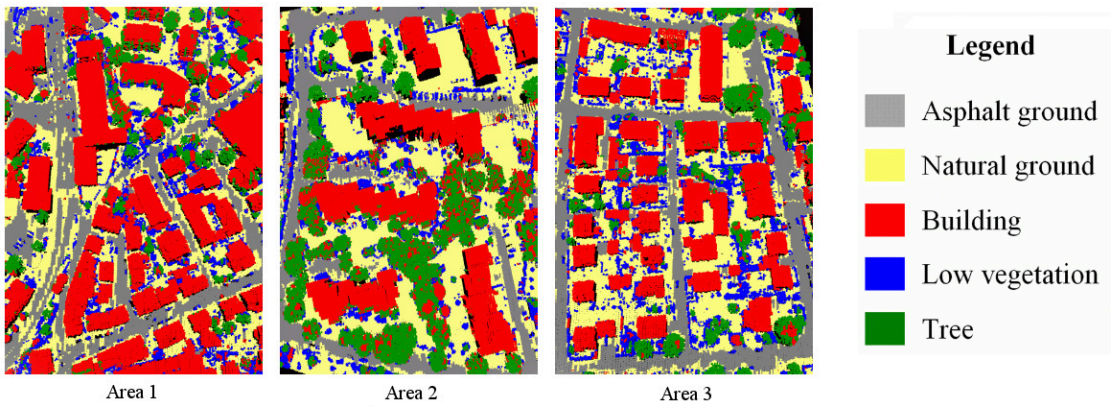


Figure 3.2: *Classified 3D point cloud with five semantic classes displayed in distinct colors. Figure from Niemeyer et al. (2012).*

the 3D point cloud data for segmentation and further classification analyses (Aijazi et al., 2013).

X. Meng et al. (2009) describe a popular algorithm for detecting ground points based on their relative height and orientation. The ground detection implemented in the framework used in this thesis is based on Meng's approach.

Analyses are either based on explicitly defined geometric characteristics of the 3D point cloud's topology (Richter, Behrens, et al., 2013) or machine learning techniques. Older approaches based the analysis on probabilistic Markov networks (Triebel et al., 2006; Munoz et al., 2008), newer solutions require previously trained artificial neural networks to identify the semantic classes of objects (Zhou and Tuzel, 2018). Fukano and Masuda (2015) analyze scan profile characteristics, which are processed by supervised machine learning methods to detect utility posts, lanterns, traffic signals, and other post-like objects. In recent years, with machine learning on the rise, using the 3D point clouds' internal structure has become increasingly popular, as exemplified by PointNet++ and similar networks (Qi, Yi, et al., 2017; Winiwarter et al., 2019). They all work on the spatial data of the 3D point cloud to attribute all points with semantic information.

Especially and prominently used are Convolutional Neural Networks (CNNs) that are a class of networks inspired by biological processes and find use mainly in the automated analysis of image data (LeCun et al., 2010). Freely available frameworks have been developed to make techniques for the automated analysis of images easy to use in own applications (Pulli et al., 2012). Many research fields, such as face recognition, license plate identification, or medical imagery analysis require efficient object detection in images.

Viola and Jones (2001) present an often-cited algorithm that can be used to detect, e. g., faces in images. U-Net is widely used for image segmentation (Ronneberger et al., 2015) in many areas, not only the original medical context, and enables the automated detection of specific areas in images, such as cancer cells but also roads in aerial images (Z. Zhang et al., 2018). Lanes and arrow markings are detected by Vacek et al. (2007),

who extract road marking information from images taken from a car. Veit et al. (2008) present an approach to evaluate the performance of algorithms for road marking detection in images in general. Extraction of road markings is performed on mobile laser scanning data as well. B. Yang et al. (2012) use the reflective properties of road markings to extract them from mobile mapping scans. Guan, Jonathan Li, Y. Yu, et al. (2014) perform a curb-based road extraction, followed by rendering intensity images of the 3D point cloud and a final extraction step, segmenting the areas containing road markings. Going into more detail, Y. Yu et al. (2014) distinguish seven specific types of road markings by five classification methods.

Autonomously driving cars require a high-speed evaluation of LiDAR data of their surroundings to evaluate the current environment without delay. Caltagirone et al. (2017) investigated the use of CNNs for this purpose. Some objects in road space, especially people, need to be detected in the data for safely controlling vehicles (Navarro-Serment et al., 2010). Guan, Jonathan Li, Cao, et al. (2016) give an overview of how approaches from various publications use LiDAR information for road information inventory. Sester (2020) illustrates several use cases for data acquired by mobile mapping vehicles, such as analyzing the availability of parking spaces or detecting roadblocks from the collective behavior of car drivers.

Many of the methods used for road data can also be transferred to railway networks. Stein et al. (2016) investigate how tracks of light rails in LiDAR scans can be detected automatically through variations in the distance values. It is important to detect specific assets in the track environment for monitoring and maintenance. Arastounia (2017) points out this necessity and presents an algorithm for how rail tracks and contact cables can be detected via a geometrical analysis of point positions based on an automated seed point search. Gézero and Antunes (2019) describe an approach to evaluate 3D point clouds using a vertically mounted scanner's angular information at a train's front. Along an imaginary line beneath the scanner, they determine the rails on either side of it and the ballast's dimensions. More detail-focused in his analysis is Taheriandani (2016), who describes approaches for track analysis with LiDAR scanners directly aimed at the rails to detect even the slightest deviations. Similar to road image analysis, CNNs can be used for detecting rail defects on image data from railways (Shang et al., 2018). A widely used implementation is YOLO (You only look once), which only returns labeled bounding boxes for input images, but can process the provided images very fast (Redmon and Farhadi, 2018). In a railway context, Yanan et al. (2018) use YOLO to detect problems on the surface of rails, and Y. Yang et al. (2020) use it to recognize numbers on signal posts in images.

The concept of scan profile analysis is used by Hu and Ye (2013) for detecting buildings in aerial data. Yan et al. (2016) present an approach using them to recognize road markings in mobile mapping data.

The above approaches show that there is no single way to classify mobile mapping 3D point clouds. Depending on the use case, the focus is set differently. Geometric methods and machine learning exist in parallel and are both continuously researched. Therefore, both approaches are examined in more detail below.

3.2 Geometric Approaches

In contrast to machine learning-based approaches, geometric classification calculates several metrics and considers the resulting values to find fitting semantic classes for points and segments in a 3D point cloud. The decisions are mainly based on predefined threshold values for the metrics, stating whether a specific semantic class should or should not be assigned. In the following, approaches for geometry-based semantic classification are presented.

This section is partially based on the author's publication in J. Wolf, Richter, and Döllner (2019).

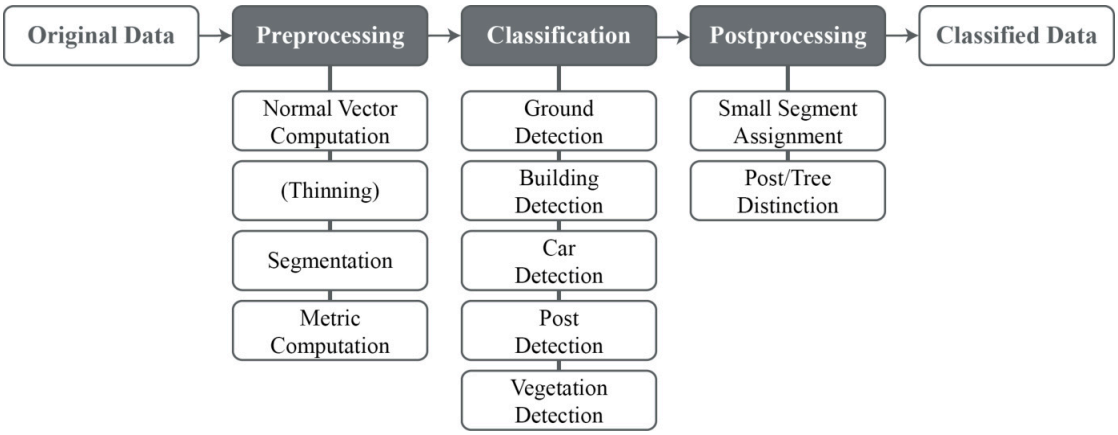


Figure 3.3: *Process for a geometric semantic classification of 3D point clouds.*

Figure 3.3 shows an exemplary process for a geometric semantic classification. For many processing steps, it is necessary to have the normal vectors computed. The 3D point cloud data may have to be harmonized in advance, and metrics are calculated during the preprocessing as described in Chapter 2. Optional 3D point cloud thinning can take place at this point. Required besides is a segmentation of the data as described in Section 2.3.2. Points forming connected surfaces without sharp edges are grouped in segments. Afterward, the classification will focus on detecting pre-specified semantic classes, and, finally, a postprocessing step will work on small improvements, resulting in the classified dataset.

In a first step during the classification phase, the **ground detection** is performed. The detection is based on the idea that the ground usually forms a large horizontal surface at the bottom of the 3D point cloud. Using an adapted and extended approach based on the description from X. Meng et al. (2009), the ground points are extended from the 3D point cloud data. Edges and slopes are considered to the extent that can be set in the parameters for the detection. Although initially developed for aerial data, this implementation can also efficiently detect ground points in mobile mapping data because the main characteristics of ground points remain valid. The ground is still below

the scanning vehicle and results in a large enough surface despite objects like cars and trees, partly shadowing areas of the ground in the scanning environment.
The ground detection is using a voxel grid where all voxels have infinite height. Thus, only the points' x and y coordinates determine to which voxel a point belongs. Scan lines are used in the algorithm to detect ground points. They move axis-aligned and diagonally through the grid in all eight possible directions. Slopes and elevation differences are considered during the detection. Whether a point is considered a ground point or not is based on majority voting by each scan line.

The **building detection** is based on the detection of large segments with appropriate orientation and sizes. Adjacent vertical surfaces above a specific size are almost exclusively building facades. If necessary, the number of points in the considered segment can be additionally considered in relation to its size to ensure sufficient density for a closed surface. This excludes, for example, some fences from classification as buildings.

The location of large, flat segments is analyzed in relation to previously detected facades to identify building roofs. Not many roof points occur in mobile mapping data because the scanner is located close to the ground. However, sloped roofs at a greater distance can still be recorded. Due to their position above detected facades, they can be recognized as roofs and assigned to the semantic class "building".

Car detection is based on segments of predefined sizes as well, but in addition, the distance to detected ground points is of importance. Thus, as a threshold value, it can be assumed that a car segment must not be further than 50 centimeters away from detected ground points. The proximity to the ground can be used to exclude segments whose rough shape would fit vehicles but are located too far above. A small tolerance range must be allowed because, for example, other vehicles can shade parts of the ground so that no ground was detected directly under the car.

Post detection is performed on the remaining points based on the post-likeness value described before. Points located above each other without other points in a certain distance around them either represent tree stems, or they are posts such as lamp posts, traffic signal posts, or flag posts. The distinction between these two is not always easy and, therefore, done in a postprocessing step after posts and vegetation have been analyzed.

Vegetation detection is based on segment density analysis because it does not have plane surfaces and thus is segmented into a large number of small segments. Points in areas with a high segment density are usually vegetation and can be classified as such. It is also possible to additionally consider the normal vector diversity metric. This metric also identifies vegetation by the scattered normal vector directions in vegetation areas.

As mentioned before, some **postprocessing** is required after these semantic class detection steps. First, all small segments which have not been assigned any semantic class can be analyzed. If these segments are surrounded only by segments of one specific semantic class, it can be assumed that the small segments belong to the same semantic class, which can be assigned to them. Also, because posts and tree stems have a similar geometric structure but belong to two different semantic classes, these must be analyzed again. If a post-like structure is detected with many vegetation points above it, it is considered a tree stem and will be reclassified as vegetation. The reclassification might

lead to problems if, e. g., a traffic sign is located under tree branches and cannot be distinguished from a tree stem. If the detection of traffic signs is of high importance for the current use case, an additional analysis of intensity values can be performed, checking whether a highly reflective traffic sign is mounted on the post structure. In this case, it would not be considered as vegetation. Similarly, a lantern with just a small number of erroneously classified vegetation points on top can be reclassified as a post because vegetation usually exists next to other vegetation and not in small separated and elevated patches.

As described in this section, semantic classification based on geometric analyses requires a large number of thresholds. The values are partly based on average values, which can be computed for each scan. However, they often require individual configuration for each new dataset, e. g., when the data was captured with a different setup in another city. The configuration is the main reason for the need for different classification approaches, which enable an even higher grade of automation for the analysis process.

3.3 Machine Learning on Rendered Images

In addition to the classification by geometric methods, machine learning approaches can also be used. These can either work on the spatial data directly or, as this section describes, on rendered images.

Established methods for image analysis using CNNs have been used for several years. As described in Section 3.1, these are applied in several use cases. For example, face recognition, separation of cells such as shown in Figure 3.4, and recognition of specific object classes in images. Other road users such as pedestrians, cyclists, and other vehicles can be detected in camera images taken by autonomously driving vehicles. Similarly, traffic signs and road markings can be recognized.

Based on these existing approaches, the idea of applying semantic classification of 3D point clouds via image analysis is justified. The 3D point cloud can be rendered from a specific position, the resulting image is analyzed with a CNN, and the result is transferred back to the 3D point cloud.

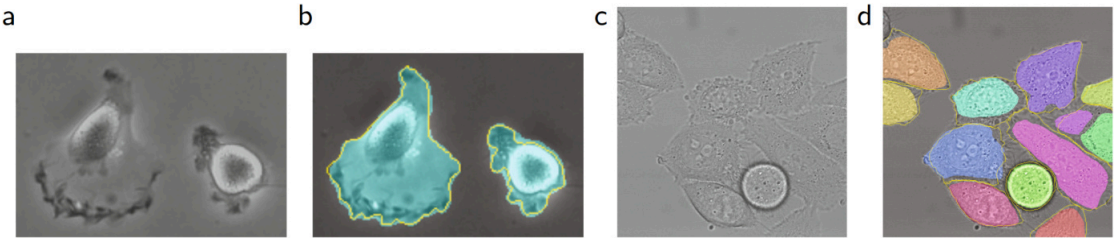


Figure 3.4: *Result on the ISBI cell tracking challenge. (a) Part of an input image of the "PhC-U373" dataset, (b) Segmentation result (cyan mask) with manual ground truth (yellow border), (c) Input image of the "DIC-HeLa" dataset, (d) Segmentation result (random colored masks) with manual ground truth (yellow border). Figure from Ronneberger et al. (2015).*

The general approach is described, for example, by Boulch, Guerry, et al. (2018), who make images from different perspectives in their "SnapNet" to classify objects mapped on them. In contrast to the work presented in this thesis, they are converting the 3D point cloud data into a mesh, which requires a preceding normal vector analysis.

In the following two subsections, the analysis is considered in detail for two exemplary use cases. Subsection 3.3.1 describes the analysis of two-dimensional objects on the roadway (road markings and utility hole covers), and Subsection 3.3.2 describes the analysis of scan profiles taken by trains to detect, e. g., tracks and signals.

In this work, the networks U-Net (Ronneberger et al., 2015) and YOLO (Redmon and Farhadi, 2018) are used for the classification of image data.

3.3.1 Ground Image Analysis

This section is partially based on the author's publications in J. Wolf, Richter, Discher, et al. (2019), J. Wolf, Richter, and Döllner (2020), and J. Wolf, Pietz, et al. (2021).

In road maintenance, the condition and markings of the roadway are of particular interest. On the one hand, it should be ensured that the roadway is in good condition and does not contain any potholes. On the other hand, vehicle drivers need to find easily recognizable markings to keep to the lane and correctly perceive turning lanes and bicycle tracks.

For regular analysis of roadways, it is therefore essential to capture this data automatically and precisely. This section describes the automatic recognition of lane markings, which can be compared with as-built data to detect wear or changes after construction work. The approach can also be used to detect utility hole covers, for example, to exclude them from further investigation in conjunction with the GPR analysis from Chapter 5 when anomalies are found to detect potholes specifically.

Road markings and utility hole covers are located flat on the ground, and their height, if at all, differs only to a minimal extent compared to their environment. Therefore, these objects are predestined for visual recognition as they can easily be represented in two-dimensional images in a top-down view of the 3D point cloud data.

The approaches presented in this thesis aim to detect objects in billions of points (e. g., 3D point clouds of whole cities), so data reduction is an important aspect. Only the road itself along the captured path is required to detect road markings and utility hole covers. A trajectory is captured during the scan of the 3D point cloud data, which describes the measuring vehicle's exact path. The 3D point cloud of interest can be cropped along this trajectory, e. g., 10 meters to its left and right. Suppose the recording path should not be available. In that case, relevant areas can also be filtered by analyzing the local point density because regions close to the measuring vehicle have a much higher density than areas further away. Outliers are removed by outlier filtering in the remaining data as described in Section 2.2.3. Filtering is applied to remove noise within the data that might affect the top-down rendering of the 3D point cloud.

A ground detection as described above identifies ground points based on their relative height and orientation. Higher points can be removed, so points representing buildings

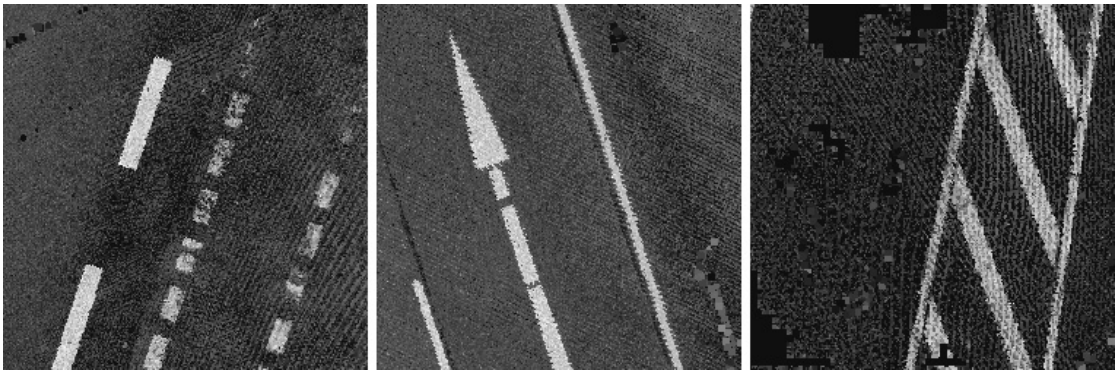


Figure 3.5: *Examples of rendered images from a 3D point cloud from the Essen dataset, showing several types of road markings. Intensity values are represented in grayscale, lighter colors have higher intensity values.*

or high vegetation will not be analyzed. The remaining 3D point cloud only consists of ground points along the measuring vehicle's trajectory without outliers. Following this preprocessing step, 3D point clouds of the test datasets have on average only about 60% of their original points left.

For the region of interest, images of the 3D point cloud can be rendered in a top-down view and are subsequently processed by the neural network.

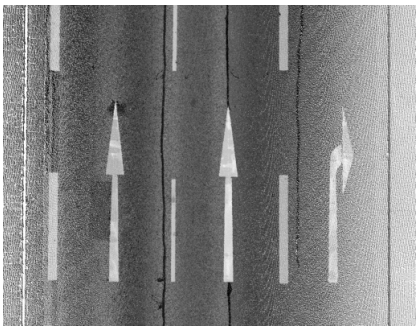
The following example illustrates the approach: The renderer receives a 3D point cloud as input and generates images of 128 by 128 pixels in orthogonal projection, as shown in Figure 3.5. Each image represents about 4.5 by 4.5 meters of the road's surface. The images are generated slightly overlapping and cover the entire area to be analyzed. By removing higher points in the preprocessing step, overhanging vegetation is removed, and the road surface can be displayed entirely.

Each image contains a channel storing the intensity value of the point visible in each pixel and a channel storing the ID of the point rendered in this position. The latter is used for projecting the classification results back into the 3D point cloud.

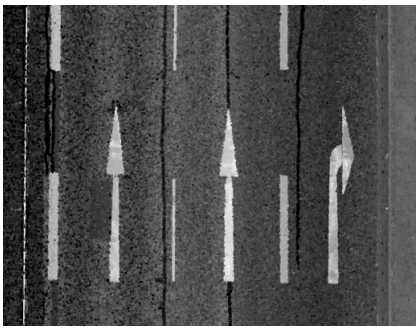
Empty pixels cause difficulties in the classification. Therefore, the results can be improved if 3D point clouds are rendered using a rendering technique that fills gaps between neighboring points, such as *paraboloids* (Schütz and Wimmer, 2015a). Rendering with different primitives is shown in Figure 3.6. Using paraboloids will fill more pixels in areas with lower density to avoid gaps in the resulting image while preserving sharp edges of individual structures, as shown in Figure 3.6(g).

The rendered images are used as input for the previously trained neural network, using adapted versions of U-Net for the classification. The result is an output mask for each input image. A semantic ID describes for each pixel, whether it represents background (road) or an object such as a particular type of road marking or a utility hole cover.

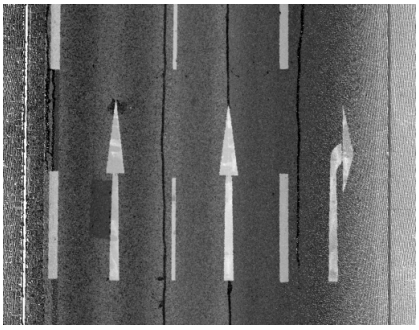
A binary separation was used for utility hole covers, distinguishing them from the road in black and white result images. Figure 3.7 shows images rendered from a 3D



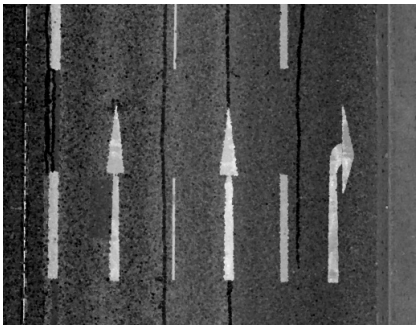
(a) *Small points (2 pixels).*



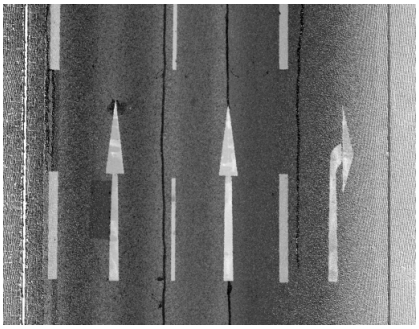
(b) *Large points (6 pixels).*



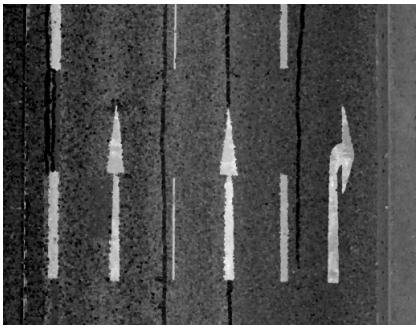
(c) *Small circles (1 x 1 cm).*



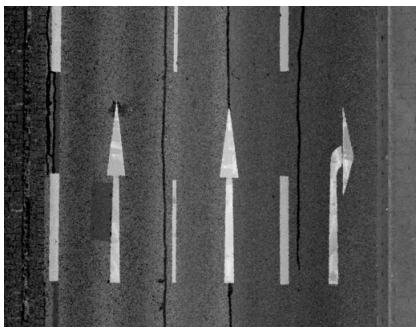
(d) *Large circles (3 x 3 cm).*



(e) *Small quads (1 x 1 cm).*



(f) *Large quads (3 x 3 cm).*



(g) *Paraboloids.*

Figure 3.6: *3D point cloud from the Essen dataset rendered with different primitives.*

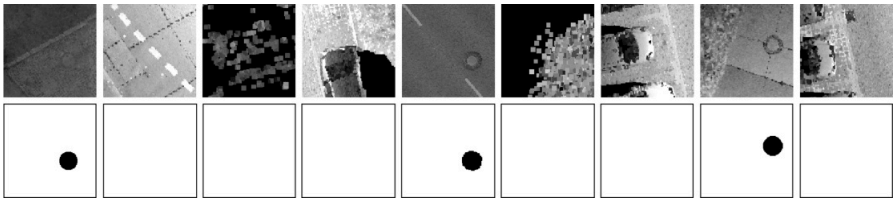


Figure 3.7: *Top row: Images rendered from 3D point clouds from the Essen dataset for utility hole cover detection. Bottom row: Classification result masks; black pixels represent detected utility hole covers, white pixels represent background.*

point cloud and the utility hole cover classifications' result masks. For road markings, a multi-class implementation produces distinct colors for the different types of detected road markings.

In Europe, utility hole covers following EN 124 are usually not larger than 85 centimeters in diameter. For the rendered images' overlapping area, a default value of 16 pixels is used, representing about 60 centimeters. The overlap ensures that at least one image contains the complete utility hole cover, even in the worst case.

The appropriate representation of road markings, on the other hand, is more complex. Road markings can be several meters long, such as arrows or barred areas. Therefore, it was found that a different approach is better suited for road markings, which clusters the markings beforehand and only then classifies them, similar to the method of Wen et al. (2019). Using U-Net, binary classification is first performed that identifies all pixels belonging to road markings. This classification works analogously to the classification of utility hole covers.

The areas detected as lane markings are clustered in a multi-step process. The procedure assumes a certain minimum distance between pixels of different road marking classes so that contiguous pixels can be assigned to the same class. In a first clustering, arrow markings are to be detected. These stand alone but can consist of multiple parts. For example, the arrowhead is often separated from the line, which can also consist of several partial lines, as shown in the center image of Figure 3.5. For this purpose, a large maximum distance between points is defined as a threshold for clustering. Thus, arrows are placed in their own clusters, but other markings of different classes can also fall together in one cluster and need to be analyzed in a later step.

In an optimization step, all clusters with insufficient points and those covering a too small area can be discarded.

All generated clusters are rendered individually. The generated images have binary values per pixel, stating whether they are part of a road marking as detected by U-Net or not. An optimization in image space applies a combination of a dilation followed by an erosion to close small gaps in areas where road markings might be dirty or rubbed off and those occurring in arrow lines. Figure 3.8 shows some rendered images.

The image is rotated and cropped to the object-oriented bounding box (OOBB) of the marking with the longer side in the vertical direction. Cropping and rotating are

Figure 3.8: *Examples of rendered 132 by 132 pixel images of detected road marking clusters. White pixels represent areas where a road marking was detected, black pixels represent background.*

important for easier processing because all road markings are then oriented in the same way.

