DIGITAL ENGINEERING FAKULTÄT
Fachgebiet Computergrafische Systeme

Konzepte und Techniken zur Evaluierung des Straßenzustands von Mobile-Mapping-Punktwolken

Concepts and Methods for Evaluating Pavement Condition of Mobile Mapping Point Clouds

Bachelorarbeit
zur Erlangung des akademischen Grades
"Bachelor of Science"
(B.Sc.)
im Studiengang IT-Systems Engineering

Hasso-Plattner-Institut // Digital Engineering Fakultät // Universität Potsdam

vorgelegt von
Adrian Ziupka

Aufgabenstellung und Anleitung:
Prof. Dr. Jürgen Döllner
Dr. Rico Richter
Leon Masopust Sören Discher

Potsdam,
21. Juli 2021

Zusammenfassung

3D-Mobile-Mapping-Punktwolken können die Realität in einer Präzision von wenigen Millimetern genau abbilden. Deren unstrukturierte Natur macht es aber herausfordernd, direkt anhand der Punkte Objektklassifizierungen vorzunehmen. Gleichzeitig bieten vermehrt verfügbare Laserscan-Technologie sowie die erhöhte Rechenleistung der letzten Jahre Möglichkeiten, diese großen Datenmengen effizient zu verarbeiten.
Ein möglicher Anwendungsfall stellt die automatisierte Erkennung von Straßenschäden in Punktwolken dar, die zuvor mit speziell ausgerüsteten Fahrzeugen aufgenommen wurden. Eine solche semantische Segmentierung, also Einteilung nach Klassen, etwa von Schlaglöchern oder Flickstellen bietet das Potenzial, die bisher meist manuell und damit kostenintensiv durchgeführten Absuchungen nach Schäden durch Straßenbaubehörden zu ergänzen oder abzulösen. Die Analyse auf den 3D-Daten bietet dabei, im Gegensatz etwa zu Bildern der Punktwolke, zusätzliche wertvolle Tiefeninformationen.
In dieser Arbeit werden zwei Ansätze zur Detektion von Schäden auf direkter Punktbasis erläutert und miteinander verglichen. Der erste Ansatz basiert auf wohlüberlegten und durch Experimente verfeinerten Features, welche die lokale Nachbarschaft jedes Punktes charakterisieren. Dem zweiten Ansatz liegt das in der Forschung zu semantischer Segmentierung von Punktwolken bekannte neuronale Netz *PointNet* zugrunde, das ähnliche Features versucht automatisch aus den Nachbarschaften zu extrahieren.
Dabei zeigte sich, dass

hier Zmsf. des Vergleichs der Ansätze bzgl. Qualität und Performance

Inhaltsverzeichnis

Zusammenfassung iii

1 Einleitung 1
- 1.1 Terminologie . . . 1
  - 1.1.1 Punktwolke . . . 1
  - 1.1.2 Laserscanner . . . 1
  - 1.1.3 Photogrammetrie . . . 1
  - 1.1.4 Mobile-Mapping . . . 2
  - 1.1.5 Machine Learning . . . 2
- 1.2 Projektüberblick . . . 2
- 1.3 Schwerpunkte des Bachelorprojekts . . . 3
  - 1.3.1 Punktwolken-Fingerprint . . . 3
  - 1.3.2 Modelltraining . . . 3
  - 1.3.3 Straßenzustandserkennung . . . 3
  - 1.3.4 Webplattform . . . 4
- 1.4 Einordnung in den Projektkontext . . . 4
- 1.5 Struktur der Arbeit . . . 4

2 Verwandte Arbeiten 7
- 2.1 Objekterkennung in Punktwolken . . . 7
- 2.2 Straßenschadenerkennung auf Bildbasis . . . 7
- 2.3 Straßenschadenerkennung auf direkter Punktbasis . . . 7