An initial classification is performed using a Support Vector Machine (SVM) previously trained on road markings like described by Greenhalgh and Mirmehdi (2015). Only clusters classified as one of the different arrow types are finished processing. The points from all other clusters are considered again in the next step.

In the second clustering step, a smaller maximum distance is chosen for clustering, for example, to separate individual strokes within boundary lines. For these newly created clusters, a classification is performed with another SVM trained for all available line types.

An alternative implementation for road marking classification uses just a multi-class U-Net for a combined detection and classification. Therefore, larger orthographic images must be rendered to depict most road markings as a whole. Images that cover 10 by 10 meters have proven suitable. These are also processed overlapping, here with an overlap area of 5 meters. This means that each pixel is represented in four images. Instead of using U-Net to create only a binary mask that delineates road markings from the surrounding area, a variant that can distinguish multiple classes is used. The binary U-Net implementation maps a 64-element feature vector into a two-element feature vector in the final step. This step is modified to perform the mapping into a feature vector with one more element than possible road marking types. Thus, U-Net's prediction's pixels determine which road marking type they represent or if they do not belong to a road marking. In the interpretation, a weighted majority decision is performed between the overlapping areas, giving higher weight to the result of those images in which the respective pixel was located further in the center of the rendered image. The majority voting is applied to minimize artifacts at the edges.

After the semantic classification of utility hole covers or road markings, the information about the semantic class of individual pixels can be transferred back to the 3D point cloud through the point ID channel. The point within the 3D point cloud whose ID matches that in this image channel is assigned the recognized semantic class as an additional attribute. The point density is generally higher than the resolution of the rendered images. Therefore, several points are covered by one pixel. All points in the immediate neighborhood of the point classified also receive its semantic class without being noted in the ID channel itself. For road markings, the points' intensity values can be used as a criterion to decide which of the neighboring points belong to the same marking. Utility hole covers are circular objects with standardized sizes so that a best-fitting circle can determine all points belonging to the detected cover.

3D point clouds that have been semantically classified as described can be used to, e. g., automatically create road maps or compare the real-world situation with existing map data. Chapter 4 describes these use cases in detail.

3.3.2 Scan Profile Analysis

This section presents how scan profile data of LiDAR scans from railroad environments can be converted into image data to subsequently identify objects therein with established image analysis methods and use this information for further analyses. This section is partially based on the author's publication in J. Wolf, Richter, and Döllner (2021).

Besides road infrastructure, another essential backbone of today's transportation sector is the railway infrastructure. Many national railroad companies, such as Deutsche Bahn in Germany and SBB in Switzerland, operate measuring trains that examine the tracks and their surroundings in detail during the journey (Wirth, 2008). Besides image data, LiDAR scans are used for precise measurements, resulting in 3D point clouds.

Figure 3.9: *Measurement train "Limez III". LiDAR scanners are attached to the front of the train. Figure from Wirth (2008).*

Figure 3.9 shows a train with LiDAR scanners mounted at the front of the train. These scanners typically generate 3D point clouds with a 5 to 15 cm scan profile distance. The rotation of the laser beam during the measurement and simultaneous movement of the train result in a series of captured points in the form of a helix. Each point can be located by its distance from the scanner and the laser's current angle at the time of measurement. The points of one rotation of the laser are called scan profile. The resulting points can be visualized as a 3D point cloud of the entire track environment. Figure 3.10 shows a section of such a 3D point cloud. The individual scan profiles, which are here 8 cm apart, are clearly visible.

The scan profile analysis is based on the assumption that the performance of 2D image recognition can be used due to the nature of the data when individual scan profiles are available in a dataset. The three-dimensional dataset can be reduced to a two-dimensional image recognition problem similar to the image analysis described in the

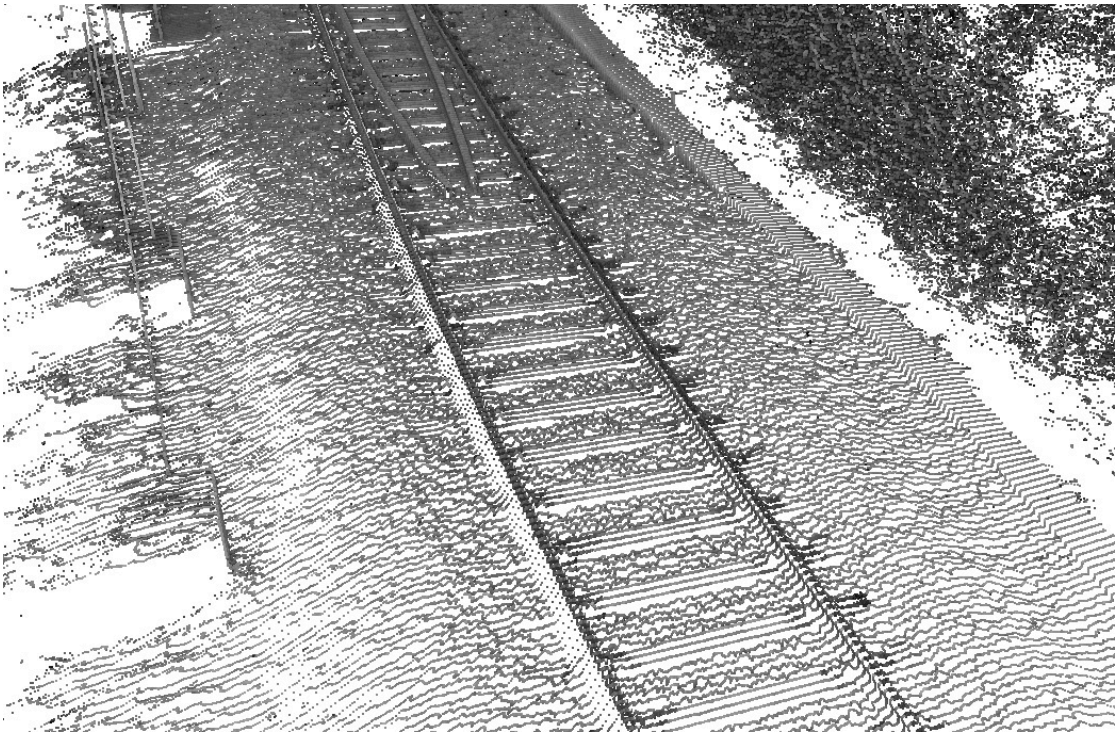


Figure 3.10: *3D point cloud from a railroad mobile mapping LiDAR scan from the Railroad dataset, showing individual scan profiles. Points are colored based on intensity values.*

previous section for ground images. Typical objects that can be found in railroad data scans and should be semantically classified with this approach are shown in Figure 3.11.

When rendering images from 3D point cloud data for object classification, positioning the virtual camera is of utmost importance. As described in the previous section, top-down views are suitable for detecting objects such as road markings and utility hole covers in mobile mapping data. However, catenaries and signal bridges above railroad tracks occlude essential parts of the track in a top-down view and hinder comprehensive classification. They cannot simply be removed from the data because they are to be included in the semantic classification. Furthermore, in tunnels, catenaries and signals are often mounted to the ceiling, making it difficult to perform a top-down view analysis. Using the scan profiles with objects captured from the train's perspective is an alternative to the top-down view.

For the semantic classification process, first, individual scan profiles are identified. This process is described in Section 2.2.5.

Then, all scan profiles can be rendered individually as 2D images. A particular color (e. g., white, represented by a value of 255) is used as masking for the areas not containing data. All other pixels can be colored with gray levels according to the intensity of the measuring points at the respective position, mapping the lowest intensity to black (0), the highest to almost white (254). An additional image channel containing the rendered

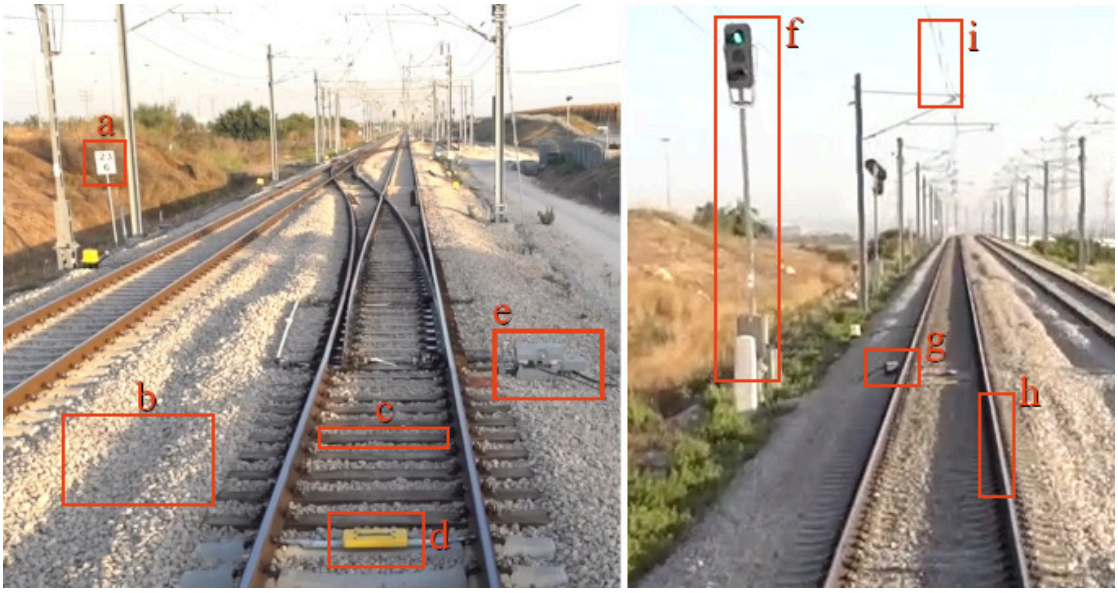


Figure 3.11: *Typical objects found in railroad environments: Sign (a), ballast (b), tie (c), balise (d), switch motor (e), signal post (f), axle counter (g), rail (h), catenary (i). Images from the Railroad dataset.*

(a) *Two rails with a tie in between.* (b) *Two rails with ballast in between.*

Figure 3.12: *Two scan profiles of a railroad track on a ballast hill. Grayscale colors represent the intensity values of the points.*

points' IDs must also be included to map the results back into the 3D point cloud after the image classification. If several points are rendered in the same pixel, the last rendered point's ID is stored.

Figure 3.12 shows the difference of the surface's smoothness between two rails depending on whether a tie is placed at this position or the ballast is exposed. Thus, ties can be identified if the scan profiles are placed at suitable distances from each other so that each tie and ballast area between two ties is captured by at least one scan profile. With the above-mentioned scan profile distance of 8 cm, each tie is visible on 3 to 4 consecutive ties.

A large variety of objects can be found in the immediate neighborhood of the rails. Figure 3.13 shows a scan profile at the beginning of a track switch. The switch tongue is placed close to one of the rails and a box with the switch motor on the other side of the rail. The box shadows the area on the right side, so this part of the ballast hill is missing.

Figure 3.14 shows two scan profiles of a signal post next to the rails. Due to its shape, the front of the actual signal can be seen in multiple scan profiles before the signal

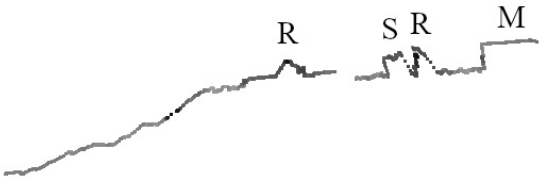


Figure 3.13: *Scan profile of two rails (R), a switch tongue (S) and a box containing the motor (M) next to the rails.*

(a) *Front part of the signal.* (b) *Signal post, six scan profiles behind the other one.*

Figure 3.14: *Scan profiles of a signal post next to two rails. An axle counter is attached to the outside of the left rail.*

post. In total, the structure has a length of approximately 1.5 meters.

A large number of rendered images is required for training purposes. This data can be generated manually by labeling individual pixels and bounding boxes within the scan profile images. A faster approach would be using pre-classified 3D point clouds, created either manually or by a different automated approach. Then, semantic information can already be included when rendering the 2D images.

Suitable networks for the analysis of the rendered images are, for example, U-Net and YOLO. Both follow a different approach but could provide similarly relevant results for the application described here. While U-Net classifies individual pixels, YOLO only determines bounding boxes for recognized objects. However, since there are hardly any overlapping objects in individual scan profiles, mapping the information to individual pixels is not a problem. When using YOLO, all non-background pixels within the bounding box of a recognized object could get assigned the corresponding semantic class. They would then be treated similarly to the images classified pixel by pixel with U-Net.

Figure 3.15 shows exemplary results of the semantic classification with YOLO and U-Net on a scan profile.

Once the semantic class for each pixel is determined, the information can be mapped back into the 3D point cloud by using the ID channel in the same way described in

(a) *Classification result using YOLO. Bounding boxes are placed around identified objects.*

(b) *Classification result using U-Net. Points are colored based on semantic class: Signal (red), rail (green), axle counter (blue), other (black).*

Figure 3.15: *Exemplary semantic classification results of the scan profile shown in Figure 3.14(b).*

Section 3.3.1. In case the point density is higher than the resolution of the rendered images, several points have been covered by the same pixel. In this case, all points in the immediate neighborhood of the point just classified can also receive the respective semantic class to ensure all points will receive semantic information.

Processing the data results in a 3D point cloud with semantic information attached to each point. This data can then be used for the previously described use cases.

Simple postprocessing steps based on plausibility checks can further improve the results of the classification. For example, axle counters must lie close to the track, and rails of a track always run parallel with a fixed, previously known distance between them. Such conditions can be checked for after the classification. For example, objects identified as axle counters but not located right next to a track could be discarded and, e. g., classified as “other”.

3.4 Machine Learning on Spatial Data

A significant property of 3D point clouds is their three-dimensionality. It is precisely the arrangement of the measurement points in three dimensions, and thus the direct mapping of reality, distinguishing them as a form of representation. Machine learning requires many resources to compute the models that are used to make statements about new data. As hardware has become more powerful, these models have also become more advanced and perform ever-increasing amounts of computation (Xie et al., 2020). While early implementations could only work with relatively small amounts of data and a limited number of attributes, today, new neural networks are being developed with continuously increasing throughput. This development gave rise to the idea of no longer examining 3D point clouds via intermediate representations such as images but instead using spatial

data directly. In terms of the basic idea, this form of machine learning on 3D point clouds returns to the original geometric analysis because positions and shapes once again are becoming more important compared to an image-based semantic classification.

An image-based semantic classification has the disadvantage of the scene in an image being visible from only one perspective. Therefore, a single image can typically only classify objects with high accuracy that do not lose critical information when projected in two dimensions. For that reason, it works for objects such as lane markings and utility hole covers when taken from a predetermined perspective. For more complex objects, a series of images from different viewing angles is required to obtain good semantic classification (Boulch, Guerry, et al., 2018). As a result, image analysis speed is partially lost again as the number of images to be processed increases. This makes it interesting to use more complex neural networks for three-dimensional processing, where viewing angles no longer matter, requiring a scene to be analyzed only once.

The fundamental problem in processing 3D point clouds with machine learning approaches is that points in a 3D point cloud have no intrinsic order. In contrast, the structure and order of pixels in an image are fixed. It is known in advance which pixels are to be considered as neighbors, and the neighborhood has a significant meaning for image filtering operations and image analysis. For a 3D point cloud, transformations such as translation, rotation, and scaling change a 3D point cloud's visual appearance, but the semantic classes it contains are unaffected. Similarly, a 3D point cloud is still the same when the individual points are randomly sorted, whereas an image with randomly re-sorted pixels has lost its original meaning. Consequently, permutation invariant functions must be used in neural networks for the semantic classification of 3D point clouds.

Pioneers in the semantic classification of 3D point clouds over their structure using 3D CNNs were Maturana and Scherer (2015). Here, VoxNet is a volumetric CNN in which the 3D point cloud is transformed into voxels. Precisely, the points are placed into a voxel grid ("occupancy grid") whose voxels are subsequently either filled or empty. 3D convolutions can now be performed on these voxels. The semantic class is derived via the neighborhood in the occupancy grid. Maturana and Scherer write: "While it is conceptually simple to extend the basic approach [of CNNs] to volumetric data, it is not obvious which architectures and data representations, if any, will yield good performance. Moreover, volumetric representations can easily become computationally intractable."

The voxel grid's use requires a lot of memory relative to the actual data it contains because typical datasets leave many voxels empty. Therefore, the resolution cannot be arbitrarily large. Besides, as with image-based classification, the classification results must be subsequently transferred back to the 3D point cloud, another additional cost. Although voxel-based approaches continue to be pursued (Huang and You, 2016; Qi, Su, Nießner, et al., 2016; Poux and Billen, 2019), the focus is now primarily on other approaches that do not use an intermediate volumetric form.

A new form of semantic classification of 3D point clouds brought *PointNet* (Qi, Su, Mo, et al., 2017). In this approach, no additional abstraction form is used; the points are processed directly. Each point is initially represented only by its x, y, and z coordinates.

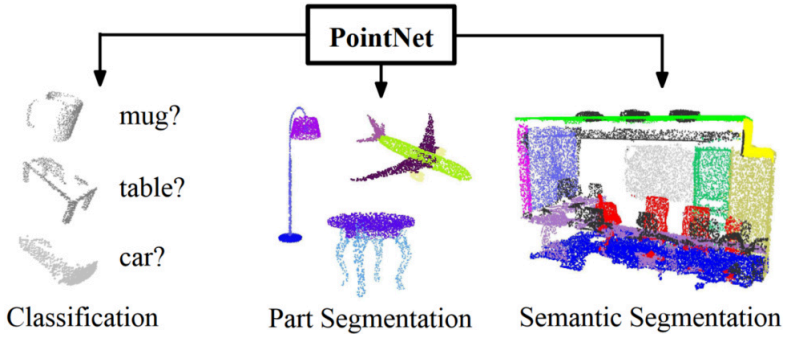


Figure 3.16: *Three components of PointNet. Figure from Qi, Su, Mo, et al. (2017).*

Further attributes like the normal vector and other values such as those described in Section 2.4 can be added additionally.

Strictly speaking, PointNet consists of three components, as shown in Figure 3.16. The first component classifies 3D point clouds representing individual objects ("Classification"). The second component segments individual objects into their constituent parts ("Part Segmentation"). The last component, which is a combination of the other two, is the one of interest in this work's context, as it can semantically classify complex scenes ("Semantic Segmentation").

Qi et al. summarize that "Key to our approach is the use of a single symmetric function, max pooling." Max pooling achieves the invariance of permutations mentioned above which is required for 3D point clouds. PointNet uses max pooling to select relevant representatives from the points that represent the rough structure of objects. On the one hand, the analysis of representatives dramatically reduces the required processing time. On the other hand, the network is robust against outliers and slight variations in the acquired data.

However, a problem with PointNet is that information about the point environment and local structures is lost, which can be important for classification. This problem is addressed in the more advanced *PointNet++*.

Qi, Yi, et al. (2017) describe how PointNet++ can be used to improve the classification result. They partition the 3D point cloud into local groups, in which local features are first determined. These are hierarchically and recursively combined into larger and larger groups, whereby higher-level features are determined, as usual for a CNN. As a result, the environment is also included in the classification, which significantly improves the results. On the ScanNet dataset, the authors achieve an accuracy of 0.74 with PointNet and 0.85 with PointNet++. Each step ("feature learner") corresponds to an application of PointNet, in which the most representative points are determined as before. Thus, the repeated application of the PointNet layers significantly increases the overall runtime of PointNet++ compared to PointNet.

Even one step further goes *DGCNN* (Dynamic Graph CNN, also called *EdgeConv*) by Y. Wang et al. (2019). In this network, point features are created that "describe the relationship between a point and its neighbors". Local structures are analyzed by creating

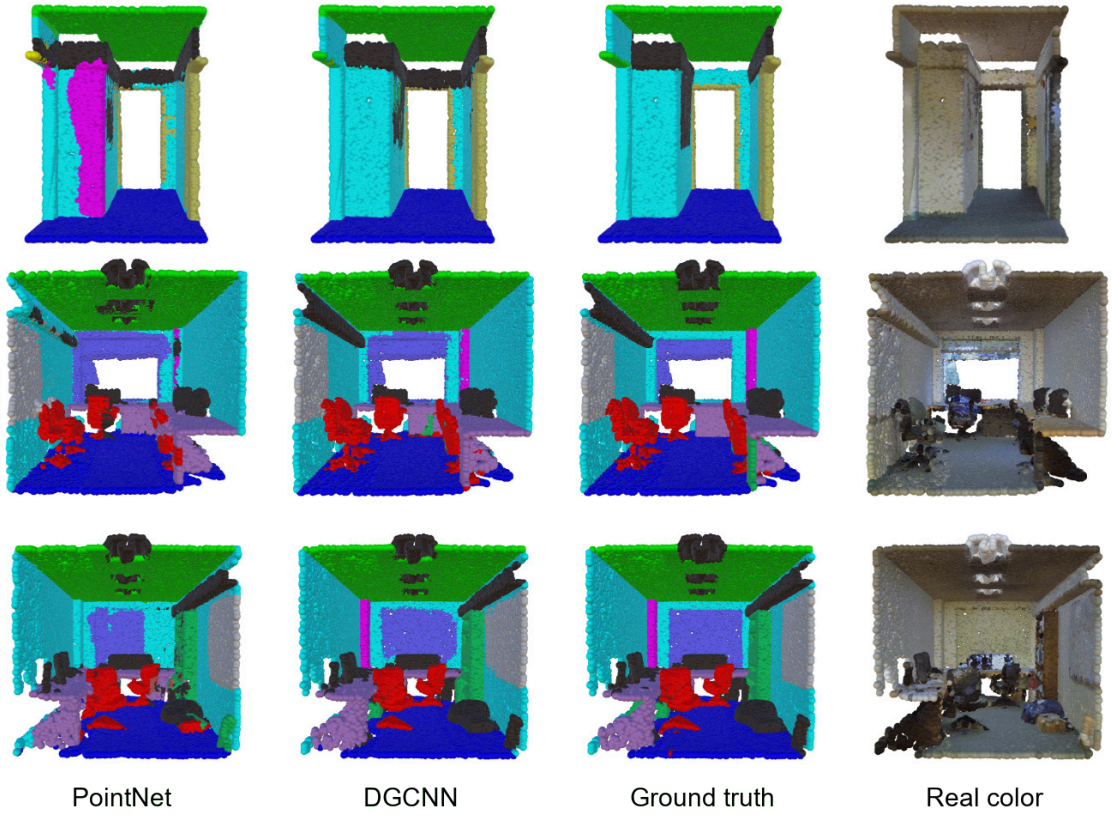


Figure 3.17: *Semantic classification results. Each color represents a semantic class. Adapted from Y. Wang et al. (2019).*

a k-neighborhood graph. Convolutions are applied to the edges in this graph which are connecting neighboring points in the embeddings. The graph is updated for each layer of the network, and the k-nearest neighbors in feature space are newly computed. DGCNN considers both local and global geometry information. It regards a point in connection to its neighbors in the current layer's feature space, which are points with similar characteristics. This contrasts with PointNet++, which only looks at the distance between fixed point coordinates. Figure 3.17 shows semantic classification results of PointNet and DGCNN in comparison.

Because DGCNN computes the k-nearest neighborhood multiple times, it has a long runtime. However, the information about local structures is beneficial for the semantic classification process. Thus, for the 3D point cloud framework used in this thesis, a combination of DGCNN and PointNet was developed, combining the advantages of both. This hybrid network, called *PCNN* (Point Cloud Neural Network), uses just one EdgeConv layer and therefore computes much faster than a complete DGCNN implementation. Before PointNet's max pooling operation is applied, the EdgeConv layer adds neighborhood information advantageous for the following analysis.

Figure 3.18 shows the architecture of PCNN. The EdgeConv layer is here used with

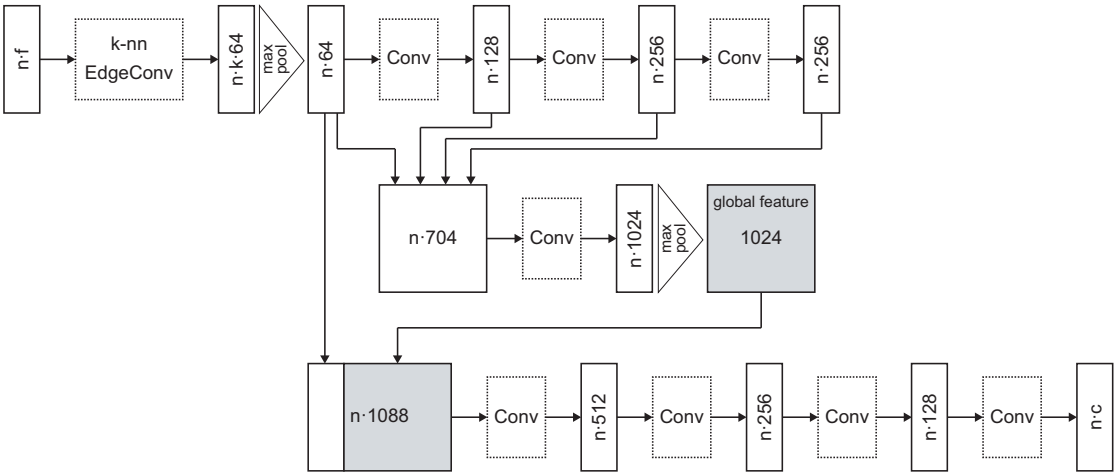


Figure 3.18: *Architecture of PCNN. n: points, f: features, k: neighbors, c: semantic classes.*

$k = 20$. The first four convolutions' outputs are used as a shortcut input for the fifth convolution, which outputs PointNet's global feature vector. This vector is appended to EdgeConv's output. Four additional convolutions follow, and a final softmax function computes the class label for each point in the result.

In addition to the x, y, and z coordinates, various other attributes can be used as input for semantic classification. Sections 2.3 and 2.4 have already presented values that can be precomputed for 3D point clouds and that can be used in further analysis. In the implementation used in this thesis, the values which are listed below are supported. For example, *intensity* enables the classification of road markings in otherwise flat surfaces. The detection of traffic signs also benefits from intensity, as the signs are highly reflective and thus stand out. Similarly, *RGB color values* can be entered if they are available from image matching acquisitions or colorization via panoramic images. For all points *normals* can also be calculated, whose x, y, and z coordinates are entered additionally. While this information can also be learned within the neural network, pre-computation of these characteristic values improves classification results. Similarly, the *curvature* can be used as additional information about edges and structures as input. Another value is the *local point density*, which can be used for classification. Last, a value for *height* above ground per point is possible as input. During semantic classification in the neural network, sampled points are scaled into the unit sphere for processing. Thus, this absolute information is lost and can contribute to the classification via the additional attribute. However, tests have shown that the classification result hardly depends on whether the information is available, so the height has no significant influence on the semantic classification.

The classification approaches presented in this chapter serve as a foundation for the use cases presented in Chapter 4. The implementation is highlighted in Chapter 7, and in Chapter 8, the classification results of the geometric approach are compared with those of PCNN.

Chapter 4

Applications of Semantically Classified Mobile Mapping Data

The presented approaches for preprocessing and classification enable a wide range of applications for mobile mapping data. For example, detected objects can be used for road cadastre creation or renewal (Caroti et al., 2005), clearance area checks (Mikrut et al., 2016), and 3D modeling (Vosselman, 2003). Interactive visualization tools allow users to explore and analyze cadastral data combined with 3D point clouds (Aringer and Roschlaub, 2014).

Navigation systems belong to the most prominent examples of geodata applications in everyday life. They provide maps with information, e. g., about roads, the number of lanes, and which lane must be used to turn into a particular direction (Bétaille and Toledo-Moreo, 2010). Autonomously driving cars must continuously detect road markings on the pavement around them to keep the car in its lane. They use a base map with lane information for anticipating the course (Maurer et al., 2016). Mobile mapping data is the key data source used for navigation system databases.

In this chapter, Section 4.1 explains how semantically enriched geospatial data can be exported in different formats. Using traffic signs as an example, Section 4.2 explains how existing map data can be verified and updated using semantically classified 3D point clouds. This chapter is partially based on the author's publications in J. Wolf, Richter, Discher, et al. (2019), J. Wolf, Richter, and Döllner (2020), and J. Wolf, Pietz, et al. (2021).

4.1 Data Export

This section presents examples for geospatial data export. Subsection 4.1.1 shows how vector-based polygons can be created by the example of road markings. Outlines, object positions, and dimensions are then available as map data, which can be further used in GIS applications. Subsection 4.1.2 explains how data can be exported as graph data by the example of a track network map with track and switch positions. Finally, Subsection 4.1.3 presents several statistical analyses that can be used for data exploration. The data export aims to provide an aggregated, generalized and quality-assured output for further work with the data. For this purpose, the classified 3D point clouds cannot be mapped unfiltered but require in-depth processing to avoid erroneous results.

4.1.1 Vector Data

Polygonal vector-based geospatial data is a standard format supported by GIS tools. ESRI shape files are the de facto standard used in this area (ESRI, 1998). Shape files are used for exporting single-point location data as well as lines and polygonal areas. Each of these entries is called a *shape*; multiple shapes can be aggregated into *shape groups*. Each shape can have arbitrary attribute-value pairs, describing additional information available for this specific shape. Resulting files usually have the endings `.shp` for the vector data and `.dbf` for attached data attributes and can be used in various GIS applications for subsequent tasks.

The use case described here is creating polygon vector data by the example of road marking shape files. The information on road marking position and their type can be used to create a detailed road cadastre. For example, this is used in traffic planning, for surveying the current road situation, checking the reconstruction of the road surface after construction work, and deriving a base map for autonomous driving vehicles (Hata and D. Wolf, 2014; Prochazka et al., 2019).

First, road markings must be detected and semantically classified as described in Section 3.3. Depending on the type of marking, one of three methods is applied to derive suitable shapes for the marking, which can be combined in a shape file for export.

Arrows on the road are used to show which lane must be used for which direction at a crossing. They are more complex markings than the various rectangular lines but have a unique form. Only a small number of arrows are used in almost all situations, namely those pointing left, straight, right, and any combination of those. Additionally, within a dataset of a particular region, the arrows have a standardized form. For this reason, templates of arrow shapes can be used to position a proper shape easily. These templates are placed in the position where an arrow marking was detected. The orientation is determined as best-fit so that they cover the largest number of points of the detected road marking. It would also be possible to calculate the lane's direction and use this for the arrow's orientation. Using templates results in clearly shaped arrows in the export file.

Rectangular lines are the most common type of road markings. They are used for, e. g., roadside markings, lane dividers, stop lines, pedestrian crossings, and cycle tracks. When creating convex hulls for rectangular road markings, these often have rough edges, resulting in a noisy visualization. Therefore, a better approach is the representation via correctly oriented rectangles concerning the width, height, and orientation of the road marking.

All connected points of the same semantic class are analyzed together as one line marking. A line's orientation is calculated by a *Principal Component Analysis* (PCA) on all points of the line, resulting in a vector describing the main direction of the points in the cluster of this line. An *Object-Oriented Bounding Box* (OOBB) with this main direction for the longer sides is drawn around the points. The resulting rectangle can be used for the shape of this road marking.

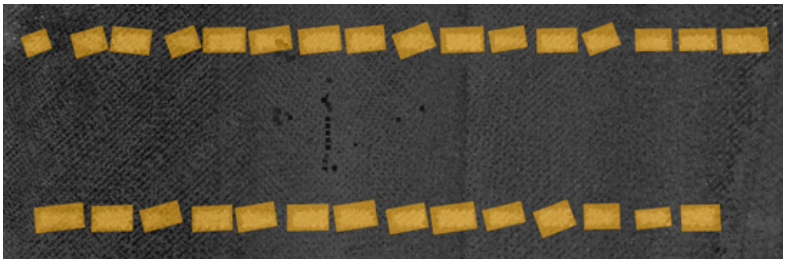


Figure 4.1: *Example for generated polygons (orange) using individually oriented rectangles for line markings placed on top of the 3D point cloud.*

An error value is calculated for each of the generated rectangles, describing which percentage of the area covered by this rectangle is not located on top of detected points of the road marking. Should this value get too high, the rectangle does not fit, which might occur if parts of the marking are missing, or several lines are merging. In this case, an outline will be computed as described for intersecting lines below.

Figure 4.1 shows that the orientation of the lines can be hard to determine, especially in situations where lines are not fully visible due to, e. g., abrasion.

The approach works well for single marks but is problematic for longer linework of successive dashed lines. The orientation and thickness of the generated rectangles deviate slightly from each other, resulting in an overall unclean image. Such line strings can be considered a cluster up to a certain extent, for which a uniform direction is determined that is applied to all generated rectangles in it to remedy this.

Lines from the same type are collected into a combined cluster if the distance to the following neighboring line is smaller than a given threshold, and the orientation only differs by a small amount. Watching the orientation prevents taking lines at corners that are oriented perpendicular to each other into the same cluster. For each of these larger clusters, the orientation can be determined with another PCA on the points of all road markings that are part of this cluster. All the rectangles in the cluster will then be oriented in the same calculated direction. The width of all rectangles for the cluster can be harmonized by using an average width or using a predefined width for standardized line types where this information is available.

The most complex markings are **barred areas and intersecting lines** and all other non-regular markings. Here, the shapes must be constructed differently. When the rectangle fitting described above does not fit a marking or rectangles or arrows cannot represent the marking's semantic type, the following approach is used to generate a proper shape.

A two-dimensional grid with squared cells with a side length of about 5 cm is created. The points of the respective marking are placed into this grid. After all points have been added, each cell either contains points or not. Those cells containing points define the area that the created shape should span. The outline is generated by iterating over all outer cells at the border of the marking. For each cell, the outermost point will be used

as a vertex for the generated polygon, resulting in a shape fitting closely to the detected road marking. Holes are cut from the area in regions with empty cells in the same way. Afterward, the Douglas-Peucker algorithm (Douglas and Peucker, 1973) is used for shape simplification.

Figure 4.2 shows a screenshot of a 3D point cloud source file and the shape file that was automatically derived via the described classification and shape creation processes.

When combining these three methods, map data with all road markings can thus be derived automatically from mobile mapping 3D point clouds of roads.

Incorrectly assigned semantic classes such as an incorrect line type and overall high intensity in wet areas of the road can affect the result. Section 8.2 evaluates the result of the automated vector data creation.

4.1.2 Graph Data

For railroad track environments, a different use case is investigated. Here, a track network can be derived as a graph in which all tracks are recorded as edges and switches as nodes. These can be placed georeferenced or summarized as an overview map with shortened distances.

This information can serve as base information for a track cadastre and provide the basis for maps for archiving and maintaining object locations. Many data at railroad companies are not yet available as digital models for the entire service area. For example, signal and balise locations relative to other objects are known but not necessarily precise coordinates and complete track network maps.

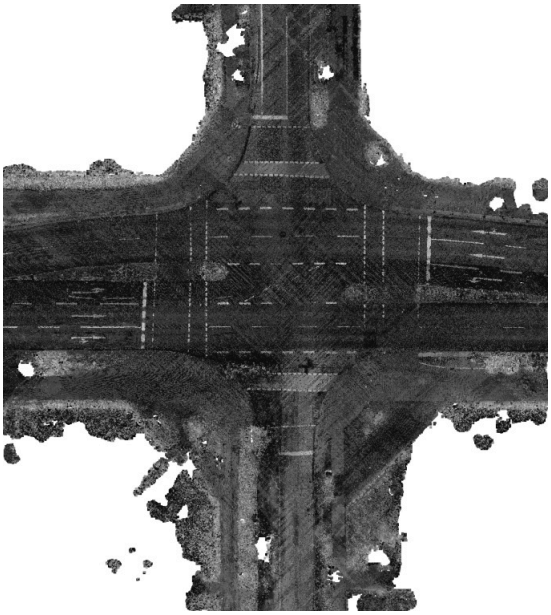
In a 3D point cloud, all rails can first be classified using one of the approaches described in Chapter 3. From this, information about the tracks and locations as well as types of switches can be derived.

Once coordinates have been determined for these within the 3D point cloud, a graph can be generated to record the infrastructure. All switches are created as spatially located nodes with precise coordinates and the switch type as an attribute. The track sections between them are represented as edges, for the length can be kept as an attribute.

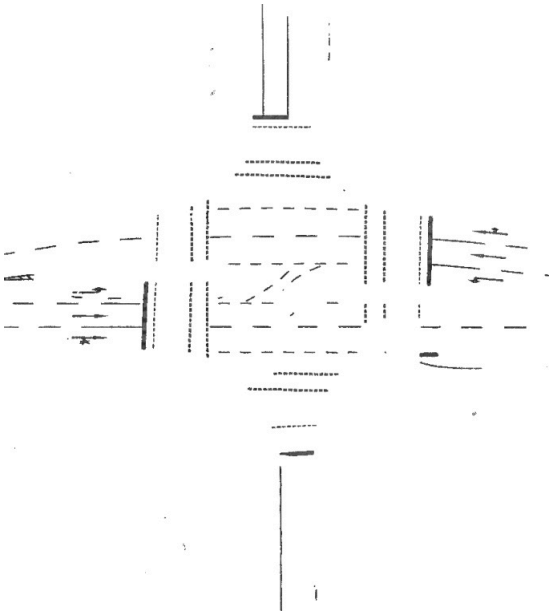
Already existing plans can be checked for correctness. For example, coordinates of objects in existing maps can be checked by analyzing the 3D point cloud to determine whether an object of the appropriate type is present at this location. If this is the case, the information can be recorded as verified; otherwise, matching structures are searched for in the immediate vicinity. For example, it can be determined if the object's actual location deviates from the recorded position by a few meters, and this can be suggested as a correction. The process is described in Section 4.2.

4.1.3 Statistics Computation and Further Analyses

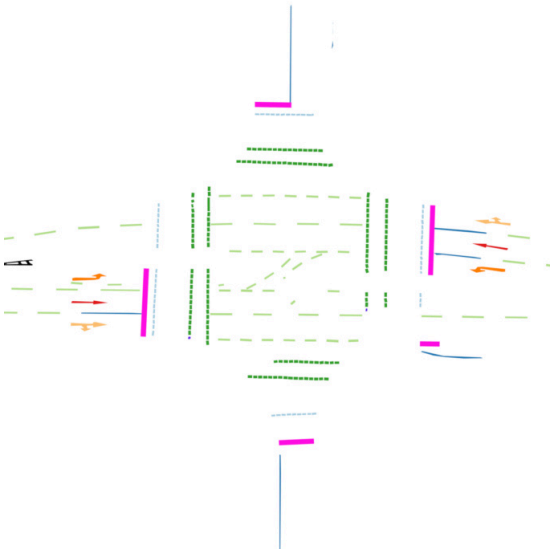
A combination of previously describes techniques enables exporting comparisons and statistical data after analyzing 3D point cloud scans and using 3D point cloud data for



(a) *3D point cloud of a road crossing. Points are colored by intensity values.*



(b) *3D point cloud filtered for road markings.*



(c) *Classification result with shapes for individual road markings. Semantic classes are represented by distinct colors.*

Figure 4.2: *Visualization of a 3D point cloud from the Hamburg dataset, the filtered road marking points, and the derived road marking shape file.*

predictive measures and simulations.

For example, differences between point clouds from two acquisition times can be highlighted. If a semantic classification of two mobile mapping scans of different points in time is performed, classes and objects detected in them can be compared with each other. For example, it can be determined whether traffic signs that were detected in a previous scan are no longer present. Similarly, it can be determined whether traffic signs or road markings have been soiled, rubbed off, or pasted over, causing their reflectance to deteriorate from one scan to the next. Even newly planted or removed street trees can be easily identified in two scans of the same street. The data from different acquisition times can also be used to detect occlusions in the data and create a composite data set in which as few objects as possible are occluded by, for example, vehicles.

Statistical data can be determined for the analyzed objects. For example, how many trees are along a roadway, how far they are from the road or track, and their approximate height. From this, it is possible to determine a storm hazard for tall trees close to roads and tracks.

It can be determined how many posts, traffic lights, signs, signals, or utility hole covers are on a route, allowing users to estimate construction or replacement costs. Similarly, the total area of road markings can determine the amount of paint required for renewal.

The surrounding vegetation is of particular importance in road and rail areas. For safe and trouble-free operation, a clearance area must be permanently ensured, i. e., branches and twigs must not protrude into the area where trucks or trains operate or come too close to power lines. Figure 4.3 shows a visualization of a clearance area profile placed into a 3D point cloud. By analyzing mobile mapping scans in these environments, it is possible to determine how far the vegetation is from the clearance area in each case (Toyama et al., 2019). In particular, if multiple measurements from different times are available, predictions can be made as to when pruning will be necessary because the vegetation has grown too much.

All the information obtained on the proximity of existing trees, the condition of the vegetation, the wear of signs and markings, or the presence of potholes can be used in large-scale surveys to calculate a kind of “health score” for specific sections of road or rail. This score can be used to decide where maintenance is most urgently required and where available resources can be most effectively deployed.

Another practical use case is investigating whether a route for oversize loads is feasible. Many thousands of oversize load transports per year must be planned in large cities. In particular, the available width of lanes and the radii of curves play a significant role. Mobile mapping scans can be used to simulate the travel of oversize load transports for a defined route and check for possible bottlenecks and collisions. In the case of problematic locations, it is possible to go into detail as to whether a bridge is in the way and passage is therefore impossible, or whether it is just an overhanging branch of a tree that could be removed for critical transports. The precise measurement data and available detailed information on the type of objects can be used to determine whether there are signs or lanterns in the way that could be dismantled and which other objects

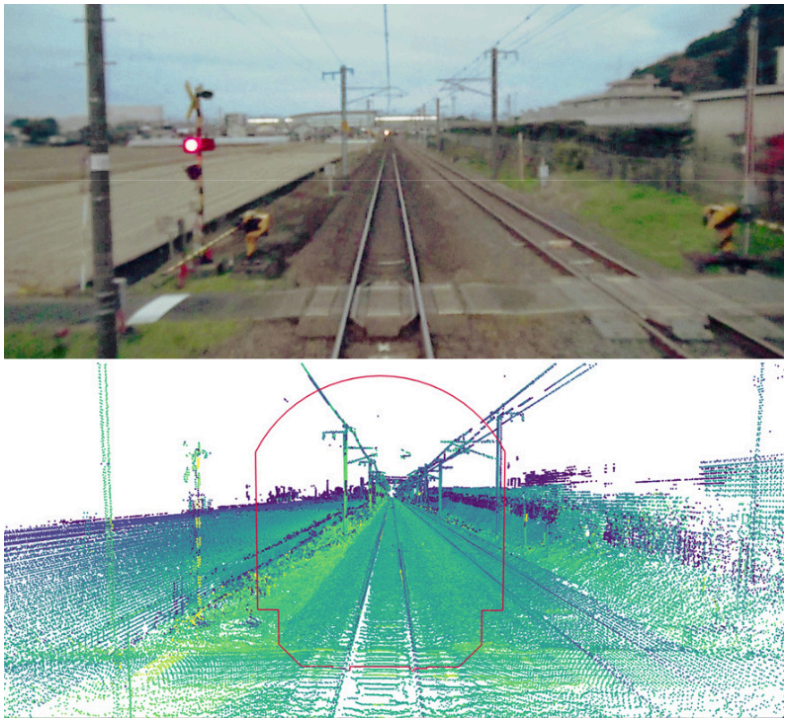


Figure 4.3: *Visualization of a clearance area profile in a 3D point cloud. Figure from Toyama et al. (2019).*

require special attention for transport. Automated analysis of the requested route can save manual inspection time if problematic locations can be automatically detected and displayed in a summarizing report to the processing personnel.

In summary, semantically classified 3D point clouds enable numerous automated analysis methods. Traffic planning, construction measures, and maintenance work can be efficiently supported if the data is processed for the individual use case.

4.2 Data Verification

An everyday use case for geospatial data analysis recognizes specific objects in 3D point clouds for automated verification of existing datasets. For example, existing map data can be matched with a 3D point cloud scan to verify or adjust object locations. This is useful, for example, to semi-automatically verify the condition of traffic signs in a city's streetscape. If the traffic signs are available on a map, current 3D point cloud scans can be used to compare the map's status.

Data from a map with locations of traffic signs can be combined with the 3D point cloud from a mobile mapping scan to compare the information. All signs in the map are compared with corresponding structures at this location in the 3D point cloud. Conceptually, this is realized using a continuously increasing search radius to determine the search point's closest sign. The detailed information of an already classified 3D point

cloud can be used, or characteristic properties for the searched object are analyzed, like a reflecting surface sitting on a post structure. In addition to the position, the type of sign can also be checked if the information is available. Matching attributes in the datasets can be noted as confirmed.

If the matching sign cannot be found in the 3D point cloud at the position noted in the map, the nearby surroundings are searched. The map's position data is automatically adjusted if the corresponding object is detected only a few centimeters away.

If the object cannot be determined in the vicinity, it is removed from the map and transferred to a separate list in which all objects are collected that could not be detected in the 3D point cloud.

Likewise, non-matching sign types are noted so that they can be adjusted in a manual revision. In LiDAR 3D point clouds, not all traffic signs can be identified by type. Since no color information is available, the type's recognition is based solely on the shape and intensity values. Especially older or dirty signs are therefore often not well recognizable. Section 6.3 addresses an approach to classify traffic signs when color information from panoramic images is available. In this case, information about the type of signs can also be adjusted automatically.

A further step creates signs in the map data that have been determined in the 3D point cloud but have not yet been recorded. These receive the entry that they were added automatically as a review note.

In the end, there are four groups of signs:

- Signs that match in the map data and the 3D point cloud. This is usually the most extensive group. All these signs are considered verified and do not need to be considered further manually.

- Signs whose position or type could be adjusted automatically. This was the case when the sign searched for was not found at the noted position but in the immediate vicinity, or the sign is most likely of a different type than indicated.

- Signs that cannot be found in the 3D point cloud scan. Signs recorded in the map may not be detected in the 3D point cloud. For example, this can happen because signs were only temporarily erected during construction work and are no longer there, because they have been damaged, or because other objects have obscured them in the 3D point cloud or they could not be correctly classified. These cases need to be checked manually.

- Signs found in the 3D point cloud but not recorded in the map data. These can be temporarily erected signs, new signs that have not been recorded so far, or other objects recognized as road signs in the 3D point cloud due to their structure.

Only some of these groups need manual postprocessing. Thus, the automated analysis greatly reduces the number of signs that must be processed manually and the time required for this work.

Chapter 5

Ground-Penetrating Radar Data

By their very nature, LiDAR scans can only capture objects in the 3D environment in the scanning system's line of sight. In road construction and maintenance, however, the subsoil is of immense importance in addition to the condition of the surface and applied markings. The subsoil impacts the stability of a road. For example, as soon as potholes or cavities develop beneath the road surface, they must be eliminated to maintain safe, long-term operation (Saarenketo and Scullion, 1994).

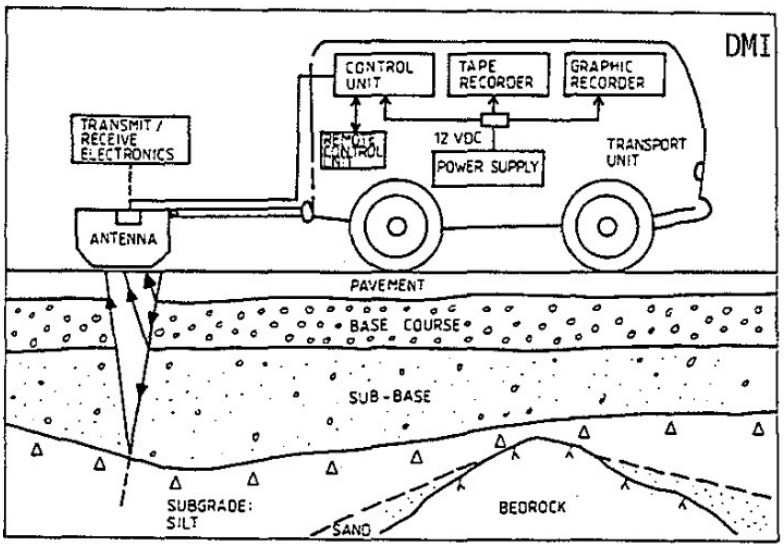


Figure 5.1: *Schematic representation of a GPR antenna and the transmitted and returned signals. Figure from Saarenketo and Scullion (1994).*

To widen the data base for digital twins of roads and the street infrastructure in general, *Ground-Penetrating Radar* (GPR) can be used as a technology to generate subsurface information beyond LiDAR 3D point clouds. GPR scanners can measure material properties several meters below ground, creating insights about the non-visible foundation of roads and pathways (Davis and Annan, 1989). Figure 5.1 shows how GPR signals are reflected from the different layers of the road. Using this data source significantly expands the analyzed area, and the acquired data can be used for even more use cases.

To illustrate the potential applications of integrating GPR data into 3D point clouds,

(a) *Four GPR antennas mounted on the back of a car.*

(b) *GPR hand cart. Figure from Geophysical Survey Systems (2018).*

Figure 5.2: *GPR measuring vehicles.*

the following scenario is considered: On the one hand, pipes and lines below the road can be detected in GPR scans. This enables the automated creation of a map with information about these structures' courses. Then, during construction or maintenance work, reliable information about the exact position is available and can be considered during planning. On the other hand, developing potholes and emerging cavities beneath the road can be detected. By comparing several captures taken at intervals, changes that indicate a weakening of the subsoil become visible. However, anomalies that may be of interest for maintenance can also be detected in a single scan. Information about these anomalies can be presented using appropriate exploration tools, allowing responsible road agencies to address problems as part of road maintenance before they have a negative impact on traffic.

GPR emits electromagnetic waves into the ground and receives the reflected signal from pavement and soil. The ground's structure impacts the propagation of the emitted signal. Thus, the returned signal provides information about the materials' conditions. Radar antennas for road inspection are usually mounted on scanning vehicles, as shown in Figure 5.2, which can drive along with uninterrupted traffic. Alternatively, they can be mounted onto hand carts. The vehicle shown in Figure 5.2(a) has four antennas that use two different frequencies for measurement in an alternating placement to better cover different depth ranges. By arranging them across the vehicle's width, the whole lane can be measured during the drive. The vehicle would have to travel along each lane once to cover the entire road's surface.

The antennas are hovering above the ground, and their distances to the surface slightly change while driving. A time zero correction is usually part of the sensor calibration to account for these changes (Yelf, 2004). After capturing, the radar scan data is usually analyzed in the form of B-scans. They are a consecutive sequence of individual measurements (A-scans), in this case, along the vehicle's driving path (Giannopoulos, 2005). The amplitudes of the radar signal recorded during the measurement are displayed as color values in a diagram. The amplitudes' height is indicated by the color intensity

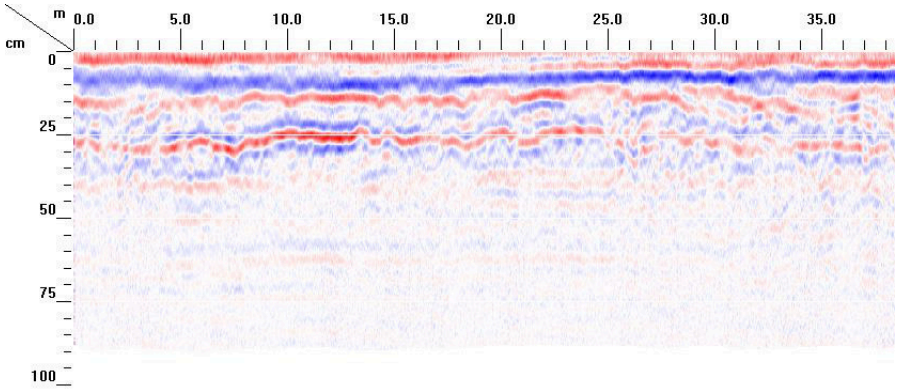


Figure 5.3: *Visualization of a GPR B-scan from the Essen dataset. Red represents positive amplitudes, blue negative. Color intensity represents the amplitudes' heights.*

and the direction by hue, usually red and blue. Figure 5.3 shows such a B-scan. The x-axis maps the traveled distance of the measuring vehicle, and the y-axis represents the depth at which the corresponding value was returned. The measured values are represented by colors as described.

Georeferencing is the process of "aligning geographic data to a known coordinate system so it can be viewed, queried, and analyzed with other geographic data" (Wade and Sommer, 2006). Mobile mapping vehicles use a Global Navigation Satellite System (GNSS) to track their current position. The information is stored "in the form of so-called trajectories, i. e., sequences of individual 2D or 3D points with time stamps" (Sester, 2020). Using this information and the knowledge about the fixed positions of LiDAR scanners and ground-penetrating radar scanners on the scanning vehicle, the different datasets can be collocated in exact spatial relation to each other.