3 Konzepte 9
- 3.1 Betrachtete Objektklassen . . . 9
- 3.2 Preprocessing . . . 11
- 3.3 Ansatz Feature-Extraction . . . 12
  - 3.3.1 Features . . . 12
  - 3.3.2 Scales . . . 13
  - 3.3.3 Ein- und mehrwertige Features . . . 14
  - 3.3.4 Uniqueness . . . 15
  - 3.3.5 Approximation für größere Scales . . . 16
  - 3.3.6 Prediction per Random Forest . . . 17
- 3.4 Ansatz Deep Learning . . . 18
- 3.5 Postprocessing . . . 19

4 Implementierung **21**
4.1 Systemüberblick . . . 21
4.2 Preprocessing . . . 22
4.2.1 Straßenextraktion . . . 22
4.2.2 Intensitätsnormalisierung . . . 24
4.3 Ansatz Feature-Extraction . . . 24
4.3.1 Features . . . 25
4.3.2 Scales . . . 26
4.3.3 Ein- und mehrwertige Features . . . 27
4.3.4 Uniqueness . . . 28
4.3.5 Approximation für größere Scales . . . 29
4.3.6 Prediction per Random Forest . . . 29
4.4 Ansatz Deep Learning . . . 30
4.5 Postprocessing . . . 31

5 Evaluierung **33**
5.1 Trainings- und Testdaten . . . 33
5.2 Metriken . . . 33
5.3 Preprocessing . . . 34
5.4 Ansatz Feature-Extraction . . . 34
5.4.1 Ergebnisse . . . 34
5.4.2 Vergleich mit nur einwertigen Features . . . 34
5.4.3 Vergleich mit größeren Scales . . . 34
5.4.4 Vergleich ohne Uniqueness . . . 34
5.5 Ansatz Deep Learning - Ergebnisse . . . 34
5.6 Postprocessing . . . 34

6 Fazit und Ausblick **35**

Literaturverzeichnis **37**

Kapitel 1

Einleitung

1.1 Terminologie

Terminologie beenden

1.1.1 Punktwolke

Eine Punktwolke ist eine Datenstruktur, welche als Menge von Punkten in einem n-dimensionalen Raum definiert ist. Zusätzlich zu den Koordinaten, die die Position im Raum beschreiben, werden bei der Erstellung einer Punktwolke häufig noch weitere Attribute wie der Intensitäts- oder Farbwert aufgenommen. Des Weiteren lassen sich über gängige Algorithmen zusätzliche Pro-Punkt -Eigenschaften wie die Oberflächennormale [9] oder ein Krümmungswert [5] berechnen. Meist werden Punktwolken mit Hilfe von 3D-Laserscannern direkt oder mittels photogrammetrischen Verfahren aus Bildern [10] erzeugt. Je nach verwendetem Verfahren können die resultierenden Punktwolken kleine Objekte, ganze Gebäude, Straßenzüge oder sogar Landschaften als virtuelles Modell abbilden.

1.1.2 Laserscanner

LiDAR-Scanner [2] verwenden Laserstrahlen, um eine Umgebung abzutasten. Wenn diese Strahlen auf eine Oberfläche treffen, werden sie reflektiert, was der Scanner registriert und so die Position des Reflexionspunktes ermitteln kann. Es können so aber auch andere Eigenschaften erfasst werden, wie Helligkeit bzw. Reflexionsfähigkeit der abgetasteten Oberfläche (die sogenannte *Intensität*), ihre Farbe oder die von der Oberfläche entsandte Infrarot-Strahlung. Letztere kann beispielsweise zum Klassifizieren von Bäumen von zentraler Bedeutung sein. Der Scanner erfasst die Oberflächen dabei jedoch nicht als Ganzes, sondern lediglich 3D-Punkte, welche in Punktwolken abgespeichert werden. Um später aus diesen Punkten wieder Semantik etwa durch Klassifizierungen zu gewinnen, existieren verschiedene Verfahren und Algorithmen.