The combination of above-ground 3D point clouds and below-ground 2D radar scans enables more extensive analysis of road environments by using two combined sources instead of evaluating each on their own. An everyday use case for ground-penetrating radar data inspection is to detect specific areas with an increased chance of developing potholes (Huston et al., 2000). By adding road surface information from the 3D point cloud, false positives like utility holes can easily be distinguished from other anomalies in the road's subsoil.

This chapter addresses requirements for the evaluation of GPR datasets in combination with 3D point clouds. It shows visualization concepts for the manual exploration and evaluation of corresponding data and procedures for automated analyses. It presents a tool developed for GPR data analysis and the combined visualization with 3D point cloud data.

This chapter is partially based on the author's publications in J. Wolf, Discher, Masopust, et al. (2018) and J. Wolf, Discher, and Döllner (2019).

5.1 Related Work

Benedetto et al. (2017) give an overview of how ground-penetrating radar can be used for road inspections. They discuss in detail which processing techniques can be used to analyze the pavement condition. Evans et al. (2008) summarize the abilities of ground-penetrating radar for general pavement investigations. Saarenketo and Scullion (1994) explicitly list localization of sinkholes and crack growth monitoring in their report about ground-penetrating radar applications on roads and highways. They further describe possible soil and road structure evaluations and required data interpretation techniques (Saarenketo and Scullion, 2000). Giannopoulos (2005) describes how ground-penetrating radar data can be visualized. In addition to two-dimensional profiles, they show an example for a three-dimensional visualization by rendering the three principal planes of a cuboid holding the data. Usually, two-dimensional ground-penetrating radar profiles are shown individually, but multiple scans can be placed next to each other to create a spatial feeling for the data (*Geo Radar: 3D and GPR* 2005).

Eitel et al. (2016) describe the relevance of 3D point clouds for many geospatial applications. Puente et al. (2013) review available mobile terrestrial laser scanning systems, showing that LiDAR is widely used on various mobile platforms for data acquisition leading to users being enabled to "experience and work directly with real-world conditions by viewing and manipulating rich point clouds". Biasion et al. (2005) describe mobile laser scanning applications for environment analysis during disaster management. Several authors discuss the automated analysis of 3D point clouds: Per-point surface category information can be derived by analyzing a 3D point cloud's topology (D. Chen et al., 2017) and applying deep learning concepts (Boulch, Saux, et al., 2017). In turn, derived surface category information can be used to reconstruct three-dimensional models of specific buildings or infrastructure assets as shown by Teizer et al. (2005).

A general overview of rendering techniques for 3D point clouds is presented by Gross and Pfister (2011). While *photorealistic* approaches (Schütz and Wimmer, 2015a; Preiner et al., 2012) aim to reduce typical artifacts (e. g., visual clutter or a holey surface representation) by rendering points with an appropriate size and orientation, *non-photorealistic* techniques (Simons et al., 2014; L. Zhang et al., 2014) deal with the inherent fuzziness of a 3D point cloud by emphasizing edges and structures. All those rendering techniques can be seamlessly integrated into one rendering system, as here demonstrated with the *eye dome lighting* technique initially introduced by Boucheny (Boucheny, 2009).

Focus+context visualization is used here to combine 3D point clouds and GPR in a combined visualization. Focus+context visualization techniques have been widely discussed in the context of mesh-based models (Vaaraniemi et al., 2013; Elmqvist and Tsigas, 2008), ranging from static visibility masks (Sigg et al., 2012) to interactive lenses (Trapp et al., 2008). Discher et al. (2017) apply such techniques to 3D point cloud depictions, enabling users to highlight task-relevant but occluded objects and structures. Similar to the approach presented here, they apply multi-pass rendering based on G-buffers; however, they do not combine 3D point clouds and additional

geodata. State-of-the-art out-of-core rendering concepts decouple rendering efforts from data management to render 3D point clouds of any size. Exemplary systems (Martinez-Rubi et al., 2016; Goswami et al., 2013; Rusinkiewicz and Levoy, 2000) subdivide 3D point clouds into small, representative subsets suitable for real-time rendering. Recent approaches combine out-of-core and web-based rendering concepts, enabling a ubiquitous visualization on a diverse range of client devices (Discher et al., 2018; Schütz and Wimmer, 2015b; Butler et al., 2014).

5.2 Visualization Concepts

Based on GPR and GNSS data characteristics, the following requirements need to be addressed for a combined visualization of 3D point clouds and GPR data:

R1 Correct positioning of GPR data and 3D point clouds into a homogeneous spatial reference system.

R2 Occlusion-free visualization of individual GPR B-scans within a GPR dataset.

R3 Visual filtering and highlighting techniques to enable a focused inspection of areas of interest that can be defined at runtime.

The combined visualization of 3D point clouds and GPR data is based on two major user interface components: A *3D scene view* and a *2D user interface* further explained in the following sections.

5.2.1 3D Scene View

The first step towards integrating GPR data and 3D point clouds into a single visualization is projecting each B-scan onto the captured GNSS trajectory (**R1**). Individual B-scans can be hidden dynamically (**R2**) to prevent different B-scans from occluding each other.

Each GPR dataset is represented by a cuboid-like structure, covering the amount of space scanned by the GPR sensors that is rendered onto the GNSS trajectory (Figure 5.4). Their values are interpolated to fill the area between the B-scans.

A 3D-texturing approach guarantees the possibility of slicing the cuboid both vertically and horizontally and moving it along the trajectory. Thus, the cuboid can be restricted to specific areas of interest, thus, facilitating visual filtering and highlighting (**R3**).

Visibility and usability of the cuboids are increased by raising them onto the GNSS trajectory instead of leaving them below ground level (**R2**). The points initially located above the cuboid are translated alongside to keep the spatial context and highlighted for better contrast to non-translated points (Figure 5.5). Furthermore, to enable a direct view onto the cuboid's surface, an interactive lens is provided that hides points around the cursor. The visualization can be switched to instead only show the points above the cuboid around the cursor for focusing on the GPR data while keeping once more the spatial context (**R3**).

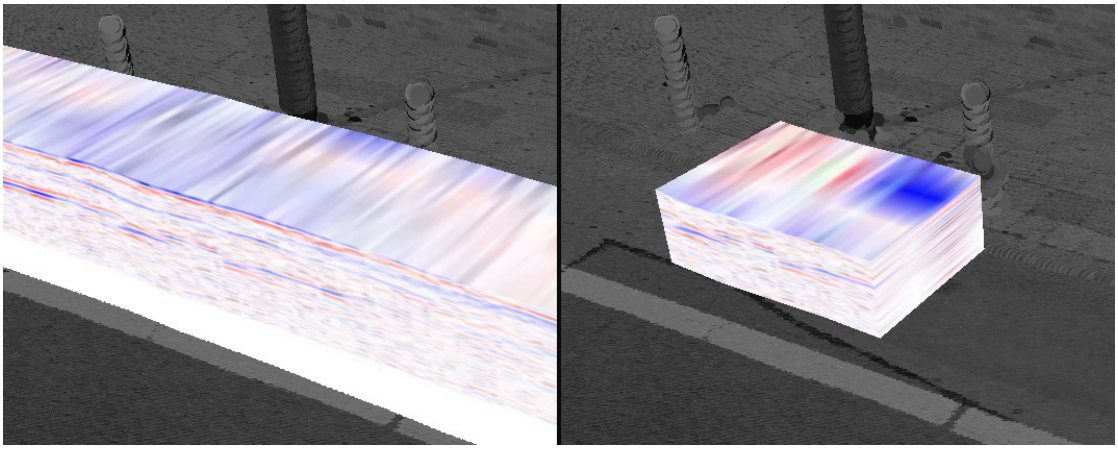


Figure 5.4: *Cuboid of four parallel interpolated GPR scans rendered onto the GNSS trajectory (left) from the Essen dataset. Vertical and horizontal slicing can be used to explore data inside the cuboid (right).*

5.2.2 2D User Interface

A supplementary widget is provided, visualizing all B-scans in total length in 2D (Figure 5.6) and enabling configuration of the 3D scene view, facilitating an in-depth exploration of the GPR data.

First, users can change how the cuboid of the given GPR dataset is rendered. Configuration includes (1) setting its elevation relative to ground level, (2) cropping to specific start and end points, (3) cropping to a specific minimum and maximum radar scanning heights, and (4) hiding specific B-scans altogether. Cropping to specific start and end points enables users to move both the cuboid and the corresponding slice in the 2D view, back and forth. Doing so moves the camera position in the 3D view accordingly, ensuring that the view is always centered on the cuboid (**R3**).

Second, users can change how textures are generated from a GPR dataset. While the input data includes raw information about the reflected signal picked up by the receiver, the generated textures show the amplitudes of these values, with positive and negative values color-coded in red and blue. Users can change how much these values should be amplified since their range can vary greatly. For example, a larger amplification should be applied when exploring parts of the data with small differences. Furthermore, parts featuring drastic changes (i. e., most often points of interest) can be highlighted by specifying thresholds. As a consequence, these parameters facilitate identifying anomalies in specific regions of the GPR data.

5.3 Analysis Techniques

During the large-scale recording of ground radar data, large amounts of data are generated. Manual evaluation of the data would require considerable time. However, large parts

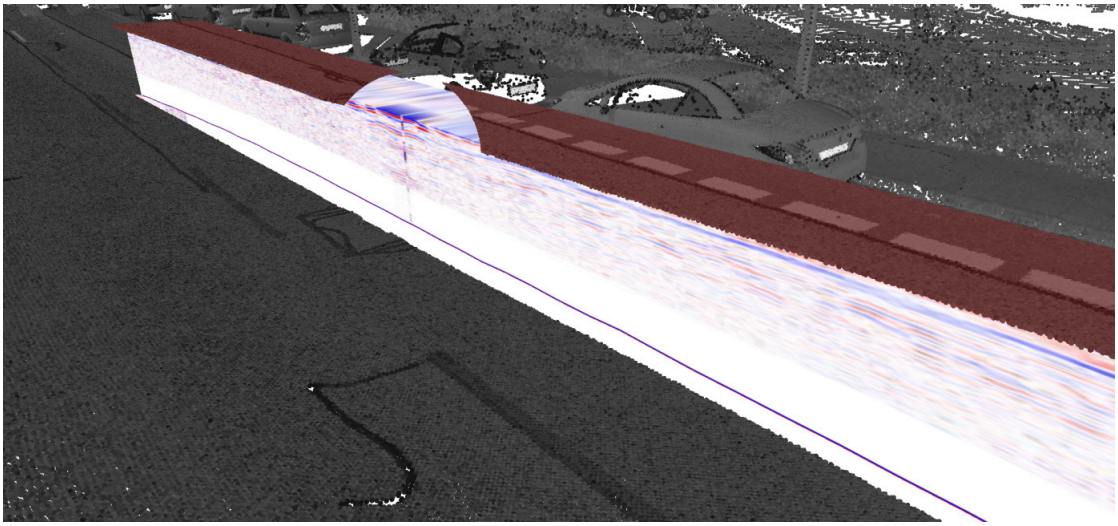


Figure 5.5: *Points originally under the cuboid are elevated and highlighted in red. An interactive lens shows the cuboid's surface below the points. Data from the Essen dataset.*

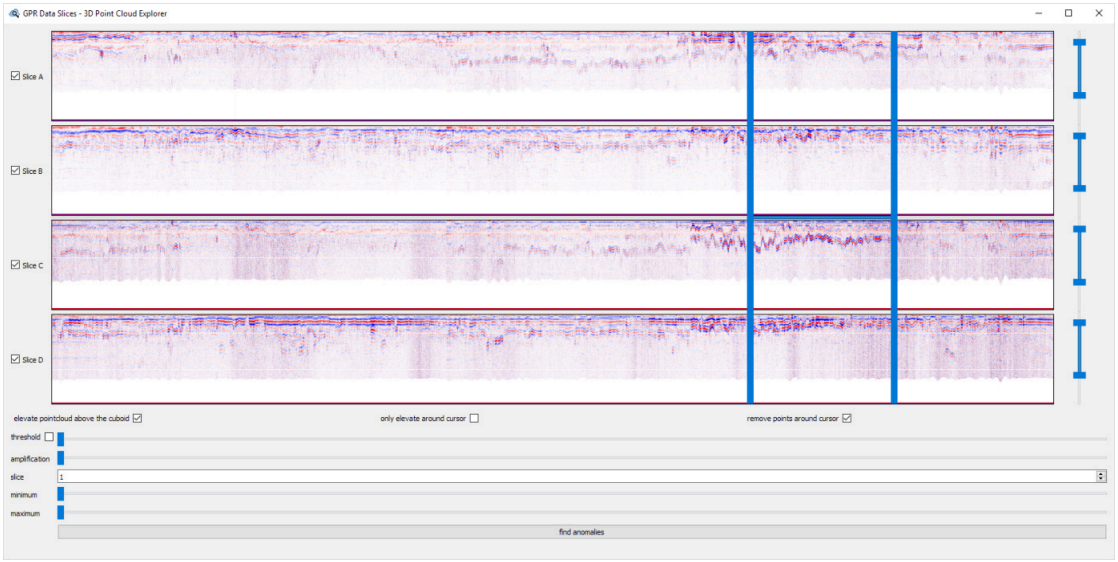


Figure 5.6: *2D User Interface for GPR data with cropping and thresholding options.*

of the data do not contain any information that is relevant in the further course of the evaluation. In particular, road layers that run uniformly over long distances are signs of good road condition, where no detailed assessment is required. On the other hand, all areas deviating from the normal condition are of interest. The ground-penetrating radar information can be automatically checked for anomalies to assist in the analysis of the data. These areas are highlighted for users. Only those areas where deviations exist need to be considered for subsequent manual analysis.

The tool described in this chapter provides a parameter-based anomaly detection

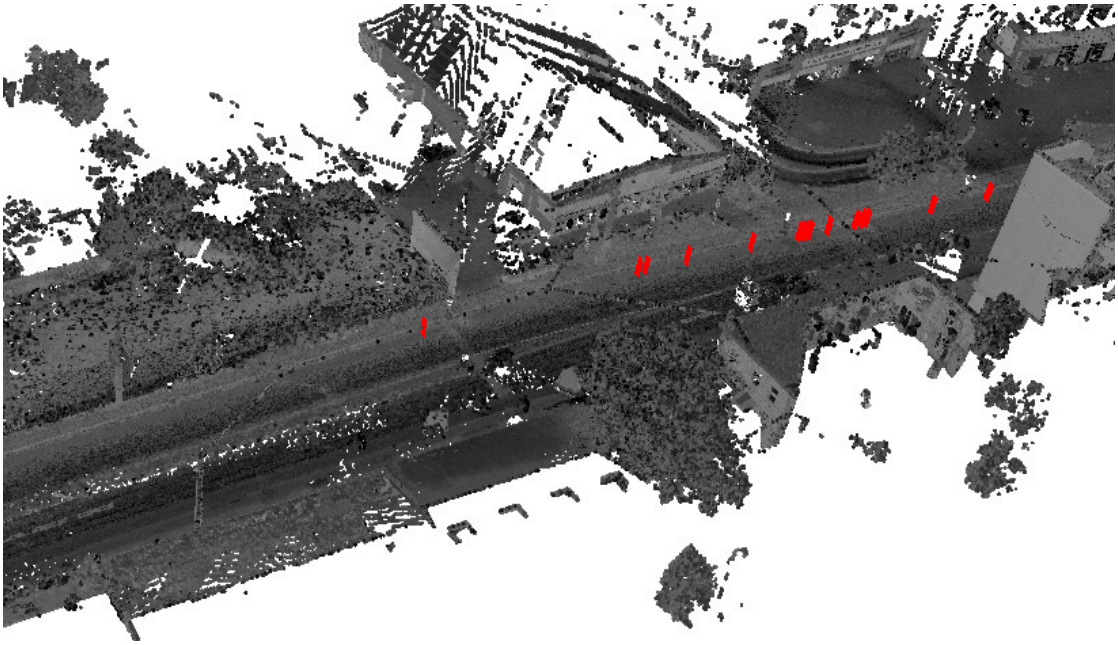


Figure 5.7: *Highlighted positions of detected anomalies in the 3D scene from the Essen dataset.*

working on the GPR textures. It highlights positions in the scan data in which specific areas in a predefined depth exceed given thresholds. The values can be configured in the UI, and the filtering results are displayed immediately for fast data exploration.

Detected anomalies are highlighted in the application both in the overview window and in the 3D scene, as shown in Figure 5.7. Users can jump to areas of interest within the dataset through this display.

Detecting relevant areas can also be automated. For this purpose, machine learning for image analysis is used. After converting the raw GPR data into textures, these can be interpreted as images. The representation of the measured values with two colors described above generates images in which the line course can be interpreted. In a prototypic implementation, a handwriting recognition network from scikit-learn was used (Pedregosa et al., 2011), which was also found to be suitable for this application. A sliding window moves over the images generated from the GPR's B-scans. In a test dataset, anomalous areas were marked. The network trained with this data can be applied to other datasets to detect areas with anomalies.

In a further step, anomalies detected in the road are merged with existing data from the semantic classification of a 3D point cloud of the environment. The 3D point cloud is examined for typical objects matching this formation at all locations detected as anomalous. In particular, streetcar rails and utility holes lead to a deviation in the road surface structure compared to areas with a smooth road surface. Both rails and utility hole covers can be detected in 3D point clouds using an automated semantic classification described in Chapter 3. Thereupon, the detected anomalies can be filtered,

and explainable deviations do not need to be further considered. Afterward, only areas of the roadway remain whose structure deviates from typical values, without this deviation being expected because of known objects. Filtering considerably reduces the number of sections of a GPR scan that must be checked manually, and significantly larger sections can be evaluated in the same amount of time as would be the case if all the data were examined manually.

5.4 Usability and Performance

The presented rendering system was implemented in C++ by extending the framework described in Chapter 7. The test data consisted of four GPR B-scans continuously captured in driving direction and a 3D point cloud from a mobile mapping LiDAR scan. The four B-scans were captured in parallel. Two antennas were measuring with a frequency of 2 000 MHz, the other two with 1 000 MHz. Their signals reached a depth of 0.45 m and 0.90 m, respectively. The scanners were mounted with a distance of 0.42 m to each other. Each scanner captured the 650 m of road data with 13 146 data points, holding 512 4-byte samples each.

The 3D point cloud is a combination of two LiDAR scans. The respective scanners were oriented to the left and the right, mounted on top of the measuring vehicle. Besides the three-dimensional coordinates, each measuring point holds information about the measured intensity, visualized as grayscale values. More than 1 000 points per square meter have been captured in the region covered by the ground-penetrating radar.

One problem in the visualization originates from the antennas' order: While the first and third antennas measured with 2 000 MHz, the second and fourth were using 1 000 MHz. While having these antennas in that order with different frequencies—and therefore different pickup patterns, maximum depths, and accuracy—covers the region with two different settings, it makes visualization more challenging since neighboring B-scans are not directly comparable anymore. Therefore, interpolating the 2D B-scans to create a 3D representation of the captured data might lead to unexpected results when B-scans with differing antenna settings are active. Since the B-scans are located close to each other, they also easily occlude each other. An essential part of the visualization is that the radar scan coordinates are precisely mapped to the 3D point cloud, so adjusting the B-scans' distances is not viable. Thus, the user can hide unwanted B-scans to make otherwise occluded B-scans visible.

The used rendering system generates interactive frame rates for arbitrary large 3D point clouds (Discher et al., 2017). The GPR data is supplied as raw data to the system before being loaded into OpenGL textures to generate the final rendered textures. Therefore, the performance cost is composed of the initial time to load the data and the time at runtime to update and draw the B-scan textures. Since the few textures in the test case have a resolution of $13\,146 \times 512$ pixels with a bit depth of 4 bytes for the raw data and one byte for result textures, and both updating and drawing of these textures are completely GPU-accelerated, this runtime cost is negligible in comparison to the one

introduced by managing and rendering the 3D point cloud. However, this was evaluated on a small dataset of 650 m captured road data. More advanced memory managing methods and level of detail approaches might be used for more extensive datasets to overcome challenges such as limited GPU memory.

Combined visualization of GPR datasets and 3D point clouds enables comparisons between both data sources and facilitates evaluations, e. g., in the context of road inspections. Anomalies in the GPR data can be compared with the 3D point cloud to detect irregularities visible from above ground. Utility hole covers and rails can easily be identified within the 3D point cloud, and their location can be considered when evaluating anomalies from the GPR data.

Cropping the GPR B-scans to a particular area of interest in length and height enables focusing on details in a small area and avoids occluding too much context information. Individual B-scans can be enabled or disabled for the visualization to further decrease the visual clutter of currently unneeded data, especially concerning B-scans scanned with different frequencies. As discussed, implementing level of detail approaches for the GPR data might improve handling larger datasets.

Visualizing B-scans slightly raised above the ground, and also raising the ground points from the 3D point cloud near them, results in a less occluded view onto the B-scans. The ground-penetrating radar cuboid's top plane can still be inspected by hiding those elevated points in a small region around the cursor.

The threshold and amplification manipulation assist while visually identifying anomalies in the GPR data. An extended automated analysis could help to detect anomalous regions in the data by highlighting them during the inspection, further facilitating the detection of areas of interest.

The visualization of B-scans from different GPR datasets in areas of intersections and for roads with multiple scanning runs holds potential for further development.

Chapter 6

Panoramic Images

Panoramic images are a combination of multiple images taken from the same viewpoint in different directions. Such images can be acquired by many mobile mapping vehicles in addition to LiDAR 3D point clouds (R. Li, 1997) and provide another source of information for object detection because they contain color information in contrast to the LiDAR data.

Panoramic image camera systems mounted on mobile mapping cars usually consist of at least five side-facing cameras and one upward-facing camera to cover the all-round view. No camera is pointed downward because the car obscures the ground. The images from the cameras can be merged at the edges as shown in Figure 6.1, creating a single panoramic image that depicts a 360-degree panoramic view, including the sky. The process of merging multiple images into one panoramic image is also known as *stitching*.

Figure 6.1: *Visualization of a panoramic image from the Essen dataset taken by five cameras oriented sidewards and one oriented upwards. The regions of the individual images are approximately bordered in red. The measuring car is partly visible.*

On the one hand, the created panoramic images can be displayed in a combined visualization with 3D point clouds to provide a more comprehensive impression of the captured environment. On the other hand, they form the basis for image-based object recognition, for example, for the recognition of traffic signs (Zhu et al., 2016), and they can be used to colorize 3D point clouds (Swart et al., 2011).

For each image, the location where the image was taken and the camera's orientation

(a) *Colored by intensity values.* (b) *Colored by RGB values from panoramic images.*

Figure 6.2: *Visualization of a 3D point cloud from the Potsdam dataset.*

(a) *Sky shining through at tree branches.* (b) *Sign projected on the floor.* (c) *Car moving during the scan.*

Figure 6.3: *Color inaccuracies in 3D point clouds from the Potsdam dataset colored by panoramic images.*

are known, so the colors of the pixels can be projected from this position to the points of the 3D point cloud. Figure 6.2 shows the result of coloring a 3D point cloud in this way. However, this colorization is only an approximation. Images are taken every few meters, and the 3D point cloud is created continuously as the vehicle moves. Thus, the fundamental problem is that LiDAR takes a discrete sample of the environment that does not necessarily match the occlusions in the imagery in terms of depth. Figure 6.3 shows such incorrect occlusions. For example, the colors on narrow pillars, small signs, and tree branches deviate from reality.

This chapter presents an approach for a combined visualization of 3D point clouds and corresponding panoramic images in Section 6.2 and an approach for an automated analysis in Section 6.3.

6.1 Related Work

Panoramic images have been created manually since the 18th century (Comment, 1999). In digital camera technology, the decisive factor for widespread entry into the market was that individual consecutively captured images could be merged fully automatically into a single panoramic image using appropriate stitching software (Mann and Picard, 1994; S. E. Chen, 1995).

Szeliski and Shum (1997) describe how images can be transformed to merge them into a panoramic image. In older approaches, the images' positions relative to each other usually had to be configured manually. Brown and Lowe (2006) have developed an approach to automatically detect contiguous images in unsorted data and join them by several transformations. This is implemented by first using Scale Invariant Feature Transform (SIFT) to determine key points in the images (Lowe, 2004). These points can be found again in other images to determine an interconnection and the positioning. Subsequently, the images are straightened, and brightness values are adjusted to obtain a visually appealing overall image.

Blaser et al. (2017) describe a system with a stereo panoramic camera setup used to generate images, which are the basis for generating 3D point clouds via image matching. The correct alignment of images for precise coverage of a 3D point cloud is called *registration*. Coloring previously acquired 3D point clouds with image data based on precise registration is the subject of numerous research works: Pintus et al. (2011) present "a simple, fast and robust technique for semi-automatic 2D-3D registration capable to align a large set of unordered images to a massive point cloud with minimal human effort". Swart et al. (2011) show that "the automated registration of separately acquired panorama and LiDAR trajectories is feasible". Jianping Li et al. (2018) base the registration on semantic features by using vehicles "as registration primitives, which are extracted from both panoramic images and point clouds".

Free navigation within panoramic data is another use case for combining 3D point clouds and panoramic images. Current applications like Google's Street View (Anguelov et al., 2010) allow navigating in panoramic images only in discrete steps based on the locations where the images have been taken. Adam and Steinbach (2021) present an approach to how virtual panoramic images can be computed for arbitrary locations within an environment for which sufficient panoramic images have been captured alongside a 3D point cloud. Within the 3D point cloud, free navigation is easily possible. A modified version of U-Net is used for inpainting those pixels not covered in a rendering of the 3D point cloud at an arbitrary position. Because the network is trained on the given environment's panoramic images, this approach can create realistic renderings.

6.2 Visualization Concepts

Visualization offers powerful methods to combine and explore spatial data from different sources and captured by different technologies. In particular, visualization allows us to seamlessly integrate that data at the visualization stage, i. e., there is no need to integrate

the data in a preprocessing step at the data stage, e. g., creating a unified data model.

Chapter 5 already outlined how GPR data can be combined with 3D point clouds. The same can be done for panoramic images. With their help, details can be better recognized than in a 3D point cloud alone.

In the following, three prototypically implemented visualizations are presented.