1.1.3 Photogrammetrie

Bei der Photogrammetrie [7] wird aus 2D-Daten wie Bildern und Videos die räumliche Position von Objekten geschätzt. Dabei stützt man sich vor allem auf das Prinzip der Stereoskopie aus der Biologie. Menschen beispielsweise sind in der Lage die Position von Objekten im Raum zu bestimmen, da diese Objekte leicht versetzt sind in den

Blickfeldern ihrer Augen. Die Photogrammetrie nutzt auf gleiche Weise eine Anzahl an versetzt aufgenommenen Bildern, um so die Distanz von Objekten zur Kamera zu bestimmen. Die so gewonnenen 3D-Darstellungen von Objekten werden dann in Punktwolken konvertiert, welche dann wiederum weiterverarbeitet werden können.

1.1.4 Mobile-Mapping

Bei einer Mobile-Mapping Punktwolke wurden die Punktdaten durch ein spezielles Fahrzeug XXX

1.1.5 Machine Learning

Unter Deep Learning versteht man das Anwenden eines lernenden, mehrschichtigen künstlichen neuronalen Netzes auf eine komplexe Problemstellung, welche sich meist nur schwer oder auch gar nicht mit nicht lernenden Algorithmen lösen lässt. Dabei orientiert man sich stark an der Funktionsweise von Nervenzellen im menschlichen Gehirn.

1.2 Projektüberblick

3D-Punktwolken können mit Hilfe von Laserscannern oder photogammetrischen Verfahren mit vergleichsweise geringem Aufwand zeit- und kosteneffizient erzeugt werden. Weiterhin enthalten Punktwolken als häufig bis auf wenige Millimeter genaues Modell der realen Welt viele Informationen wie Position oder Farbe der dargestellten Objekte. Entsprechend kann das Modell, je nach Detailgrad und Verfügbarkeit weiterer Attribute, genutzt werden, um Erkenntnisse über Gegebenheiten der realen Welt zu erlangen. Für die Extraktion dieser Informationen werden allerdings auch geeignete Algorithmen benötigt: Besonders wichtig ist dabei, dass diese Algorithmen effizient sind, um die großen Datenmengen, die Punktwolken in sich tragen, verarbeiten zu können. Es existieren jedoch auch Problemstellungen, welche algorithmisch nur vergleichsweise ineffizient gelöst werden können, weshalb die Verarbeitung von Punktwolken mittels Machine Learning und vor allem Deep Learning immer häufiger Einsatz findet. Für diese Fälle wird allerdings generell leistungsfähige Hardware - insbesondere moderne CPUs, GPUs sowie RAM in ausreichender Menge - und viel Rechenzeit benötigt. Insofern gibt es Anwendungsfälle, die nur dann sinnvoll umgesetzt werden können, wenn der verwendete Algorithmus in ausreichender Form skalierbar ist.
Weiterhin sind viele der Algorithmen sehr komplex, um auch verschiedene Arten von Punktwolken mit unterschiedlichen Charakteristika wie ihrer Dichte, aber auch Pro-Punkt-Eigenschaften wie der Intensität und Farbwerten sinnvoll verarbeiten zu können. Dies setzt voraus, dass der Anwender über ein tiefgreifendes Verständnis sowohl vom Aufbau der Punktwolken selbst als auch von verfügbaren Algorithmen und ihren Parametern verfügt. Daraus folgt jedoch, dass Nutzer mit weniger fundiertem Fachwissen nur begrenzte Möglichkeiten haben, Punktwolken vor allem für komplexere Anwendungsfälle gewinnbringend zu nutzen.

Dementsprechend soll für diese Problemstellung eine Plattform entwickelt werden, die sowohl die nötige Skalierbarkeit zur Anwendung hochkomplexer Algorithmen und Deep-Learning-Verfahren bietet als auch eine entsprechend hohe Nutzerfreundlichkeit sicherstellt.

1.3 Schwerpunkte des Bachelorprojekts

Im Projektverlauf sind diverse Analyse- und Verarbeitungsverfahren für 3D-Punktwolken sowie eine webbasierte Plattform zur Bündelung dieser und bereits vorhandener Funktionalitäten entstanden. Die einzelnen Schwerpunkte werden im Folgenden kurz vorgestellt.