Panoramic images are generally available as a series of consecutive images, and besides, a metadata file can be used to associate the capture position with each image. A possible overview for panoramic images of a dataset is created by displaying their capture positions on a map.

The first presented visualization thus consists of a map and the currently selected panoramic image next to it. Users can click on the location of a panoramic image within a map to select it. This image is displayed next to the map, and users can adjust its orientation. Because a panoramic image represents the environment as seen from one point, it is rendered on the inside of a sphere with the camera placed in the sphere's center. Users can click and drag to rotate the sphere, creating a visualization that resembles turning the head in the real world and looking in different directions.

As shown in Figure 6.4, the current viewing angle is additionally displayed on the map. This visualization is used for finding the appropriate panoramic image for a particular position and aligning it correctly.

The concept of showing available panoramic images in a spatial context is the basis for a combination of panoramic image data and 3D point clouds. Instead of displaying the capture positions on a map, they can also be placed directly in the 3D point cloud scene, as shown in the second visualization presented here. Figure 6.5 shows the spheres that are placed at the capture positions of the available panoramic images. A click on a sphere again opens the corresponding image.

The third presented visualization is another combination with 3D point cloud data. A panoramic image is projected onto the inside of a sphere as described above. The camera of the 3D scene is located in the center of this sphere. All points from the

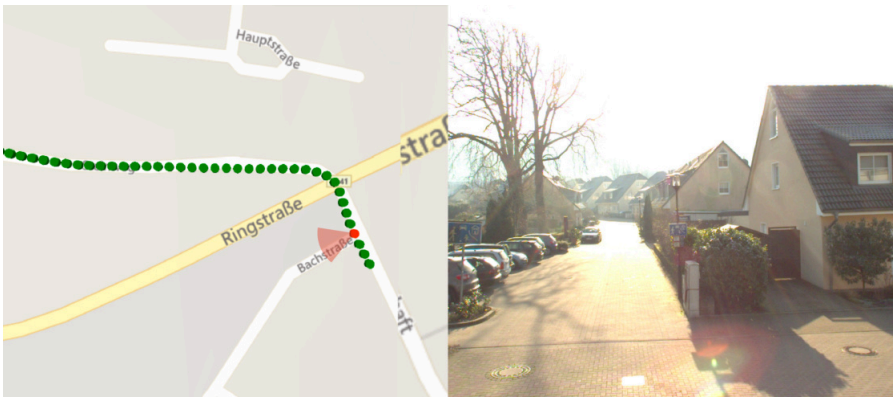


Figure 6.4: *Visualization of a panoramic image from the Essen dataset next to a map showing the point and direction of capture as well as other available panoramic image locations.*

corresponding 3D point cloud in the immediate vicinity are additionally rendered into the georeferenced scene. If the registration is correct, the effect is that the points cover the structures visible on the panoramic image, as shown in Figure 6.6.

However, problems arise in several situations during blending. For example, ledges in buildings' facades have been recorded in the 3D point cloud from all sides. At buildings' corners, the 3D point cloud contains points from both sides, but not all areas are necessarily visible in each of the images, which were taken from a fixed viewing angle. In this case, the 3D point cloud creates a translucent effect because structures are visible that cannot be seen in the image data from the current viewing position.

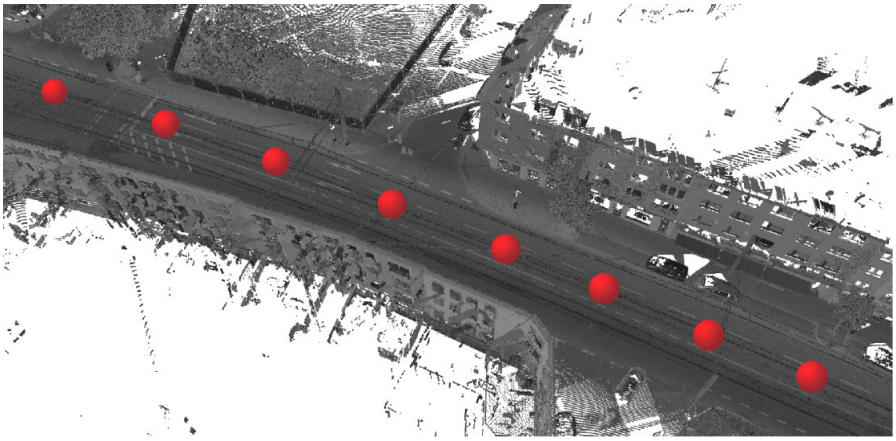


Figure 6.5: *Visualization of panoramic image locations within a 3D point cloud from the Essen dataset.*

Figure 6.6: *Visualization of a panoramic image with the points of a 3D point cloud from the Essen dataset projected on top of it.*

Figure 6.7: *Visualization of a panoramic image from the Essen dataset. Image stitching artifacts are visible at the catenary cables (red circles).*

The stitching of the individual images into a panoramic image causes another problem. Figure 6.7, for example, shows catenaries of a streetcar that do not merge seamlessly at the image boundaries. These problems imply that 3D point clouds and panoramic images can only be approximately blended.

6.3 Analysis Techniques

Combining 3D point cloud data and panoramic images is not only useful for enhanced visualization approaches, but it can also be used to achieve powerful analysis tools. Panoramic images, for example, are ideal for the recognition of traffic signs in the scanned environment (Zhu et al., 2016); the recognition is based on image analysis with artificial neural networks. For example, the implementation described in this thesis uses a combination of YOLO and Capsule Net for data analysis.

In Chapter 3, YOLO is applied for image-based classification; it detects those positions in unknown images where objects are located (Redmon and Farhadi, 2018) for which the network has been previously trained. A rectangular bounding box is returned for each detected object, outlining the image content in which the object was detected and a label indicating the respective semantic class.

When captured with panoramic images from a mobile mapping vehicle, the traffic signs have different orientations to the camera. Also, the signs are located at various heights, resulting in many different viewing angles. To prevent errors in the classification based on the orientation, a neural network designed to be independent of viewing angles, rotations, and distortions is used in the classification process.

Kumar (2018) published Capsule Net, a method based on Capsular Networks

Figure 6.8: *Panoramic images from the Essen dataset with annotated traffic signs.*

presented by Hinton et al. (2018). Capsule Net is used exclusively for classification, i. e., it cannot recognize object positions in images but only determines the semantic class for a mapped object. Thus, a so-called Region Proposal Network (RPN) must first be used to select sections from the panoramic images in which a traffic sign is depicted with a high probability. For this purpose, YOLO is used, which has been trained on traffic signs of all classes without any particular distinction. The bounding boxes determined by YOLO outline image regions that are subsequently extracted and input to Capsule Net. This network determines the semantic class of the respective sign.

The results are visualized in the panoramic images. Figure 6.8 shows the results of automatic annotation of traffic signs in panoramic images.

Based on the described approach in this chapter, panoramic images as a general source of semantic-rich information can be evaluated such that the derived semantic information eventually is assigned back as per-point information in 3D point clouds. The

corresponding process is briefly outlined in the following.

Using the position within the panoramic image and the information about the location where the image was taken, the approximate position of, for example, a sign in the 3D point cloud is determined. At this position, the 3D point cloud is searched for an object that corresponds to the expected structure, or previous semantic classification results are used to detect the points that correspond to the sign. That is, the approach approximately determines a subset of points in the 3D point cloud that is attributed by semantic information, extending the semantic classification of the 3D point cloud.

The described object detection in panoramic images can be applied to other object types besides traffic signs. For example, signals, balises, or ties on railroad tracks can also be recognized with this method if the networks are trained for these object types.

Another possible application is the anonymization of panoramic images to prepare them for publishing. For this purpose, faces and license plates can be detected and blurred in the image or eliminated in the 3D point cloud.

Chapter 7

Software Architecture and Implementation

The prototypical implementation based on the concepts presented in this thesis uses a pipeline structure for large 3D point cloud data, which can be outlined as follows: Input data can be specified as a list of files or directories, and the desired processing steps can be configured. Execution takes place locally or on a server set up for this purpose, parallelizing depending on the available resources. The results are made available at a defined location. The data analysis is performed using specially developed geometric and machine learning-based approaches or hybrid solutions.

The developed prototypical implementation has been used as the basis for the described analysis and visualization approaches.

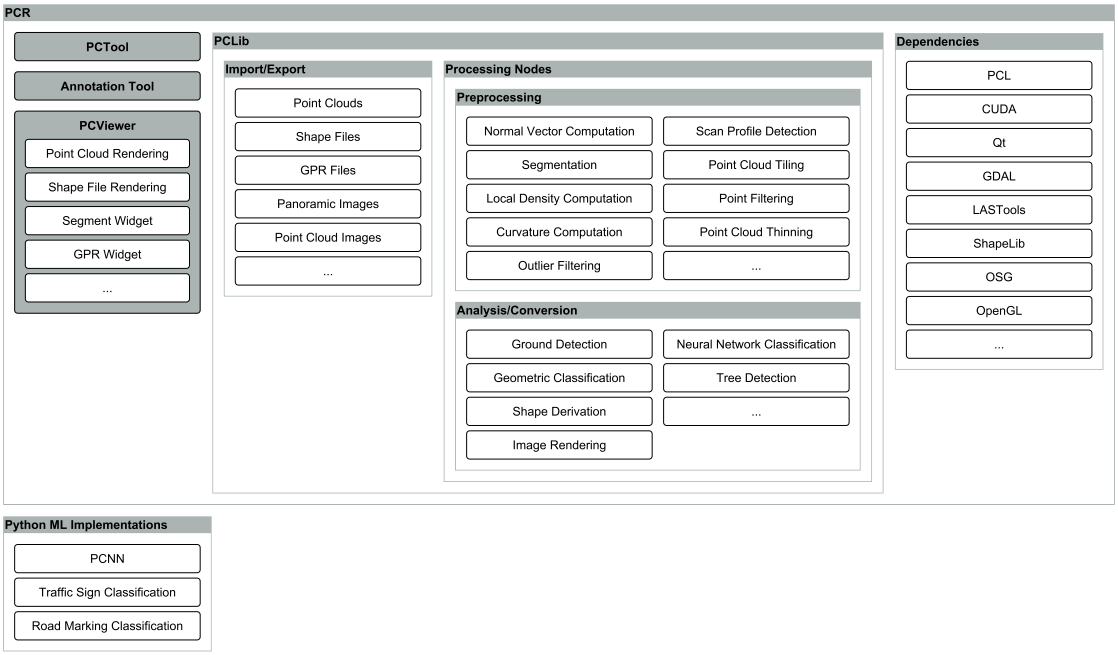


Figure 7.1: *Visualization of the software architecture used in the context of this work.*

7.1 Overview

Figure 7.1 provides a schematic overview of the software architecture of the prototypical implementation. Its core is the research framework PCR ("Point Cloud Research"). It provides a tool for processing 3D point clouds (*PCTool*), a visualization tool (*PCViewer*), and a tool called *Annotation Tool* for manual classification. The processing is based on *PCLib*, a library that provides the essential components for data processing. Additional machine learning tools, which are also presented below, have been developed for classification tasks and integrated into the described ecosystem.

7.2 Basic System Components

PCR is a general framework written in C++ for processing and visualizing 3D point clouds and related data. This framework has been used as an implementation basis for this thesis and partially extended for the specific features described in the thesis.

Three tools serve as the user interface for operating the framework. They rely on the following main dependencies:

Qt is a cross-platform framework used to create the user interfaces of the tools described in this thesis. Qt "contains a comprehensive set of [...] modularized C++ library classes". Detailed technical specifications can be found at the framework's website[1].

OSG and OpenGL/GLSL are used for 3D scene management and rendering. "The OpenSceneGraph [OSG] is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modelling"[2]. OSG is written in C++ and OpenGL and can be used cross-platform. OpenGL (Open Graphics Library) is a cross-platform API for hardware-accelerated rendering. GLSL (OpenGL Shading Language) is the C-style language used to write code for detailed control of the rendering pipeline within OpenGL. Technical specifications can be found on the OpenGL website[3].

PCLib as a complex "middleware for 3D point clouds" processes the data and provides functions for I/O operations. It is described in more detail in Section 7.2.2.

PCL (Point Cloud Library) was presented by Rusu and Cousins (2011) and provides basic functionality for processing 3D point clouds. For example, it is used by PCLib for the normal vector computation.

CUDA is a "parallel computing platform and programming model developed by NVIDIA for general computing on GPUs"[4]. First released in 2007, CUDA enables highly

[1] https://www.qt.io/product/qt6/technical-specifications
[2] http://www.openscenegraph.org
[3] https://www.opengl.org
[4] https://developer.nvidia.com/cuda-zone

parallel general-purpose processing on CUDA-enabled hardware and is used to accelerate computations of suitable parallelizable steps.

LAStools and ShapeLib are used for processing the standard data formats LAS and ESRI shape files, respectively. More detailed information about the libraries can be found on their websites[5,6].

GDAL (Geospatial Data Abstraction Library) is an open-source translator library for raster and vector geospatial data formats, which is used to process various data formats within the tools presented in this thesis. The library's documentation can be found on its website[7].

7.2.1 Core Tools

In this thesis, three core tools provide the basis for application scenarios and use cases related to processing, visualization, and manual classification. These tools have been designed, extended, implemented, and tested based on the described middleware.

The **PCTool** is used to configure and execute any combination of processing steps for supported geospatial data. Many specialized *nodes* are provided for this purpose, which can be connected in pipelines, as shown in Figure 7.2. The nodes are grouped into three types: Reader nodes, processing nodes, and writing nodes. They are presented in more detail in Section 7.2.2.

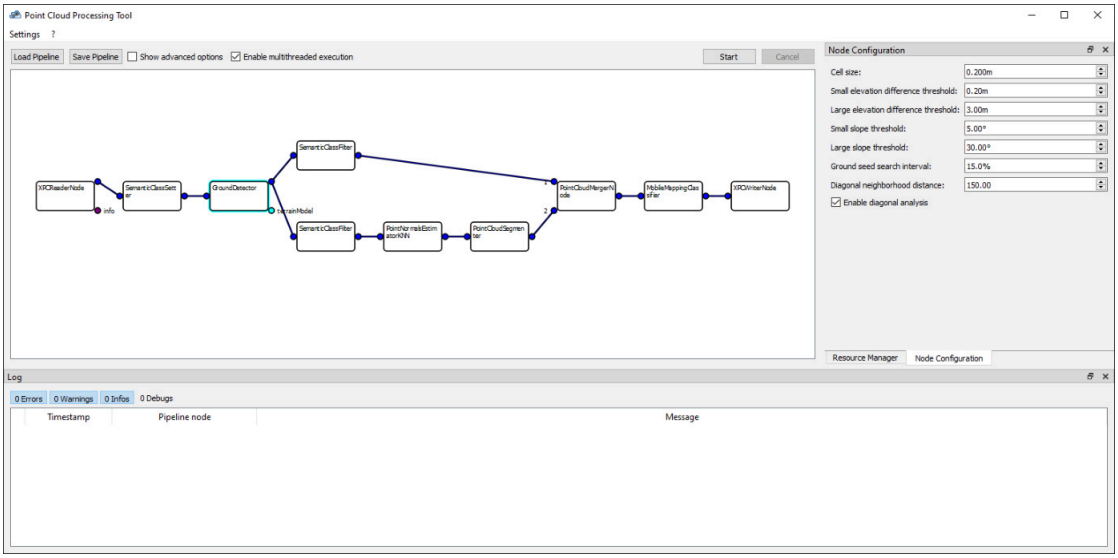


Figure 7.2: *Screenshot of an exemplary pipeline for mobile mapping data classification in the PCTool. On the right edge is the interface for configuring the currently selected node; here: Ground Detector.*

[5] https://rapidlasso.com/lastools
[6] https://github.com/OSGeo/shapelib
[7] https://gdal.org

The pipelines configured in the PCTool can be adapted to the specific execution environment. Resources such as CPU and GPU are adaptively managed and allocated to process multiple data packets in parallel. They can run on individual PCs as well as taking advantage of GPU clusters if available. Not only 3D point clouds are processed with the tool. Nodes can also support shape files, digital terrain models, and other data formats as input or output, depending on their task.

The second tool is called **PCViewer**. It is a visualization tool capable of displaying 3D point clouds and shape files.

Rendering 3D point clouds requires out-of-core data management due to the substantial number of individual measurement points that need to be represented. Datasets from mobile mapping surveys often contain several billion points. The visualization tool based on the works of Richter, Discher, et al. (2015) and Discher et al. (2017) has been extended and adapted to the specific requirements of the described approaches of this thesis. PCViewer uses a layered, multi-resolution kd-tree for massive 3D point clouds to manage the data. Extensive datasets, e. g., the 3D point cloud of an entire city with several billion points, can be handled in real-time using out-of-core algorithms. For rendering, suitable representative points are selected depending on the area being viewed and the zoom level so that the rendering time allows for interactive frame rates. Richter, Discher, et al. (2015) show that "Interactive frame rates can be achieved for each rendering technique as long as the overall number of rendered points does not exceed a certain threshold. Since the proposed out-of-core rendering approach limits the number of rendered points by dynamically selecting them, arbitrarily large datasets with varying point densities can be rendered in real-time as well."

Movement and rotation within the 3D scene are performed by moving the mouse while holding the left or right mouse button. The mouse wheel is used to zoom in and out. Alternatively, a so-called first-person mode can be activated, enabling movement with the keyboard, in which case only the viewing direction is controlled with the mouse.

Different rendering techniques can be selected to render the points. Thus, the rendered primitives and their size can be configured, and the color can be determined from a selected attribute. For example, 3D point clouds can be colored using their RGB values, height gradients can be used for coloring, or the color can be determined from the semantic class. If displayed 3D point clouds have been classified, each semantic class's visualization can be configured individually, or displaying this class can be disabled completely. Figure 7.3 shows multiple rendering options for a dataset with a 3D point cloud and shape files.

In addition to 3D point clouds, other data such as shape files or GPR data can also be displayed. GPR information is displayed in a separate widget next to the 3D scene. The widget can be seen in Figure 5.6.

Finally, the **Annotation Tool** enables manual semantic classifications in 3D point clouds or post-correction of automatically semantically classified data. The manually refined data can be used as training data for machine learning. This tool has been developed as part of several research projects with external partners.

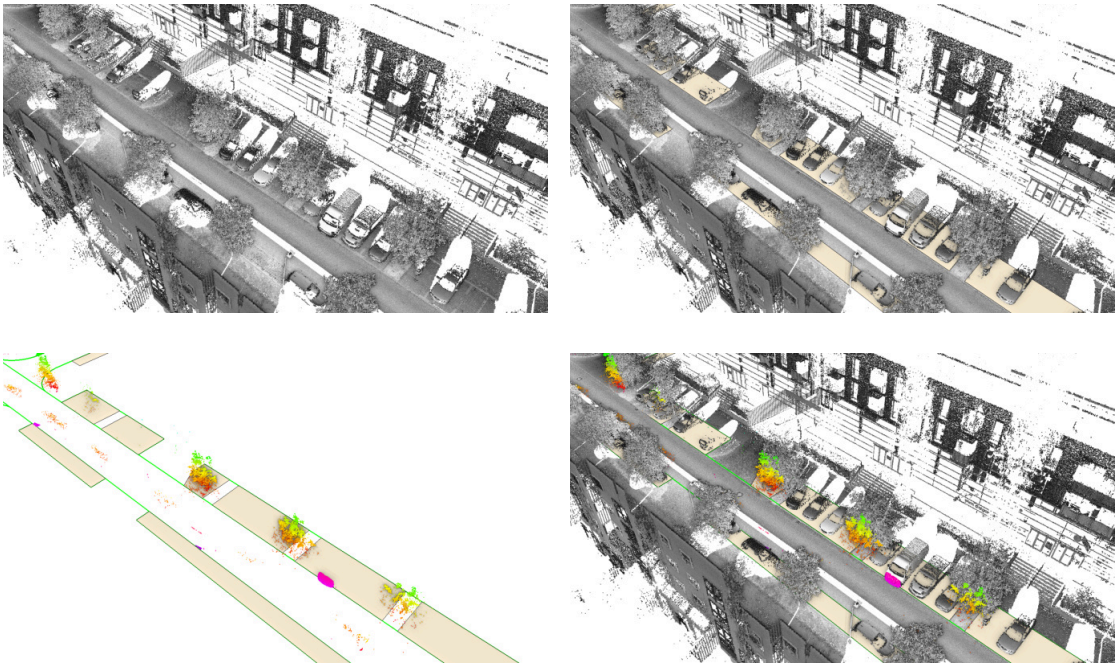


Figure 7.3: *Screenshots of PCViewer with different configurations for data visualization. Shown are parking areas (light brown areas), curbs (green lines), points above the road (pink to green height gradient) and the surrounding points (grayscale intensity values).*

Various selection modes are available for selecting points to accomplish this. For example, a sphere or a cuboid can be displayed as an auxiliary body and, when clicked, all points within the body are selected. A lasso tool and semi-automatic selections of neighboring points with similar height, color, or intensity are also implemented. The 3D point cloud can be displayed in parallel in a 3D scene and an orthographic 2D view, especially for a more straightforward classification of rails and their surroundings. Selected points can each be assigned the desired semantic class. Editing is simplified by options for locking already classified points and hiding individual semantic classes. Figure 7.4 shows a screenshot of the annotation tool.

7.2.2 PCLib

PCLib is a library that provides functions for processing 3D point clouds and other geospatial data; it forms part of the PCR framework. Of particular importance are the nodes, which are available to create pipelines in the PCTool. These are, on the one hand, reader and writer classes for I/O operations and, on the other hand, processing nodes in which the data is processed step by step.

Importers (readers) and **exporters** (writers) are available for 3D point cloud, vector, and image formats. Readers load a previously configured list of files from the file system into memory and make them available to subsequent nodes in the form of

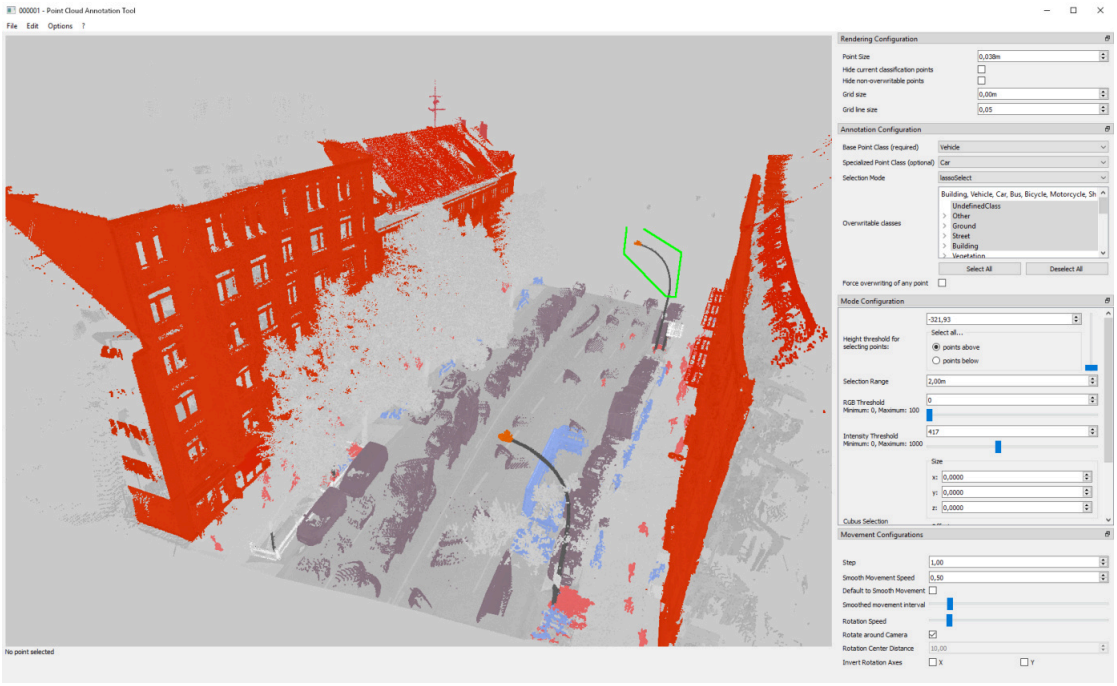


Figure 7.4: *Screenshot of a 3D point cloud in the Annotation Tool. Selecting points with the lasso tool is currently in progress. The area to be selected is visualized by the not yet closed green line.*

data packets. Writers write all received data packets in the respective file format into a previously defined directory.

For 3D point clouds, on the one hand, PCR's format `PCX` is supported, which enables straightforward access to individual attributes. Standard formats such as `LAS`, `LAZ` and `XYZ` are also supported.

Furthermore, shape files, GPR datasets, digital terrain models, and panoramic images can be read, and rendered images can be saved, e. g., for image-based classification.

The **processing nodes** provide a variety of functions for processing the data. They receive one or more data packets as input, process them according to their specific implementation, and pass them on to the following node. They can be further divided into preprocessing nodes and analysis nodes.

Preprocessing nodes are used for various tasks when preparing 3D point clouds for visualization or analysis. The tasks described in Chapter 2 can be accomplished with these nodes. Among them are nodes for computing normal vectors, curvature values, and local density. A segmentation of the input 3D point cloud can be performed, a scan profile detection, and thinning of dense 3D point clouds.

Most nodes have configuration options such as settings for the percentage of points remaining after thinning or for the number of nearest neighbors to be considered during a normal vector computation.

Finally, the **analysis nodes** are the main components for 3D point cloud classifica-

tion and other processing tasks. Nodes exist for, e. g., a ground detection and a multi-step geometric classification as described in Section 3.2, as well as the derivation of shapes for, e. g., detected road markings and nodes for scene rendering to generate images used for image-based classification as described in Section 3.3. For road marking classification, a node for neural network classification exists, which executes the external Python script for U-Net classification described in Section 7.3.3.

Several nodes have been implemented, extended, or optimized for the approaches described in this thesis. For example, the calculation of metrics and the extensive geometry-based semantic classification have been implemented, machine learning approaches for classifying 3D point clouds and image data have been integrated, and functionality for post-processing and exporting vector data has been added.