1.3.1 Punktwolken-Fingerprint

Since massive 3D point clouds easily contain millions of points, elaborate operations on them can be very time consuming. However there are also operations which are not executed per point, but rather generate only one result for the whole point cloud. To reduce the time consumption of such operations, a fingerprint of a point cloud can be computed, which has only a fraction of the size of the original point cloud. This fingerprint is then used for further processing steps instead of the point cloud. There are different possibilities how a fingerprint can be created. In this project the fingerprint consists of a collection of histograms, called PCFP. The PCFP is then used to perform operations like a point cloud classification using a deep neural network or parameter estimation based on parameters of similar point clouds.

1.3.2 Modelltraining

Für eine schnelle und gute Schätzung von Parametern für verschiedene Analysen auf noch ungesehenen Punktwolken benötigen wir ein Set an Parametern und Modellen, die auf bereits vorhandenen Punktwolken aufwendig berechnet oder trainiert werden müssen. Dafür haben wir Mechanismen entwickelt, die geometrische Eigenschaften von Punktwolken extrahieren und daraus mittels Heuristiken Eingabeparameter für beispielsweise eine Bodenerkennung ermitteln können. Daneben haben wir eine genetische Hyperparameteroptimierung für Deep Learning Modelle zur semantischen Segmentierung auf Punktwolken implementiert, damit wir für Punktwolken mit vorannotierten Klassen die besten Trainingsparameter finden und so auf unannotierten Punktwolken optimierte Modelle anwenden können.

1.3.3 Straßenzustandserkennung

Moderne Laserscanner, die auf einem entsprechenden Fahrzeug montiert sind, bieten höchste Präzision und können Straßenzüge millimetergenau erfassen. Damit wird eine automatische Analyse dieser Straßen-Punktwolken motiviert, die auch kleine Schäden wie Schlaglöcher und feine Ausbesserungen wie Flickstellen identifizieren soll. Herausfordernd sind dabei vor allem die hohe Punktdichte und die unregelmäßige Natur solcher Schäden.

In diesem Projektteil werden zwei Ansätze getestet und verglichen: ein klassischer Machine-Learning-Ansatz mit der Extraktion manuell definierter Features sowie ein moderner Deep-Learning-Ansatz, der aber ebenfalls direkt auf Punktmengen arbeitet.

1.3.4 Webplattform

Die sinnvolle Analyse und Verarbeitung von 3D-Punktwolken erfordert aktuell häufig komplexe Verarbeitungsprozesse und sehr leistungsstarke Hardware. Durch die während des Projekts entwickelte webbasierte Plattform werden verschiedene Anwendungen in einem einzelnen Projekt gebündelt und um eine hohe Skalierbarkeit mittels Lastenverteilung auf viele Rechner in einem Cluster erweitert. Dies ermöglicht auf der einen Seite eine Vereinfachung der Verarbeitungsprozesse, was die Plattform auch für Nutzer mit weniger fundiertem Fachwissen nutzbar macht, und auf der anderen Seite eine Beschleunigung bei der Verarbeitung vieler 3D-Punktwolken durch die Verteilung der nötigen Berechnungen auf alle verfügbaren Rechner.

1.4 Einordnung in den Projektkontext

Die potenzielle Menge an Anwendungsfällen von Punktwolken ist groß. Entsprechend sollte auch die Webplattform neben ihrer einfachen und intuitiven Bedienung bereits entsprechende nützliche Funktionalität bieten. Abseits von zumindest teilweise ausgereiften Verfahren wie der Bodenerkennung bot sich die Erkennung von Straßenschäden in Punktwolken aus zweierlei Gründen an:

- Die Genauigkeit des *Mobile Mapping* von Bodenfahrzeugen erlaubt es - im Gegensatz zu eher groberen Luftscans - grundsätzlich, auch kleinere und feinere Schäden oder sonstige Objekte von Interesse zu identifizieren.
- Die Suche nach Straßenschäden durch etwa Straßenbaubehörden ist meist noch immer eine manuelle Aufgabe. Ob die Schäden bei einer Fahrt durch die Stadt durch bloßen Blick gesichtet werden oder bei Ansehen von aufgenommenen Videos der Fahrbahnoberfläche: Solche Methoden sind personal- und zeitaufwändig und naturgemäß trotzdem fehleranfällig.