As an example, the node used to calculate the local point density explained in Section 2.4.1 will be presented here. First, the current 3D point cloud input data packet points are placed into a spatial data structure. Based on the node's configuration, this can either be a `pcl::gpu::Octree` or a `pclib::GeoRaster`. The octree implementation enables a precise analysis but is more time-expensive than the georaster, which implements a voxel grid as described before. When using the octree, a radius search is performed to count the number of neighbors for each point within the radius defined in the node's configuration. The local density of each point is calculated by its neighbor count divided by the volume of the sphere with the configured radius. When using the voxel grid, the neighborhood for each voxel cell is defined as all adjacent voxel cells whose center is within the configured radius from the current voxel cell's center. For each voxel cell, the summed number of points in its neighborhood is computed. This number divided by the combined volume of the neighborhood voxel cells approximately represents the local density of all points within this voxel cell. All points from the input data packet get assigned the additional attribute of their local point density, and the packet is output to the following pipeline node.

7.3 Python ML Implementations

In addition to PCR, several Python scripts are used for machine learning on 3D point clouds and rendered images. In the following, these scripts are presented in more detail.

7.3.1 PCNN

PCNN is the artificial neural network already presented in Section 3.4 for semantic classification of 3D point clouds. It is a combination of PointNet++ (Qi, Yi, et al., 2017) and DGCNN (Y. Wang et al., 2019) to form a network that has shown to be suitable for processing mobile mapping 3D point clouds. Figure 3.18 in the classification chapter depicts the structure of the neural network.

The project includes a GUI to control processing, but it can equally be controlled from the command line. PCNN supports processing 3D point clouds in `CSV`, `LAS`, and

`XPC` format. It is written in Python using PyTorch as the main dependency. In a first step, all input 3D point clouds must be preprocessed (`preprocess.py`) before they can be used further. Preprocessing involves collecting metadata such as min and max values and remapping the semantic classes' IDs to use consecutive natural numbers without gaps. Density reduction takes place to save time in subsequent steps. Finally, the data is stored in the standardized `h5` format, with all values normalized using the previously collected metadata. Preprocessed 3D point clouds can then be used for training and prediction.

The training uses already classified 3D point clouds as train and test data to train the artificial neural network (`train.py`). The training can be interrupted after each epoch and continued later. Also, the best model so far is stored after each epoch. As measurement value for this accuracy, mIoU (mean Intersection over Union), mAP (mean Average Precision), or loss can be used. The 3D point clouds used for train and test must have been preprocessed together to have undergone the same normalization.

The prediction script classifies 3D point clouds using a previously trained model (`predict.py`). In addition to the preprocessed 3D point clouds, the training's metadata file must also be input to obtain correct information for mapping the semantic classes. The output of the classified data is a `CSV` file. It contains only the points sampled during classification with their semantic class. Points in the immediate neighborhood can subsequently be assigned the same semantic class to classify all points of the original 3D point cloud.

The project can also execute previously configured classification pipelines, including the preprocessing step (`execute_pipeline.py`) to automate more complex workflows.

Finally, `benchmark.py` provides a tool that calculates metrics such as precision, recall, F_1-score, IoU, and a confusion matrix based on ground truth data.

7.3.2 Traffic Sign Classification

The traffic sign classification in images described in Section 6.3 is also implemented as a prototypical proof-of-concept implementation in Python, using TensorFlow and Keras.

The project contains a YOLOv3 model trained on the German Traffic Sign Detection Benchmark (GTSDB) from Houben et al. (2013). For the training, all traffic signs are considered the same class because the network is only used to detect the position of traffic signs in panoramic images, not to classify them.

An implementation of Capsule Net performs the classification. This network, in turn, is trained on the German Traffic Sign Recognition Benchmark (GTSRB) by Stallkamp et al. (2012), taking into account the different classes of traffic signs.

Both networks used are adapted variants of publicly available implementations.

A script for traffic sign classification receives a directory of panoramic images as input. YOLO is first applied to the images. The bounding boxes of all areas recognized as traffic signs with a probability of at least 0.9 are collected. Each of these areas is extracted as an independent image and classified using Capsule Net. The results are output as a `CSV` file in which the semantic class is assigned to each image. A projection of the results into 3D point clouds acquired in parallel has not been implemented yet.

7.3.3 Road Marking Classification

The road marking classification project essentially consists of a PyTorch implementation of U-Net, which can be trained and used for predictions using two scripts.

The network is trained using orthoimages rendered from 3D point clouds in which all road markings have already been classified. The classification can be realized manually using the annotation tool, or the information is derived from existing shape files. In the latter case, it must be ensured that not only the mere content of the specified shape areas is classified. The actual situation with intensity values must also be considered to avoid mistakenly classifying places as markings where there are no markings in the 3D point cloud.

Subsequently, prediction can be used to determine the type of road marking for each pixel for previously unknown orthoimages like described in Section 3.3.1. Similarly, if a pixel does not represent a road marking, it is classified as “other”.

The results can be further processed within pipelines in the PCTool. A corresponding node exists for a direct connection. This node generates the required orthoimages, activates the external prediction script, and reads the results back in.

Chapter 8

Evaluation of Application Scenarios

This chapter presents two evaluations of application scenarios in detail. First, in Section 8.1, 3D point clouds of a mobile mapping dataset are semantically classified using geometric methods and machine learning. The results are compared in terms of quality and runtime. Section 8.2 focuses on a specific use case and provides information about runtime and accuracy in the detection of road markings in 3D point clouds and subsequent derivation of vector data.

All performance measurements have been made on hardware with an Intel i7-6700 8 core 3.4 GHz CPU, 32 GB RAM, and an NVIDIA GeForce GTX 1080 Ti GPU with 11 GB dedicated and 16 GB shared memory.

8.1 Results for Semantic Classification of 3D Point Clouds

As described in Chapter 3, different methods can be used for the automatic semantic classification of 3D point clouds. A geometric method and the machine learning method PCNN, based on DGCNN and PointNet, have been presented as methods that can generally be used for a semantic classification without focusing on a particular use case. They are used for the overall semantic classification of mobile mapping 3D point clouds to enrich the raw data with semantic information. Thus, they do not focus on individual use cases and must handle large datasets efficiently. The two methods are compared here to evaluate the quality of the results and the required computation times.

The data used for the evaluation is from the Hamburg dataset. This dataset is particularly suitable as a reference dataset within the evaluation, as it is representative for major European cities. The acquisition was performed with modern hardware, and the 3D point cloud is available in high density. The data was acquired in moving traffic, so typical artifacts and occlusions of vehicles and pedestrians occur. The acquired data is available in unprocessed original form; thus, realistic measurements can be made with this real-world data. The 185 MB file size test area consists of 9 569 752 points. The chosen test area represents a street with buildings on both sides, some trees, vehicles on and beside the roadway, and various post-like structures such as traffic signs and lanterns. It is thus exemplary for an average section of a modern city.

PCNN supports significantly more different semantic classes and can make detailed distinctions, e. g., between a post and signs mounted on it. It also recognizes bicycles and people, among other objects. The current implementation of the geometric methods

supports only the base classes “Building”, “Ground”, “Post-like structure”, “Vegetation”, and “Vehicle”, plus a preceding outlier analysis. To make the results of both methods comparable, PCNN is also limited to these classes within the evaluation. For this, the unsupported classes are mapped to their parent category. For example, road markings are considered as ground.

For the evaluation, a ground truth classification was manually created. In this ground truth, pedestrians, bicycles, bicycle holders, and other classes that cannot be mapped to an appropriate higher-level category and are not covered by the geometric method's current implementation were removed. This creates an equal basis for both methods and allows the computed accuracy values to be compared directly. Besides, boundary regions of the 3D point cloud were removed, in which the point density decreases strongly and which do not provide helpful information in any classification method. This cropping corresponds to a restriction to the vehicle's closer vicinity when the data were acquired.

Figure 8.1 first shows an overview visualization of the entire scan colored by intensity values, and in Figure 8.1(b) the ground truth information of the area relevant for the classification is shown. Subsequently, only those points of the automatic classification are shown for the geometric method and the machine learning method PCNN, which deviate from the ground truth. The color in these visualizations is based on the incorrectly assigned semantic class making it possible to see which points have been assigned the wrong class and which class was chosen. The correctly classified points are not shown.

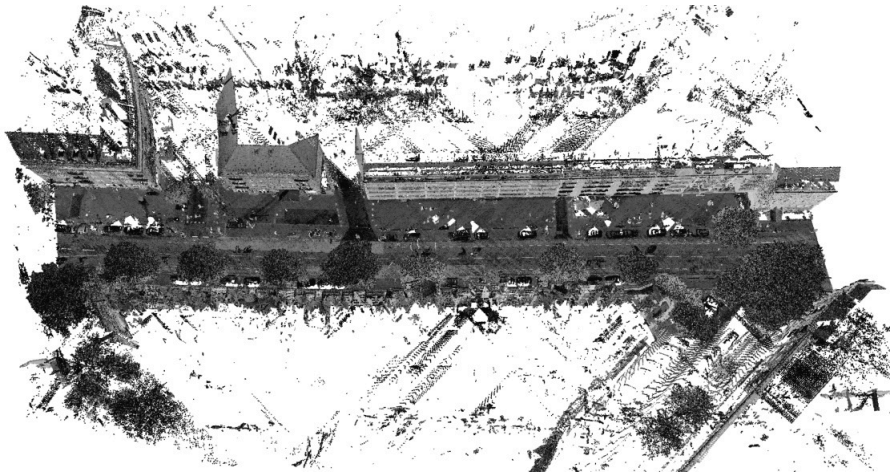
For a more detailed insight into the classification results, the 3D point cloud is shown again from a different perspective in Figure 8.2. Again, the entire 3D point cloud colored by intensity values, the ground truth colored by semantic class, and the points incorrectly classified in both methods are shown.

Semantic Class	Number of Points	Precision	Recall	F_1-Score	IoU
Building	4 093 082	99.3 %	96.5 %	97.9 %	95.8 %
Ground	3 587 280	96.8 %	99.5 %	98.1 %	96.3 %
Outlier	43 966	28.3 %	8.7 %	13.3 %	7.1 %
Post-like	42 371	89.4 %	65.6 %	75.7 %	60.9 %
Vegetation	1 562 153	95.9 %	95.1 %	95.5 %	91.4 %
Vehicle	240 900	89.8 %	88.2 %	89.0 %	80.2 %
Average		*83.2 %*	*75.6 %*	*78.2 %*	*72.0 %*
Weighted mean		*97.2 %*	*96.7 %*	*96.9 %*	**94.3 %**

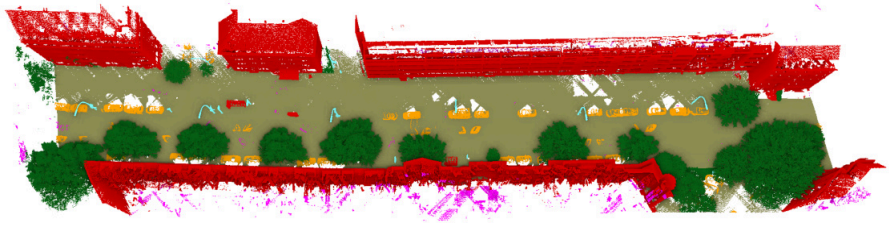
Table 8.1: *Accuracy values of the geometry-based semantic classification.*

Tables 8.1 and 8.2 list the precision, recall, F_1-score, and IoU (Intersection over Union) values for the geometry-based and the machine learning-based method for each of the semantic classes and as overall aggregations. With weighted mean IoUs of 94.3 % (geometry-based) and 94.5 % (machine learning-based), both methods delivered similar results concerning the overall quality.

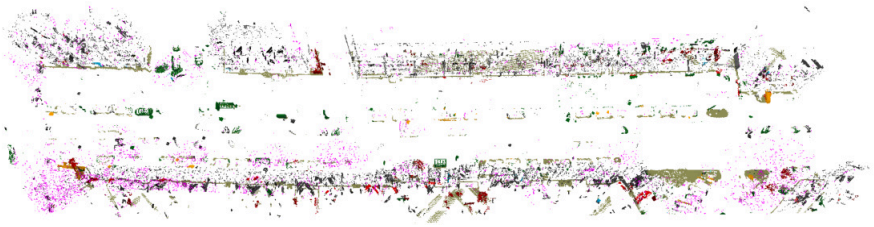
The most notable difference occurred with the outliers. The geometric method uses a purely spatially determined delimitation for this purpose. All points with less than the specified threshold value of neighboring points in their vicinity were marked as outliers.



(a) *Complete 3D point cloud colored by intensity.*



(b) *Ground truth information. Colored by semantic class: Building (red), Ground (brown), Outlier (magenta), Post-like (cyan), Vegetation (green), Vehicle (orange).*

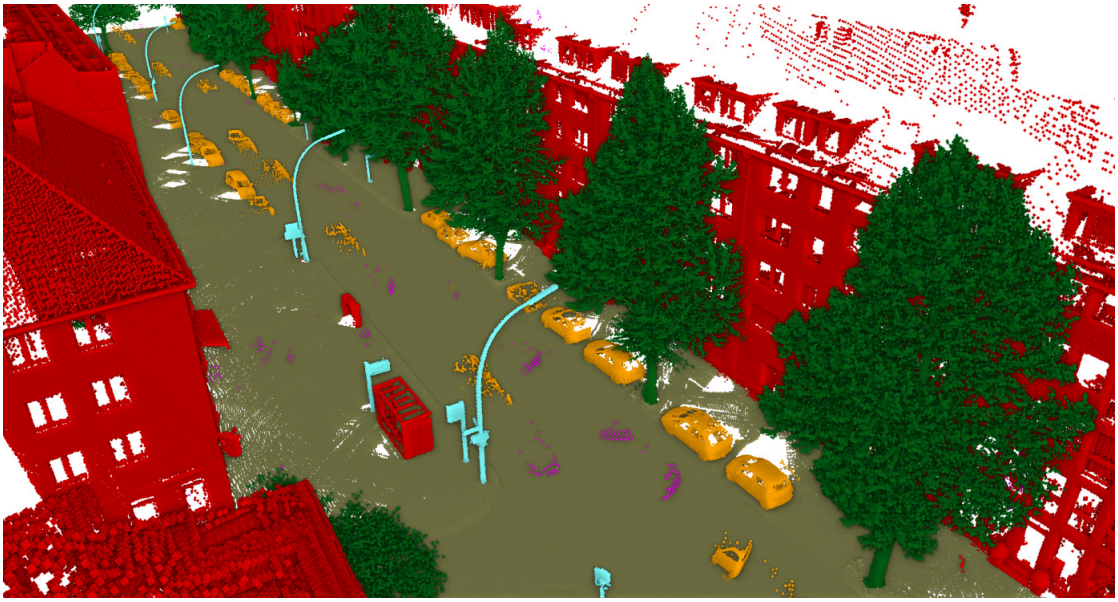


(c) *Points from geometry-based classification differing from ground truth. Colored by wrongly assigned semantic class.*

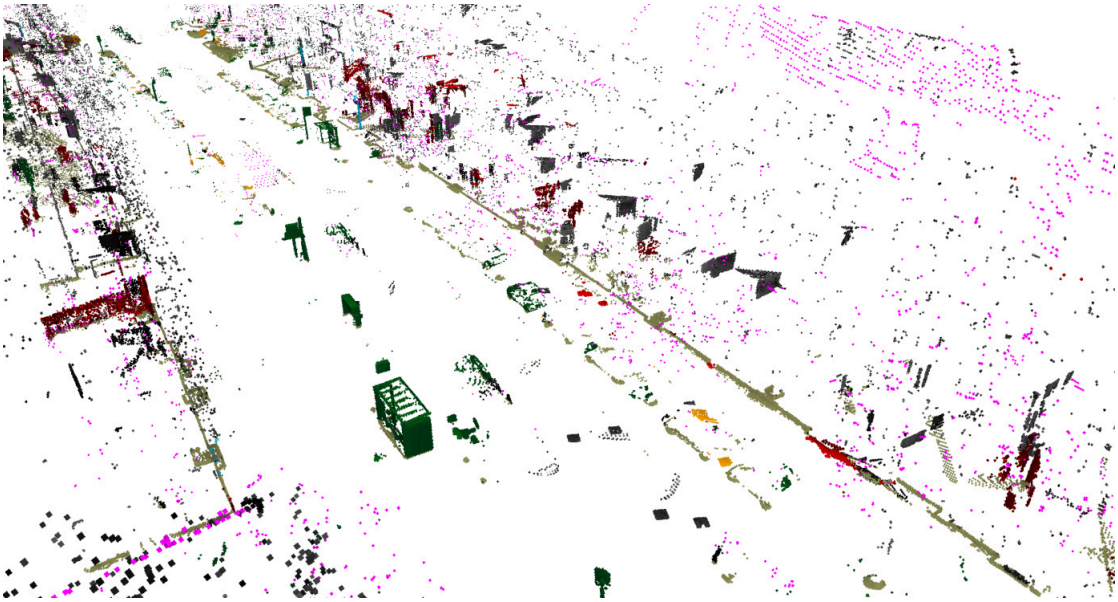
(d) *Points from machine learning-based classification differing from ground truth. Colored by wrongly assigned semantic class.*

Figure 8.1: *Visualization of ground truth semantic classes in a 3D point cloud and the points differing from ground truth in both classification methods. The original 3D point cloud from the Hamburg dataset (a) was cut to the relevant area for evaluation (b–d).*

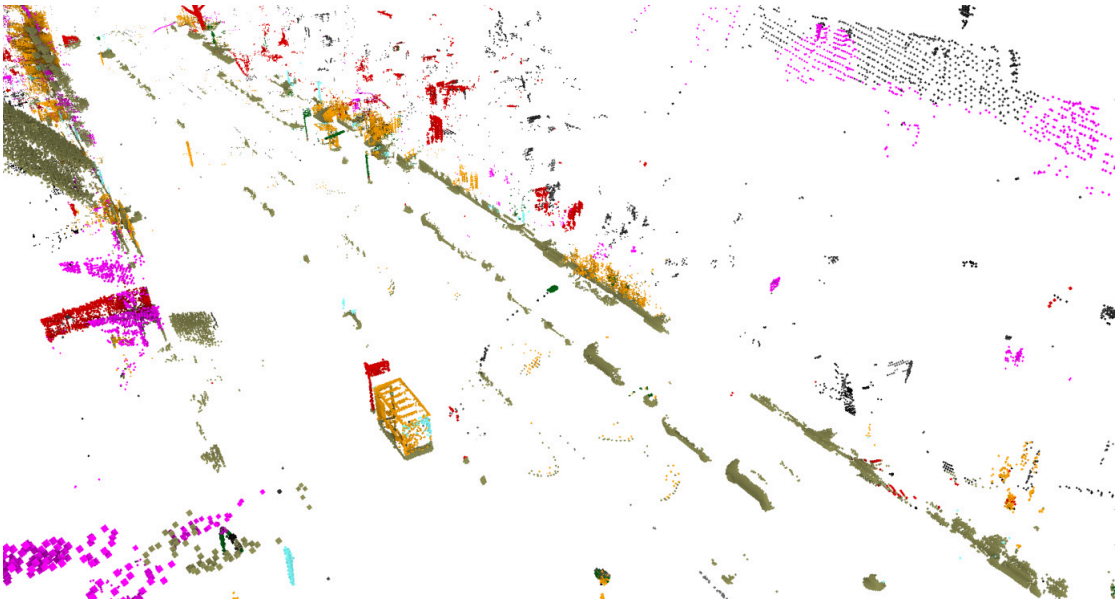
(a) *Complete 3D point cloud colored by intensity.*



(b) *Ground truth information. Colored by semantic class: Building (red), Ground (brown), Outlier (magenta), Post-like (cyan), Vegetation (green), Vehicle (orange).*



(c) *Points from geometry-based classification differing from ground truth. Colored by wrongly assigned semantic class.*



(d) *Points from machine learning-based classification differing from ground truth. Colored by wrongly assigned semantic class.*

Figure 8.2: *Detail view of the 3D point cloud shown in Figure 8.1.*

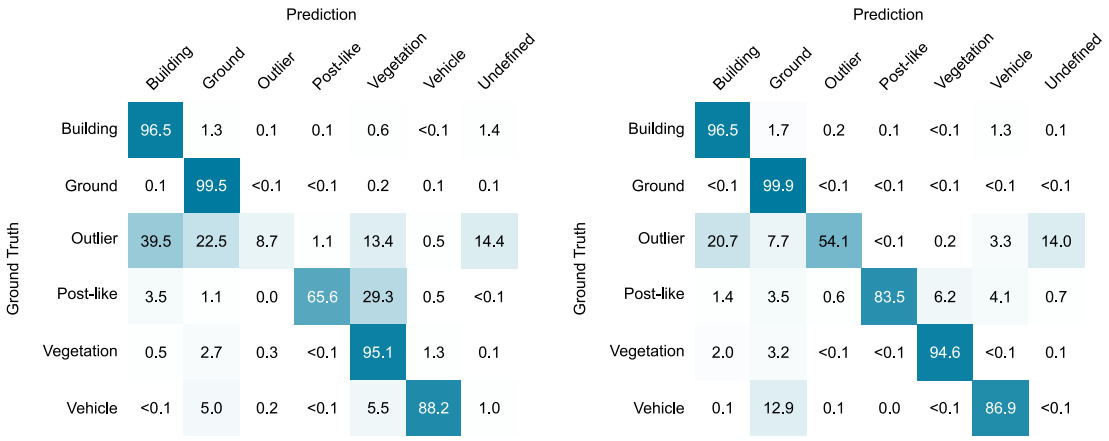
Semantic Class	Number of Points	Precision	Recall	F_1-Score	IoU
Building	4 093 082	99.0 %	96.5 %	97.7 %	95.6 %
Ground	3 587 280	95.9 %	99.9 %	97.9 %	95.8 %
Outlier	43 966	68.0 %	54.1 %	60.3 %	43.1 %
Post-like	42 371	89.9 %	83.5 %	86.6 %	76.4 %
Vegetation	1 562 153	99.7 %	94.6 %	97.1 %	94.4 %
Vehicle	240 900	78.8 %	86.9 %	82.6 %	70.4 %
Average		*88.6 %*	*85.9 %*	*87.0 %*	*79.3 %*
Weighted mean		*97.2 %*	*97.0 %*	*97.1 %*	**94.5 %**

Table 8.2: *Accuracy values of the machine learning-based semantic classification.*

On the other hand, in the ground truth, points were also marked as outliers that lie behind the building facades, as seen from the roadway. Because of the high number of points in these areas, only a few were marked as outliers by the geometric method. With PCNN, a larger number of outlier points was recorded. However, due to the small overall number of outlier points, this hardly affected the classification result's overall value.

For a more detailed insight, confusion matrices for the semantic classes are presented in Figure 8.3. In these, for each semantic class of the ground truth, it is shown to which proportion its points were assigned to which semantic class in the automatic classification. In both methods, points that could not be assigned to any semantic class were marked as "Undefined".

Significant deviations from ground truth also occur in PCNN classification during outlier detection. The points wrongly classified as "building" originated mainly from roof areas and partly from the area behind the facades.



(a) *Geometry-based classification.* (b) *Machine learning-based classification.*

Figure 8.3: *Confusion matrices showing which percentage of points has been assigned which semantic class compared to ground truth. "Undefined" denotes points that did not receive any semantic class during the automatic classification.*

In the PCNN classification results, it is visible that the distinction between vehicle

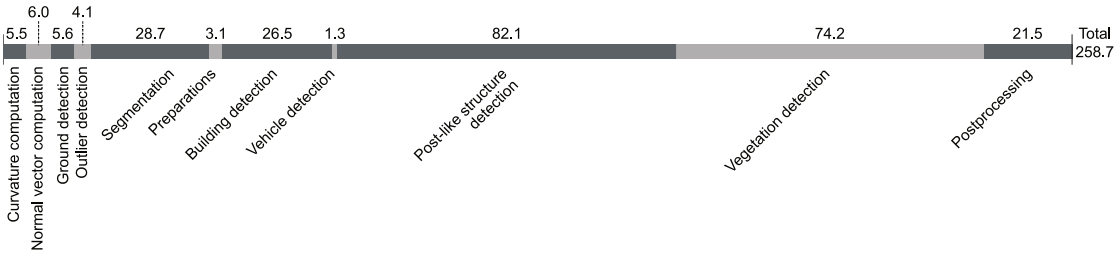


Figure 8.4: *Processing times for the geometry-based 3D point cloud classification. Values in seconds.*

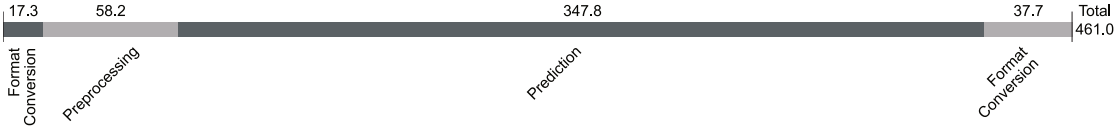


Figure 8.5: *Processing times for the machine learning-based 3D point cloud classification. Values in seconds.*

and ground causes difficulties in addition to the outlier detection. Here, as can also be seen in Figure 8.2(d), it was the wheels and sidewalls of the vehicles that were incorrectly assigned to the ground. The same also occurred in the geometric method but to a lesser extent. In return, some vehicles were falsely identified as vegetation in the geometric method. This was not a problem in PCNN.

The biggest weakness of the geometric method apart from the outliers is the wrong assignment of post-like structures to vegetation. Some posts of lanterns and traffic signs were not assigned correctly in whole or in part.

Despite the listed weaknesses, both methods are well suited for the semantic classification of mobile mapping datasets of this kind and provided good overall results for further processing.

Especially concerning the processing time, it became clear that automatic classification is inevitable. A dataset of this size requires several hours if classified manually at the level of detail shown. Classification using the geometric method took just over four minutes (259 seconds), and classification using PCNN took just under eight minutes (461 seconds). For the geometric method, this corresponds to about 37 000 points or 0.73 meters per second. For PCNN to about 21 000 points or 0.41 meters per second. Figures 8.4 and 8.5 show how much time was spent on each processing step in the automatic classification.

Extrapolated to an entire city's dataset, Potsdam's classification takes about nine days with the geometric method and about 16 days with the classification via PCNN.

Additionally, time was spent on configuration for the geometric method, which must be adjusted for each dataset, but can be used across most scans within a dataset.

For PCNN, additional time was spent to train the artificial neural network. Training requires an average of at least four hours, but the time varies by configuration and dataset

and is often significantly higher. This training also requires ground truth data, which must be generated in advance. For example, the geometric method can be used for an initial classification, which is corrected and refined by additional semantic classes before being used for the training.

In summary, both methods provided equivalent results on the test dataset. However, the processing time, including all necessary configurations and training, was significantly lower for the geometric method than for the machine learning-based method PCNN. Nevertheless, PCNN has the significant advantage of supporting arbitrarily detailed semantic classes, making it more suitable for some use cases. The high amount of time used to create ground truth information and train the network is amortized when large datasets such as whole urban areas are classified. Individual configurations as needed for the geometric method, which require specific process knowledge, are unnecessary. Thus, both methods can provide relevant results, and the use must be made dependent on the particular application.

8.2 Road Marking Detection and Vector Data Derivation

This section investigates the automatic detection of road markings and the derivation of vector data based on them.

3D point clouds and shape files of intersections from the Hamburg dataset are used as test datasets. The 3D point clouds acquired with LiDAR enable recognition of road markings via the contained intensity values. Ground truth information is available via manually created shape files in which the position and type of road markings are recorded. Line markings, however, are only available as lines instead of rectangles, so no exact evaluation of polygon areas can be performed.

An image-based classification is performed as described in Section 3.3.1. Subsequently, the vector data is derived as described in Section 4.1.1 and stored as shape files.

Figure 8.7 shows the resulting shapes in comparison to the ground truth information. The corresponding 3D point cloud is shown in Figure 4.2.

The artificial neural network U-Net, which was used for the classification, was trained for 5 hours using 3D point clouds previously classified with the ground truth information.

The shape file shown in Figure 8.7 was created in 129.3 seconds from a 3D point cloud with 25.8 million points. Thus, the throughput of the combined classification and vector data derivation was about 12 million points per minute, which corresponds to approximately 450 meters of captured road per minute. The bottleneck in the analysis is the file operations during rendering and classification because many images are written to and read from disk. This could be avoided with an in-memory implementation. Figure 8.6 shows the time distribution for individual processing steps as measured on average during five runs.

The resulting vector data all corresponded in shape to the recognized structures from the 3D point cloud data. Matching the shape with the ground truth information is not possible as described. Therefore, the specified accuracy values were only determined based on the assigned type of road marking.

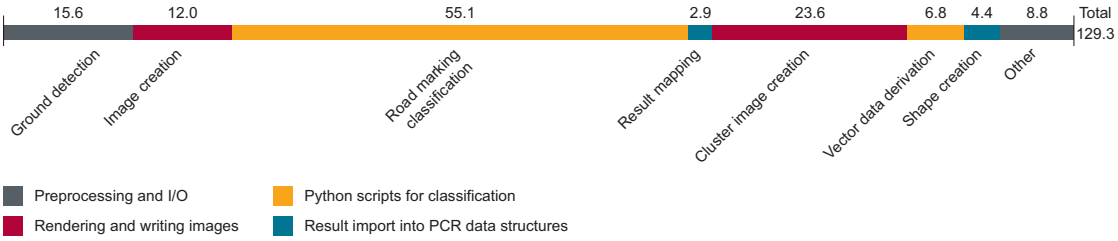


Figure 8.6: *Processing times for the automated shape derivation for road markings in 3D point clouds. Values in seconds. "Other" includes I/O for 3D point cloud and shape files.*

In an evaluation of a larger test dataset, an overall precision of 92.5 % and an overall recall of 93.9 % were achieved, as shown in Table 8.3.

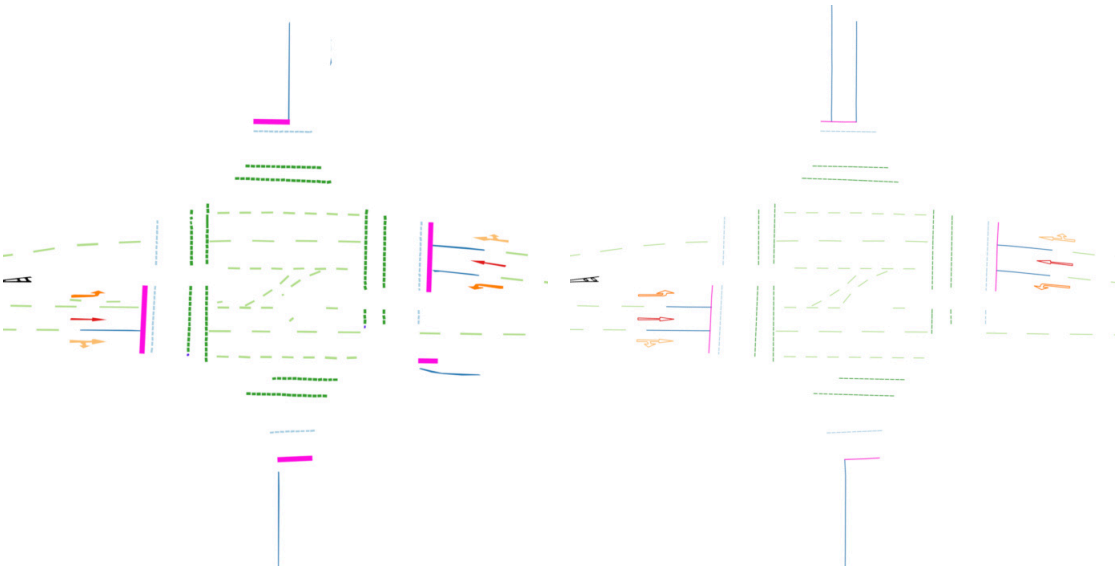
Errors in the classification mainly were undetected road markings in areas of overall high intensity in wet regions of the road, wrongly assigned semantic classes like an incorrect type of line, and high-intensity regions wrongly classified as road markings.

Marking	Precision	Recall	F_1-score
Arrows (48)	93.5 %	89.6 %	91.5 %
Lane dividers (230)	96.6 %	98.7 %	97.7 %
Stop lines (39)	84.8 %	100.0 %	91.8 %
Pedestrian crossing lines (228)	91.2 %	97.0 %	94.0 %
Cycle track lines (270)	91.2 %	87.7 %	89.4 %
Barred areas (3)	100.0 %	100.0 %	100.0 %
Weighted average	92.5 %	93.9 %	93.1 %

Table 8.3: *Road marking classification accuracy values.*

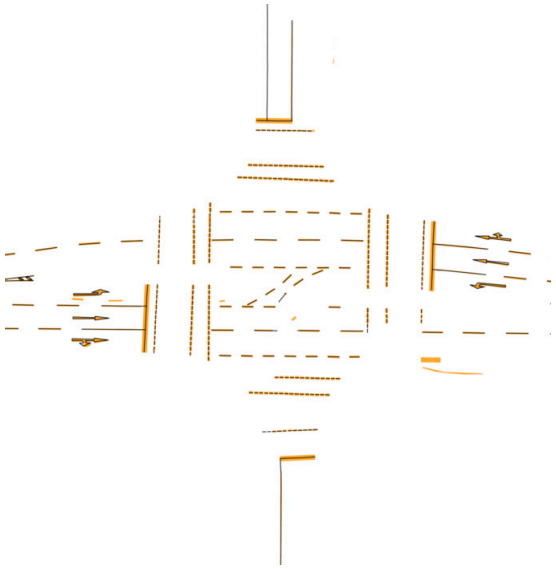
Both applications clearly show that large amounts of data can be processed and analyzed efficiently and automatically using suitable approaches and that high-quality results can be achieved in the process. Manual processing would be significantly more time-consuming and can instead be replaced by automated processing with less costly manual rework.

The lack of ground truth data continues to limit many comparable analyses in research publications. However, more datasets are gradually becoming available that can be used as ground truth in various use cases. With the strong development in machine learning, the availability of such data is expected to further improve in the coming years.



(a) *Classification result with shapes for individual road markings. Semantic classes are represented by distinct colors.*

(b) *Ground truth information. Note that no information about line widths is given. Colors represent the same semantic classes as in (a).*



(c) *Comparison of automated classification result (orange) with ground truth data (black).*

Figure 8.7: *Visualization of automatically generated road marking shapes and the corresponding ground truth data from the Hamburg dataset.*

Chapter 9

Conclusions and Future Work

This thesis shows how 3D point clouds from mobile mapping scans can be processed, analyzed, and visualized to detect objects and structures in road and railroad environments. The combination of 3D point cloud data acquisition hardware and software partly based on machine learning enables semantic-sensitive interpretation of the built environment. In this regard, mobile mapping scan data serves as the fundamental data source used to create spatial digital twins of the built environment.

Among the challenges for a seamless processing chain for current and future applications and services is a solid and robust but also extensible semantic classification based on the needs of each application domain. In this respect, 3D point clouds as unsorted, unstructured data represent ideal input data for machine learning approaches. Object categories can be filtered through semantic classification, and individual objects in the data can be derived and exported to external systems. Different methods for automatic semantic classification of 3D point clouds, the necessary preprocessing, and enrichment with other data sources have been presented in this work.

Preprocessing is a key step for creating spatial digital twins based on 3D point clouds. It is required for semantic classification approaches as well as for real-time visualization techniques. It manages data harmonization and fusion—a typical requirement when the built environment is captured frequently and redundantly. Outlier detection and point density reduction, especially for dense areas, remove unnecessary points from 3D point clouds and reduce the amount of data processed or visualized.

Several methods have been presented in this work that can be used for use-case independent semantic classification of mobile mapping 3D point clouds. The geometric method is based on metrics presented in Section 3.2; it requires a specific configuration for each dataset but provides convincing results and has a lower runtime than the machine learning-based analysis called PCNN, a PointNet and DGCNN hybrid. However, PCNN can perform a much more detailed classification with an arbitrarily large number of semantic classes, provided that suitable training data are available to train the artificial neural network. The classification results of the presented approaches have been compared in detail in Chapter 8. As far as the classification of semantic classes supported by both methods is concerned, the quality is almost identical with a weighted mean IoU of about 94 %.

The results of this work have shown that both approaches can be combined, e. g., by using the geometric method to generate the training data for the machine learning

method. The resulting datasets can be manually improved with a suitable tool, and the classification can be refined, generating extensive ground truth information. Since sufficiently large ground truth information is essential for machine learning, an efficiently usable tool for manual semantic information processing in 3D point clouds is important. Such a tool has been developed and extended for the framework used in this work.

Image-based classification methods have also been presented in this thesis. For example, an image recognition method that works on orthographically rendered images of ground points is particularly well suited for recognizing road markings. Road markings can be detected and classified in these images. The results can be used to derive vector data of, for example, intersections, representing a map of all existing road markings. These maps can be used for georeferenced digitization of road data and restoring the previous condition after construction works. Based on available ground truth information, the vector data generated by the presented method have been evaluated and achieved a F_1-score of 93.1 %.

A scan profile analysis specifically tailored to those 3D point clouds often generated during mobile mapping scans in railroad networks enables further analysis steps for the railway infrastructure. For this purpose, individual scan profiles can be extracted and rendered as described. Image classification techniques are applied to the rendered images, enabling the detection of tracks and specific objects in the track environment, such as signals and balises.

However, 3D point clouds are not the only source of spatial information. For this reason, this work has explored how 3D point clouds can be combined with other types of spatial data, such as through GPR and with panoramic images. A suitable visualization is critical for exploring such combined data. Therefore, appropriate ways to display the different data sets together have been presented and explored for their usability. The GPR data enables the analysis of the road substructure so that problematic areas of the roadway, such as developing potholes, can be identified. When combined with information from the 3D point cloud, utility hole covers, for example, can be identified and considered separately when evaluating pavement anomalies. Supplementing the 3D point clouds with information from panoramic images adds additional color information about the environment. In particular, this enables an unambiguous classification of traffic signs, which is only possible to a limited extent using intensity values alone. This classification can be performed directly on the images using artificial neural networks, for example, and then be transferred to 3D point clouds acquired in parallel using positional data about the capture locations.

To demonstrate the feasibility and practicality of the presented approaches, an extensive C++ 3D point cloud framework has been adapted and extended with tools for visualization, analysis, and manual processing of classification data. For example, Python machine learning scripts have been developed to provide a solid foundation for comprehensive visualization and processing of mobile mapping 3D point clouds and can be used to process large amounts of data.

Future Work

Extensions of the presented approaches that have not yet been implemented are reasonable at various points. For example, the geometry-based classification could benefit from the support of additional semantic classes. If the existing base classes are subdivided into subclasses, more detailed assumptions can be made. Subdividing ground points into the street, curb, sidewalk, and unsealed ground could help identify vehicles or lampposts better because additional assumptions about relative positions become possible. The identification of doors in building facades would enable the recognition of specific entrances. In combination with an analysis of the curb height, accessibility statements would become possible. As described in the evaluation, there are points behind building facades due to reflections or the inclusion of interior spaces behind windows. Better differentiation of building facades and these points would help to remove many points from the data sets that do not have significant meaning.

It would also be possible to perform classifications on a combination of multiple scans of the same area to fill in gaps created by shading or unfavorable viewing angles. Not only could multiple mobile mapping scans be combined, but the data could be supplemented with aerial scans when available. This would first require precise registration for the correct relative positioning of the data sets. If this is successful, the combination offers new potential for a comprehensive representation of the scanned environment.

A combination of the geometry-based method and the machine learning-based method could better assess classification results. Matching results are more likely to be correct, and areas with divergent semantic classes could be examined in more detail by specific methods in a detailed analysis. Machine learning-based methods could also be adapted or developed for both segmentation and outlier detection.

For semantic classification with PCNN, it is essential to use sufficiently large training data to train a network that is less dependent on a particular dataset. For example, mobile mapping scans from multiple cities could be used to achieve greater independence from the characteristics of a particular city. It could be analyzed to what extent cities in different parts of the world are similar and whether better results are obtained when networks are explicitly trained for different regions.

The presented image-based method for classifying road markings has also been used for utility hole covers. Similarly, the approach could certainly be applied to rail data and enable detecting rails, ties, balises, axle counters, and other ground-level objects in the track environment. Accordingly, the image-based approaches could be combined with rendered ground images and rendered scan profiles. For the analysis of scan profiles, in addition to the presented approach, it remains to be investigated whether one-dimensional CNNs can be used, such as those used in noise classification. Similarly, viewing multiple scan profiles simultaneously as parallel layers in an image could aid classification by providing contextual information.

In addition, future work could explore how multiple scans acquired in parallel or overlapping at intersections could be combined for analysis and what an appropriate visualization would need to look like for the analysis of overlapping GPR data.

The concepts and techniques presented in this thesis provide a broad basis for analyzing large-scale geospatial mobile mapping data of transport infrastructure. Increasingly frequent and detailed "snapshots" of our built environment by more and more users will continue to offer tremendous potential for advancing spatial visualization and analysis methods, with the ultimate goal of achieving real-time, semantic-rich spatial digital twins.

Bibliography

[1] Michael Georg Adam and Eckehard Steinbach. "PIU-Net: Generation of Virtual Panoramic Views from Colored Point Clouds". In: *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* (2021), 269–276.

[2] Ahmad Kamal Aijazi, Paul Checchin, and Laurent Trassoudaine. "Segmentation Based Classification of 3D Urban Point Clouds: A Super-Voxel Based Approach with Evaluation". In: *Remote Sensing* 5.4 (2013), 1624–1650.

[3] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. "A new Voronoi-based Surface Reconstruction Algorithm". In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998, 415–421.

[4] Ioannis Anagnostopoulos, Viorica Pătrăucean, Ioannis Brilakis, and Patricio Vela. "Detection of Walls, Floors, and Ceilings in Point Cloud Data". In: *Construction Research Congress 2016*. 2016, 2302–2311.

[5] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. "Google Street View: Capturing the World at Street Level". In: *Computer* 43.6 (2010), 32–38.

[6] Mostafa Arastounia. "An Enhanced Algorithm for Concurrent Recognition of Rail Tracks and Power Cables from Terrestrial and Airborne Lidar Point Clouds". In: *Building Information Modelling for Civil Infrastructures* 2.2 (2017), 8.

[7] Klement Aringer and Robert Roschlaub. "Bavarian 3D Building Model and Update Concept Based on LiDAR, Image Matching and Cadastre Information". In: *Innovations in 3D Geo-Information Sciences*. Springer, 2014, 143–157.

[8] ASPRS. *LASer (LAS) File Format Exchange Activities*. Accessed: 2021-05-18. 2019. URL: https://www.asprs.org/divisions-committees/lidar-division/laser-las-file-format-exchange-activities.

[9] ASTM International. *Standard Specification for 3D Imaging Data Exchange, Version 1.0*. Accessed: 2021-05-18. 2019. URL: https://www.astm.org/Standards/E2807.htm.

[10] Pouria Babahajiani, Lixin Fan, Joni-Kristian Kämäräinen, and Moncef Gabbouj. "Urban 3D Segmentation and Modelling from Street View Images and LiDAR Point Clouds". In: *Machine Vision and Applications* 28.7 (2017), 679–694.

[11] Eric Charles Barrett and Leonard Frank Curtis. *Introduction to Environmental Remote Sensing*. Psychology Press, 1999.

[12] Sebastian Bechtold and Bernhard Höfle. "HELIOS: A Multi-purpose Lidar Simulation Framework for Research, Planning and Training of Laser Scanning Operations with Airborne, Ground-based Mobile and Stationary Platforms". In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 3.3 (2016), 161–168.

[13] Andrea Benedetto, Fabio Tosti, Luca Bianchini Ciampoli, and Fabrizio D'Amico. "An Overview of Ground-Penetrating Radar Signal Processing Techniques for Road Inspections". In: *Signal Processing* 132 (2017), 201–209.

[14] David Bétaille and Rafael Toledo-Moreo. "Creating Enhanced Maps for Lane-Level Vehicle Navigation". In: *IEEE Transactions on Intelligent Transportation Systems* 11.4 (2010), 786–798.

[15] Andrea Biasion, Leandro Bornaz, and Fulvio Rinaudo. "Laser Scanning Applications on Disaster Management". In: *Geo-information for Disaster Management*. Springer, 2005, 19–33.

[16] Stefan Blaser, Stephan Nebiker, and Stefan Cavegn. "System Design, Calibration and Performance Analysis of a Novel 360° Stereo Panoramic Mobile Mapping System". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2017), 207–213.

[17] Christian Boucheny. "Interactive Scientific Visualization of Large Datasets: Towards a Perceptive-Based Approach". PhD thesis. Université Joseph Fourier, Grenoble, 2009.

[18] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. "SnapNet: 3D Point Cloud Semantic Labeling with 2D Deep Segmentation Networks". In: *Computers & Graphics* 71 (2018), 189–198.

[19] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. "Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks". In: *Proceedings of 3DOR*. Vol. 2. 2017, 1.

[20] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. "LOF: Identifying Density-Based Local Outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, 93–104.

[21] Matthew Brown and David G. Lowe. "Automatic Panoramic Image Stitching using Invariant Features". In: *International Journal of Computer Vision* 74 (2006), 59–73.

[22] H. Butler, D. C. Finnegan, P. J. Gadomski, and U. K. Verma. "plas.io: Open Source, Browser-based WebGL Point Cloud Visualization". In: *AGU Fall Meeting Abstracts*. 2014.

[23] Carlos Cabo, Celestino Ordoñez, Silverio García-Cortés, and J. Martínez. "An Algorithm for Automatic Detection of Pole-like Street Furniture Objects from Mobile Laser Scanner Point Clouds". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 87 (2014), 47–56.

[24] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. "Fast LIDAR-Based Road Detection Using Fully Convolutional Neural Networks". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, 1019–1024.

[25] Gabriella Caroti, Andrea Piemonte, and B. Pucci. "Terrestrial Laser Scanning as Road's Cadastre Revision And Integration Support". In: *ISPRS Workshop Italy-Canada 2005 3D Digital Imaging and Modeling: Applications of Heritage, Industry*. Vol. 1. CIRGEO. 2005, 1–3.

[26] Dong Chen, Ruisheng Wang, and Jiju Peethambaran. "Topologically Aware Building Rooftop Reconstruction from Airborne Laser Scanning Point Clouds". In: *IEEE TGRS* 55.12 (2017), 7032–7052.

[27] Shenchang Eric Chen. "Quicktime VR: An Image-based Approach to Virtual Environment Navigation". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. 1995, 29–38.

[28] Xin Chen, Brad Kohlmeyer, Matei Stroila, Narayana Alwar, Ruisheng Wang, and Jeff Bach. "Next Generation Map Making: Geo-referenced Ground-level LIDAR Point Clouds for Automatic Retro-reflective Road Feature Extraction". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2009, 488–491.

[29] Phuong Minh Chu, Seoungjae Cho, Jisun Park, Simon Fong, and Kyungeun Cho. "Enhanced Ground Segmentation Method for Lidar Point Clouds in Human-centric Autonomous Robot Systems". In: *Human-centric Computing and Information Sciences* 9.1 (2019), 17–30.

[30] Bernard Comment. *The Panorama*. Reaktion Books, 1999.

[31] Malvin Danhof, Tarek Schneider, Pascal Laube, and Georg Umlauf. "A Virtual-reality 3D-laser-scan Simulation". In: *BW-CAR Symposium on Information and Communication Systems* (2015), 68–73.

[32] J. L. Davis and A. P. Annan. "Ground-penetrating Radar for High-resolution Mapping of Soil and Rock Stratigraphy". In: *Geophysical prospecting* 37.5 (1989), 531–551.

[33] David Deibe, Margarita Amor, and Ramón Doallo. "Big Data Geospatial Processing for Massive Aerial LiDAR Datasets". In: *Remote Sensing* 12.4 (2020), 719.

[34] Jérôme Demantké, Bruno Vallet, and Nicolas Paparoditis. "Streamed Vertical Rectangle Detection in Terrestrial Laser Scans for Facade Database Production". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. I-3. September. 2012, 99–104.

[35] Emon Kumar Dey, Mohammad Awrangjeb, and Bela Stantic. “An Unsupervised Outlier Detection Method For 3D Point Cloud Data”. In: *IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2019, 2495–2498.

[36] Sören Discher, Rico Richter, and Jürgen Döllner. “A Scalable webGL-based Approach for Visualizing Massive 3D Point Clouds Using Semantics-dependent Rendering Techniques”. In: *Proceedings of Web3D*. 2018, 19:1–19:9.

[37] Sören Discher, Rico Richter, and Jürgen Döllner. “Interactive and View-Dependent See-Through Lenses for Massive 3D Point Clouds”. In: *Advances in 3D Geoinformation*. 2017, 49–62.

[38] Juergen Dold and Jessica Groopman. “The Future of Geospatial Intelligence”. In: *Geo-spatial information science* 20.2 (2017), 151–162.

[39] Jürgen Döllner. *Geospatial Artificial Intelligence: Potentials of Machine Learning for 3D Point Clouds and Geospatial Digital Twins*. Tech. rep. Hasso Plattner Institute, 2019.

[40] David H. Douglas and Thomas K. Peucker. “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature”. In: *Cartographica: the international journal for geographic information and geovisualization* 10.2 (1973), 112–122.

[41] Xiaolei Du and Yong Zhuo. “A Point Cloud Data Reduction Method Based on Curvature”. In: *2009 IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design*. IEEE. 2009, 914–918.

[42] Nira Dyn, Armin Iske, and Holger Wendland. “Meshfree Thinning of 3D Point Clouds”. In: *Foundations of Computational Mathematics* 8.4 (2008), 409–425.

[43] Jan U. H. Eitel, Bernhard Höfle, Lee A. Vierling, Antonio Abellán, Gregory P. Asner, Jeffrey S. Deems, Craig L. Glennie, Philip C. Joerg, Adam L. LeWinter, Troy S. Magney, Gottfried Mandlburger, Douglas C. Morton, Jörg Müller, and Kerri T. Vierling. “Beyond 3-D: The New Spectrum of Lidar Applications for Earth and Ecological Sciences”. In: *Remote Sensing of Environment* 186 (2016), 372–392.

[44] Niklas Elmqvist and Philippas Tsigas. “A Taxonomy of 3D Occlusion Management for Visualization”. In: *IEEE TVCG* 14.5 (2008), 1095–1109.

[45] ESRI. *ESRI Shapefile Technical Description*. Accessed: 2017-03-16. 1998. URL: `https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`.

[46] Robert D. Evans, Matthew W. Frost, Martyn Stonecliffe-Jones, and Neil Dixon. “A Review of Pavement Assessment using Ground Penetrating Radar (GPR)”. In: *Proceedings of the 12th International Conference on Ground Penetrating Radar* (2008).

[47] K. Fukano and H. Masuda. “Detection and Classification of Pole-Like Objects from Mobile Mapping Data”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 1 (2015), 57–64.

[48] Stefan Funke and Edgar A. Ramos. “Smooth-surface Reconstruction in Near-linear Time”. In: *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*. SODA ’02. Society for Industrial and Applied Mathematics, 2002, 781–790.

[49] *Geo Radar: 3D and GPR*. Accessed: 2018-06-25. 2005. URL: http://www.geo-radar.pl/en/methods/georadar/3d/.

[50] Geophysical Survey Systems. *Utility Scan Pro*. Accessed: 2021-01-13. 2018. URL: https://www.geophysical.com/products/utilityscan-pro.

[51] Luis Gézero and Carlos Antunes. “Automated Three-Dimensional Linear Elements Extraction from Mobile LiDAR Point Clouds in Railway Environments”. In: *Infrastructures* 4.3 (2019), 46.

[52] Antonis Giannopoulos. “Modelling Ground Penetrating Radar by GprMax”. In: *Construction and building materials* 19.10 (2005), 755–762.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[54] Prashant Goswami, Fatih Erol, Rahul Mukhi, Renato Pajarola, and Enrico Gobbetti. “An Efficient Multi-Resolution Framework for High Quality Interactive Rendering of Massive Point Clouds Using Multi-Way Kd-Trees”. In: *The Visual Computer* 29.1 (2013), 69–83.

[55] Jack Greenhalgh and Majid Mirmehdi. “Automatic Detection and Recognition of Symbols and Text on the Road Surface”. In: *International Conference on Pattern Recognition Applications and Methods*. Springer. 2015, 124–140.

[56] Adrien Gressin, Bertrand Cannelle, Clément Mallet, and Jean-Pierre Papelard. “Trajectory-based Registration of 3D LiDAR Point Clouds Acquired with a Mobile Mapping System”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 1 (2012), 117–122.

[57] E. Grilli, F. Menna, and F. Remondino. “A Review of Point Clouds Segmentation and Classification Algorithms”. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017), 339.

[58] Markus Gross and Hanspeter Pfister. *Point-based Graphics*. Morgan Kaufmann, 2011.

[59] Haiyan Guan, Jonathan Li, Shuang Cao, and Yongtao Yu. “Use of Mobile LiDAR in Road Information Inventory: A Review”. In: *International Journal of Image and Data Fusion* 7.3 (2016), 219–242.

[60] Haiyan Guan, Jonathan Li, Yongtao Yu, Cheng Wang, Michael Chapman, and Bisheng Yang. “Using Mobile Laser Scanning Data for Automated Extraction of Road Markings”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 87 (2014), 93–107.

[61] Norbert Haala and Stefan Cavegn. “High Density Aerial Image Matching: State-of-the-art and Future Prospects”. In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 41 (2016), 625–630.

[62] Ayman Habib, Mwafag Ghanma, Michel Morgan, and Rami Al-Ruzouq. “Photogrammetric and LiDAR Data Registration Using Linear Features”. In: *Photogrammetric Engineering & Remote Sensing* 71.6 (2005), 699–707.

[63] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan D. Wegner, Konrad Schindler, and Marc Pollefeys. “Semantic3D.net: A New Large-Scale Point Cloud Classification Benchmark”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* (2017), 91–98.

[64] Timo Hackel, Jan D. Wegner, and Konrad Schindler. “Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density”. In: *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences* 3 (2016), 177–184.

[65] Xian-Feng Han, Jesse S. Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, and Liping Xiao. “A Review of Algorithms for Filtering the 3D Point Cloud”. In: *Signal Processing: Image Communication* 57 (2017), 103–112.

[66] Alberto Hata and Denis Wolf. “Road Marking Detection Using LIDAR Reflective Intensity Data and its Application to Vehicle Localization”. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2014, 584–589.

[67] Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst. “Matrix Capsules with EM Routing”. In: *International Conference on Learning Representations*. 2018.

[68] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. “Detection of Traffic Signs in Real-world Images: The German Traffic Sign Detection Benchmark”. In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2013, 1–8.

[69] Xiangyun Hu and Lizhi Ye. “A Fast and Simple Method of Building Detection from Lidar Data Based on Scan Line Analysis”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2013), 7–13.

[70] Jing Huang and Suya You. “Point Cloud Labeling Using 3D Convolutional Neural Network”. In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, 2670–2675.

[71] Dryver R. Huston, Noel V. Pelczarski, Brian Esser, and Kenneth R. Maser. “Damage Detection in Roadways with Ground Penetrating Radar”. In: *Eighth International Conference on Ground Penetrating Radar*. Vol. 4084. 2000, 91–95.

[72] IOGP Geomatics Committee. *EPSG Geodetic Parameter Dataset*. Accessed: 2020-11-06. 2020. URL: https://epsg.org.

[73] Masoud Bakhtyari Kia, Saied Pirasteh, Biswajeet Pradhan, Ahmad Rodzi Mahmud, Wan Nor Azmin Sulaiman, and Abbas Moradi. "An Artificial Neural Network Model for Flood Simulation using GIS: Johor River Basin, Malaysia". In: *Environmental Earth Sciences* 67.1 (2012), 251–264.

[74] Holger Koss. "On Differences and Similarities of Applied Wind Comfort Criteria". In: *Journal of Wind Engineering and Industrial Aerodynamics* 94.11 (2006), 781–797.

[75] Werner Kuhn. "Core Concepts of Spatial Information for Transdisciplinary Research". In: *International Journal of Geographical Information Science* 26.12 (2012), 2267–2276.

[76] Amara Dinesh Kumar. "Novel Deep Learning Model for Traffic Sign Detection Using Capsule Networks". In: *arXiv preprint arXiv:1805.04424* (2018).

[77] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. "Convolutional Networks and Applications in Vision". In: *Proceedings of 2010 IEEE international symposium on circuits and systems*. IEEE. 2010, 253–256.

[78] Matti Lehtomäki, Anttoni Jaakkola, Juha Hyyppä, Antero Kukko, and Harri Kaartinen. "Detection of Vertical Pole-Like Objects in a Road Environment Using Vehicle-Based Laser Scanning Data". In: *Remote Sensing* 2.3 (2010), 641–664.

[79] Jianping Li, Bisheng Yang, Chi Chen, Ronggang Huang, Zhen Dong, and Wen Xiao. "Automatic Registration of Panoramic Image Sequence and Mobile Laser Scanning Data Using Semantic Features". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 136 (2018), 41–57.

[80] Rongxing Li. "Mobile Mapping: An Emerging Technology for Spatial Data Acquisition". In: *Photogrammetric Engineering and Remote Sensing* 63.9 (1997), 1085–1092.

[81] David G. Lowe. "Distinctive Image Features from Scale-invariant Keypoints". In: *International journal of computer vision* 60.2 (2004), 91–110.

[82] Steve Mann and Rosalind W. Picard. "Virtual Bellows: Constructing High Quality Stills from Video". In: *Proceedings of 1st International Conference on Image Processing*. Vol. 1. IEEE. 1994, 363–367.

[83] Oscar Martinez-Rubi, Maurice de Kleijn, Stefan Verhoeven, Niels Drost, Jisk Attema, Maarten van Meersbergen, Rob van Nieuwpoort, Rens de Hond, Eduardo Dias, and Pjotr Svetachov. "Using Modular 3D Digital Earth Applications Based on Point Clouds for the Study of Complex Sites". In: *International Journal of Digital Earth* 9.12 (2016), 1135–1152.

[84] Daniel Maturana and Sebastian Scherer. "VoxNet: A 3D Convolutional Neural Network for Real-time Object Recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, 922–928.

[85] Markus Maurer, J. Christian Gerdes, Barbara Lenz, and Hermann Winner. *Autonomous Driving: Technical, Legal and Social Aspects*. Springer, 2016.

[86] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. "VV-NET: Voxel VAE Net with Group Convolutions for Point Cloud Segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 8500–8508.

[87] Xuelian Meng, Le Wang, José Luis Silván-Cárdenas, and Nate Currit. "A Multi-directional Ground Filtering Algorithm for Airborne LIDAR". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 64.1 (2009), 117–124.

[88] Sławomir Mikrut, Piotr Kohut, Krystian Pyka, Regina Tokarczyk, Tomasz Barszcz, and Tadeusz Uhl. "Mobile Laser Scanning Systems for Measuring the Clearance Gauge of Railways: State of Play, Testing and Outlook". In: *Sensors* 16.5 (2016), 683.

[89] Niloy J. Mitra and An Nguyen. "Estimating Surface Normals in Noisy Point Cloud Data". In: *Proceedings of the nineteenth annual symposium on Computational geometry*. 2003, 322–328.

[90] Daniel Munoz, Nicolas Vandapel, and Martial Hebert. "Directional Associative Markov Network for 3-d Point Cloud Classification". In: *Fourth International Symposium on 3D Data Processing, Visualization and Transmission* (2008).

[91] AL-Rousan Nadia, Nor Ashidi Mat Isa, and Mohd Khairunaz Mat Desa. "Advances in Solar Photovoltaic Tracking Systems: A Review". In: *Renewable and sustainable energy reviews* 82 (2018), 2548–2569.

[92] Luis E. Navarro-Serment, Christoph Mertz, and Martial Hebert. "Pedestrian Detection and Tracking Using Three-Dimensional Ladar Data". In: *The International Journal of Robotics Research* 29.12 (2010), 1516–1528.

[93] Elisa Negri, Luca Fumagalli, and Marco Macchi. "A Review of the Roles of Digital Twin in CPS-based Production Systems". In: *Procedia Manufacturing* 11 (2017), 939–948.

[94] Anh Nguyen and Bac Le. "3D Point Cloud Segmentation: A Survey". In: *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE. 2013, 225–230.

[95] Huan Ni, Xiangguo Lin, and Jixian Zhang. "Classification of ALS Point Cloud with Improved Point Cloud Segmentation and Random Forests". In: *Remote Sensing* 9.3 (2017), 288.

[96] J. Niemeyer, F. Rottensteiner, and U. Soergel. "Conditional Random Fields for LiDAR Point Cloud Classification in Complex Urban Areas". In: *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences* 1.3 (2012), 263–268.

[97] Xiaojuan Ning, Fan Li, Ge Tian, and Yinghui Wang. "An Efficient Outlier Removal Method for Scattered Point Cloud Data". In: *PloS one* 13.8 (2018), e0201280.

[98] Andreas Nüchter, Oliver Wulf, Kai Lingemann, Joachim Hertzberg, Bernardo Wagner, and Hartmut Surmann. "3D Mapping with Semantic Knowledge". In: *RoboCup 2005: Robot Soccer World Cup IX*. Springer, 2006, 335–346.

[99] Abdul Nurunnabi, Geoff West, and David Belton. “Outlier Detection and Robust Normal-curvature Estimation in Mobile Laser Scanning 3D Point Cloud Data”. In: *Pattern Recognition* 48.4 (2015), 1404–1419.

[100] Bastian Oehler, Joerg Stueckler, Jochen Welle, Dirk Schulz, and Sven Behnke. “Efficient Multi-Resolution Plane Segmentation of 3D Point Clouds”. In: *International Conference on Intelligent Robotics and Applications*. Springer. 2011, 145–156.

[101] Umut Ozkaya and Levent Seyfi. “Deep Dictionary Learning Application in GPR B-scan Images”. In: *Signal, Image and Video Processing* 12.8 (2018), 1567–1575.

[102] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *the Journal of machine Learning research* 12 (2011), 2825–2830.

[103] Ruggero Pintus, Enrico Gobbetti, and Roberto Combet. “Fast and Robust Semi-Automatic Registration of Photographs to 3D Geometry”. In: *Proceedings of the 12th International conference on Virtual Reality, Archaeology and Cultural Heritage*. 2011, 9–16.

[104] Florent Poux and Roland Billen. “Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised Geometric and Relationship Featuring vs Deep Learning Methods”. In: *ISPRS International Journal of Geo-Information* 8.5 (2019), 213.

[105] Reinhold Preiner, Stefan Jeschke, and Michael Wimmer. “Auto Splats: Dynamic Point Cloud Visualization on the GPU”. In: *Proceedings of EGPGV*. 2012, 139–148.

[106] David Prochazka, Jana Prochazkova, and Jaromir Landa. “Automatic Lane Marking Extraction from Point Cloud into Polygon Map Layer”. In: *European Journal of Remote Sensing* 52 (2019), 26–39.

[107] Shi Pu, Martin Rutzinger, George Vosselman, and Sander Oude Elberink. “Recognizing Basic Structures from Mobile Laser Scanning Data for Road Inventory Studies”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 66.6 (2011), 28–39.

[108] I. Puente, H. González-Jorge, J. Martínez-Sánchez, and P. Arias. “Review of Mobile Mapping and Surveying Technologies”. In: *Measurement* 46.7 (2013), 2127–2145.

[109] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. “Real-time Computer Vision with OpenCV”. In: *Communications of the ACM* 55.6 (2012), 61–69.

[110] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. “Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 652–660.

[111] Charles R. Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. "Volumetric and Multi-view CNNs for Object Classification on 3D Data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 5648–5656.

[112] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems*. 2017, 5099–5108.

[113] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[114] Rico Richter, Markus Behrens, and Jürgen Döllner. "Object Class Segmentation of Massive 3D Point Clouds of Urban Areas Using Point Cloud Topology". In: *International Journal of Remote Sensing* 34.23 (Dec. 2013), 8408–8424.

[115] Rico Richter, Sören Discher, and Jürgen Döllner. "Out-of-Core Visualization of Classified 3D Point Clouds". In: *3D Geoinformation Science*. Springer, 2015, 227–242.

[116] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional Networks for Biomedical Image Segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, 234–241.

[117] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. "Paris-Lille-3D: A Large and High-quality Ground-truth Urban Point Cloud Dataset for Automatic Segmentation and Classification". In: *The International Journal of Robotics Research* 37.6 (2018), 545–557.

[118] Szymon Rusinkiewicz and Marc Levoy. "QSplat: A Multiresolution Point Rendering System for Large Meshes". In: *Proceedings of SIGGRAPH*. 2000, 343–352.

[119] Radu Bogdan Rusu and Steve Cousins. "3D is Here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation*. IEEE. 2011.

[120] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Andreas Holzbach, and Michael Beetz. "Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009*. IEEE. 2009, 3601–3608.

[121] Martin Rutzinger, Bernhard Höfle, Markus Hollaus, and Norbert Pfeifer. "Object-based Point Cloud Analysis of Full-waveform Airborne Laser Scanning Data for Urban Vegetation Classification". In: *Sensors* 8 (2008), 4505–4528.

[122] Martin Rutzinger, Arun Kumar Pratihast, Sander J. Oude Elberink, and George Vosselman. "Tree Modelling from Mobile Laser Scanning Data-Sets". In: *The Photogrammetric Record* 26.135 (2011), 361–372.

[123] Timo Saarenketo and Thomas Scullion. *Ground Penetrating Radar Applications on Roads and Highways*. Tech. rep. 1994.

[124] Timo Saarenketo and Thomas Scullion. “Road Evaluation with Ground Penetrating Radar”. In: *Journal of Applied Geophysics* 43.2-4 (2000), 119–138.

[125] Markus Schütz and Michael Wimmer. “High-quality Point-based Rendering Using Fast Single-pass Interpolation”. In: *2015 Digital Heritage*. Vol. 1. IEEE. 2015, 369–372.

[126] Markus Schütz and Michael Wimmer. “Rendering Large Point Clouds in Web Browsers”. In: *Proceedings of CESCG* (2015), 83–90.

[127] Monika Sester. “Analysis of Mobility Data–A Focus on Mobile Mapping Systems”. In: *Geo-Spatial Information Science* 23.1 (2020), 68–74.

[128] Michael Seufert, Julian Kargl, Johannes Schauer, Andreas Nüchter, and Tobias Hoßfeld. “Different Points of View: Impact of 3D Point Cloud Reduction on QoE of Rendered Images”. In: *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2020, 1–6.

[129] Lidan Shang, Qiushi Yang, Jianing Wang, Shubin Li, and Weimin Lei. “Detection of Rail Surface Defects Based on CNN Image Recognition and Classification”. In: *20th International Conference on Advanced Communication Technology (ICACT)* (2018), 45–51.

[130] Stephan Sigg, Raphael Fuchs, Robert Carnecky, and Ronald Peikert. “Intelligent Cutaway Illustrations”. In: *Proceedings of PacificVis*. 2012, 185–192.

[131] Tom Simonite. *Self-Driving Cars’ Spinning-Laser Problem*. Accessed: 2020-12-04. Mar. 2017. URL: https://www.technologyreview.com/2017/03/20/153129/autonomous-cars-lidar-sensors.

[132] Lance Simons, Stewart He, Peter Tittman, and Nina Amenta. “Point-based Rendering of Forest LiDAR”. In: *Proceedings of EnvirVis*. 2014, 19–23.

[133] Mario Soilán, Belén Riveiro, Joaquín Martínez-Sánchez, and Pedro Arias. “Traffic Sign Detection in MLS Acquired Point Clouds for Geometric and Image-based Semantic Inventory”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 114 (2016), 92–101.

[134] Hao Song and Hsi-Yung Feng. “A Progressive Point Cloud Simplification Algorithm with Preserved Sharp Edge Data”. In: *The International Journal of Advanced Manufacturing Technology* 45.5-6 (2009), 583–592.

[135] Stadt Berlin. *Pressemitteilung: DGM-Erzeugung durch LiDAR-Befliegung*. Accessed: 2021-01-13. 2020. URL: https://www.stadtentwicklung.berlin.de/aktuell/pressebox/archiv_volltext.shtml?arch_2012/nachricht7004.html.

[136] Stadt Essen. *Für die Vermessung der Stadt: Essen punktet mit neuem Messfahrzeug*. Accessed: 2020-12-10. 2017. URL: https://www.essen.de/meldungen/pressemeldung_1081249.de.html.

[137] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. “Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition”. In: *Neural Networks* 32 (2012), 323–332.

[138] D. Stein, M. Spindler, J. Kuper, and M. Lauer. “Rail Detection Using Lidar Sensors”. In: *International Journal of Sustainable Development and Planning* 11.1 (2016), 65–78.

[139] Arjen Swart, Jonathan Broere, Remco Veltkamp, and Robby Tan. “Refined Non-rigid Registration of a Panoramic Image Sequence to a LiDAR Point Cloud”. In: *ISPRS Conference on Photogrammetric Image Analysis*. Springer. 2011, 73–84.

[140] Richard Szeliski and Heung-Yeung Shum. “Creating Full View Panoramic Image Mosaics and Environment Maps”. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), 251–258.

[141] Masood Taheriandani. “The Application of Doppler LIDAR Technology for Rail Inspection and Track Geometry Assessment”. PhD thesis. Virginia Tech, 2016.

[143] Mohamed Lamine Tazir, Paul Checchin, and Laurent Trassoudaine. “Color-based 3D Point Cloud Reduction”. In: *14th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (2016), 1–7.

[142] Mohamed Lamine Tazir, Paul Checchin, and Laurent Trassoudaine. “Color-based 3D Point Cloud Reduction”. In: *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE. 2016, 1–7.

[144] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. “RGCNN: Regularized Graph CNN for Point Cloud Segmentation”. In: *Proceedings of the 26th ACM international conference on Multimedia*. 2018, 746–754.

[145] Jochen Teizer, Changwan Kim, Carl Haas, Katherine Liapi, and Carlos Caldas. “Framework for Real-time Three-dimensional Modeling of Infrastructure”. In: *TRR Journal* 1913 (2005), 177–186.

[146] Takashi Toyama, Nozomi Nagamine, Tatsuya Omori, Kenichi Kitao, and Ryuta Nakasone. “Structure Gauge Measuring Equipment Using Laser Range Scanners and Structure Gauge Management System”. In: *Quarterly Report of RTRI* 60.1 (2019), 40–45.

[147] Matthias Trapp, Tassilo Glander, Henrik Buchholz, and Jürgen Döllner. “3D Generalization Lenses for Interactive Focus+context Visualization of Virtual City Models”. In: *Proceedings of International Conference Information Visualisation*. 2008, 356–361.

[148] Rudolph Triebel, Kristian Kersting, and Wolfram Burgard. “Robust 3D Scan Point Classification Using Associative Markov Networks”. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE. 2006, 2603–2608.

[149] Mikael Vaaraniemi, Martin Freidank, and Rüdiger Westermann. “Enhancing the Visibility of Labels in 3D Navigation Maps”. In: *Progress and New Trends in 3D Geoinformation Sciences*. 2013, 23–40.

[150] Stefan Vacek, Constantin Schimmel, and Rüdiger Dillmann. “Road-marking Analysis for Autonomous Vehicle Guidance”. In: *EMCR*. 2007, 1–6.

[151] Thomas Veit, Jean-Philippe Tarel, Philippe Nicolle, and Pierre Charbonnier. "Evaluation of Road Marking Feature Extraction". In: *2008 11th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2008, 174–181.

[152] Paul Viola and Michael Jones. "Rapid Object Detection Using a Boosted Cascade of Simple Features". In: *CVPR* 1 (2001), 511–518.

[153] Anh-Vu Vo, Linh Truong-Hong, Debra F. Laefer, and Michela Bertolotto. "Octree-based Region Growing for Point Cloud Segmentation". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 104 (2015), 88–100.

[154] George Vosselman. "3D Reconstruction of Roads and Trees for City Modelling". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Dresden, Germany* 34 (2003), 3.

[155] George Vosselman. "Point Cloud Segmentation for Urban Scene Classification". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL-7/W2 (2013), 257–262.

[156] Tasha Wade and Shelly Sommer. *A to Z GIS: An Illustrated Dictionary of Geographic Information Systems*. ESRI Press, 2006.

[157] Caihua Wang, Hideki Tanahashi, Hidekazu Hirayu, Yoshinori Niwa, and Kazuhiko Yamamoto. "Comparison of Local Plane Fitting Methods for Range Data". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE. 2001, 663–669.

[158] Dingkang Wang, Connor Watkins, and Huikai Xie. "MEMS Mirrors for LiDAR: A Review". In: *Micromachines* 11.5 (2020), 456.

[159] Qian Wang and Min-Koo Kim. "Applications of 3D Point Cloud Data in the Construction Industry: A Fifteen-year Review from 2004 to 2018". In: *Advanced Engineering Informatics* 39 (2019), 306–319.

[160] Weimin Wang, Ken Sakurada, and Nobuo Kawaguchi. "Incremental and Enhanced Scanline-Based Segmentation Method for Surface Reconstruction of Sparse LiDAR Data". In: *Remote Sensing* 8.11 (2016), 967.

[161] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds". In: *ACM Transactions On Graphics (TOG)* 38.5 (2019), 146.

[162] Chenglu Wen, Xiaotian Sun, Jonathan Li, Cheng Wang, Yan Guo, and Ayman Habib. "A Deep Learning Framework for Road Marking Extraction, Classification and Completion from Mobile Laser Scanning Point Clouds". In: *ISPRS journal of photogrammetry and remote sensing* 147 (2019), 178–192.

[163] Konrad Wenzel, Mohammed Abdel-Wahab, Alessandro Cefalu, and Dieter Fritsch. "High-resolution Surface Reconstruction from Imagery for Close Range Cultural Heritage Applications". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 39-B5 (2012), 133–138.

[164] Lukas Winiwarter, Gottfried Mandlburger, Stefan Schmohl, and Norbert Pfeifer. "Classification of ALS Point Clouds Using End-to-End Deep Learning". In: *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 87.3 (2019), 75–90.

[165] Holger Wirth. "Der neue Lichtraummesszug LIMEZ III der Deutschen Bahn AG". In: *Zeitschrift für Geodäsie, Geoinformation und Landmanagement* 133.3 (2008), 180–186.

[166] Johannes Wolf, Sören Discher, and Jürgen Döllner. "Techniken zur kombinierten Darstellung von 2D-Bodenradar und 3D-Punktwolken zur Analyse des Straßenraums". In: *Tagungsband der 39. Wissenschaftlich-Technischen Jahrestagung der DGPF e.V.* (2019), 154–166.

[167] Johannes Wolf, Sören Discher, Leon Masopust, Sebastian Schulz, Rico Richter, and Jürgen Döllner. "Combined Visual Exploration of 2D Ground Radar and 3D Point Cloud Data for Road Environments". In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 42.4/W10 (2018), 231–236.

[168] Johannes Wolf, Tobias Pietz, Rico Richter, Sören Discher, and Jürgen Döllner. "Image-Based Road Marking Classification in Mobile Mapping 3D Point Clouds". In: *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* (2021), 227–234.

[169] Johannes Wolf, Rico Richter, Sören Discher, and Jürgen Döllner. "Applicability of Neural Networks for Image Classification on Object Detection in Mobile Mapping 3D Point Clouds". In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 42.4/W15 (2019), 111–115.

[170] Johannes Wolf, Rico Richter, and Jürgen Döllner. "Asset Detection in Railroad Environments using Deep Learning-based Scanprofile Analysis". In: *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* (2021), 465–470.

[171] Johannes Wolf, Rico Richter, and Jürgen Döllner. "Techniques for Automated Classification and Segregation of Mobile Mapping 3D Point Clouds". In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* (2019), 201–208.

[172] Johannes Wolf, Rico Richter, and Jürgen Döllner. "Verwendung künstlicher neuronaler Netzwerke zur Bilderkennung für die Analyse von Straßenmarkierungen in Mobile-Mapping-3D-Punktwolken". In: *Tagungsband der 40. Wissenschaftlich-Technischen Jahrestagung der DGPF e.V.* 29 (2020), 219–229.

[173] Yuxing Xie, Jiaojiao Tian, and Xiao Xiang Zhu. "Linking Points with Labels in 3D: A Review of Point Cloud Semantic Segmentation". In: *IEEE Geoscience and Remote Sensing Magazine* 8.4 (2020), 38–59.

[174] Li Yan, Hua Liu, Junxiang Tan, Zan Li, Hong Xie, and Changjun Chen. "Scan Line Based Road Marking Extraction from Mobile LiDAR Point Clouds". In: *Sensors* 16.6 (2016), 903.

[175] Song Yanan, Zhang Hui, Liu Li, and Zhong Hang. "Rail Surface Defect Detection Method Based on YOLOv3 Deep Learning Networks". In: *2018 Chinese Automation Congress (CAC)*. IEEE. 2018, 1563–1568.

[176] Bisheng Yang, Lina Fang, Qingquan Li, and Jonathan Li. "Automated Extraction of Road Markings from Mobile LiDAR Point Clouds". In: *Photogrammetric Engineering & Remote Sensing* 78.4 (2012), 331–338.

[177] Yang Yang, Wensheng Zhang, Zewen He, and Ding Li. "High-Speed Rail Pole Number Recognition through Deep Representation and Temporal Redundancy". In: *Neurocomputing* 415 (2020), 201–214.

[178] Wei Yao and Hongchao Fan. "Automated Detection of 3D Individual Trees along Urban Road Corridors by Mobile Laser Scanning Systems". In: *Proceedings of International Symposium on Mobile Mapping Technology (MMT)*. Vol. 13. 2013, 1–6.

[179] Richard Yelf. "Where is True Time Zero?" In: *Proceedings of the Tenth International Conference on Grounds Penetrating Radar*. Vol. 1. IEEE. 2004, 279–282.

[180] Manzhu Yu, Chaowei Yang, and Yun Li. "Big Data in Natural Disaster Management: A Review". In: *Geosciences* 8.5 (2018), 165.

[181] Yongtao Yu, Jonathan Li, Haiyan Guan, Fukai Jia, and Cheng Wang. "Learning Hierarchical Features for Automated Extraction of Road Markings from 3-D Mobile LiDAR Point Clouds". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.2 (2014), 709–726.

[182] Ke Zhang, Marcus Hutter, and Huidong Jin. "A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data". In: *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. 2009, 813–822.

[183] Long Zhang, Qian Sun, and Ying He. "Splatting Lines: An Efficient Method for Illustrating 3D Surfaces and Volumes". In: *Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 2014, 135–142.

[184] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. "Road Extraction by Deep Residual U-net". In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018), 749–753.

[185] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end Learning for Point Cloud Based 3D Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 4490–4499.

[186] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. "Traffic-sign Detection and Classification in the Wild". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 2110–2118.

[187] Stefania Zourlidou and Monika Sester. "Traffic Regulator Detection and Identification from Crowdsourced Data—A Systematic Literature Review". In: *ISPRS International Journal of Geo-Information* 8.11 (2019), 491.

Statutory Declaration

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The thesis was not used in the same or in a similar version to achieve an academic grading and was not being published elsewhere.

Potsdam, June 29, 2021
(Place, Date)

(Signature)