Diese Anwendung kann also für einige Nutzer der Plattform sinnvoll sein. Entsprechend wurde sie darin integriert und kann durch eine einfache Bestätigung gestartet werden. Wie bei anderen Aktionen auf Punktwolken werden auch in diesem Fall die notwendigen Schritte gestartet und, wo immer möglich, verteilt auf verschiedenen Rechnern ausgeführt.

1.5 Struktur der Arbeit

Die restliche Arbeit gliedert sich wie folgt: In Kapitel 2 werden verwandte Arbeiten vorgestellt, anschließend folgen in Kapitel 3 Erläuterungen zu den Konzepten zur Straßenzustandserkennung. Kapitel 4 behandelt die Implementierung jener Konzepte, während

in Kapitel 5 eine Evaluierung der beiden getesteten Ansätze vorgenommen wird. Das abschließende Kapitel 6 stellt ein Fazit zu den Ergebnissen und gibt einen Ausblick auf mögliche Verbesserungen und Erweiterungen.

Kapitel 2

Verwandte Arbeiten

2.1 Objekterkennung in Punktwolken

2.2 Straßenschadenerkennung auf Bildbasis

2.3 Straßenschadenerkennung auf direkter Punktbasis

Kapitel 3

Konzepte

Das Ziel dieser Arbeit ist eine hinreichend genaue und effiziente Erkennung von Straßenschäden in 3D-Punktwolken auf Basis einzelner Punkte. Training und Evaluierung sind auf einer gescannten Asphaltstraße durchgeführt worden. Etwaige Bilder von Punktwolken oder Objekten in diesen entstammen der Trainings- oder Test-Punktwolke, die im Kapitel *Evaluierung* genauer vorgestellt werden. Die grundlegenden Ideen und Techniken hinter den dafür nötigen Verarbeitungsschritten sollen in diesem Kapitel erläutert werden. Dabei wird sowohl auf den eigens konstruierten Ansatz per Feature-Extraction eingegangen als auch auf einen auf Deep Learning fußenden. Auch das beiden Ansätzen gemeine Pre- und Postprocessing wird kurz dargestellt.

3.1 Betrachtete Objektklassen

Entsprechend des Ziels nehmen Objekte, die auf einen mangelhaften Straßenzustand hinweisen, den vornehmlichen Fokus ein. Einfluss auf die Entstehung von Schäden verschiedener Art haben unter anderem der Niederschlag, temperaturbedingte Verformungen sowie die Verkehrsbelastung selbst.[1] Es existieren Leitfäden[2] zur empfohlenen Auswertung solcher *Substanzmerkmale*, an denen sich unter anderem bzgl. der verschiedenen Klassen orientiert wurde. Diese Klassen sind zum besseren Verständnis als Bilder der Punktwolke in Abbildung X dargestellt.

Was anderes als Fußnoten nutzen?

Darunter fallen insbesondere merkliche Absenkungen wie Schlaglöcher oder Risse. Schlaglöcher zeichnen sich durch lokale Höhenunterschiede, welche von mehreren Millimetern bis hin zu einigen Zentimetern reichen können, sowie ihre meist eher rundliche Form aus. Solche Absenkungen stellen gerade bei höheren Geschwindigkeiten und entsprechender Tiefe eine erhebliche Gefahr für die Kontrolle des Fahrzeugs und somit ein potenzielles Unfallrisiko dar.
Auch Flickstellen bilden eine interessante Klasse: Sie werden angebracht, um beschädigte Straßenteile - wie zum Beispiel ein Schlagloch - zu überdecken und auf diese Weise ungefährlich zu machen. Charakterisiert werden sie durch ihre wegen der maschinellen Fertigung meist rechteckige Struktur sowie der im Vergleich zur unmittelbaren Umgebung oft dunkler erscheinenden Fugendichtmasse. Letztere kann durch Überquellen ebenfalls für marginale Höhenunterschiede sorgen. Auch wenn Flickstellen selbst also keinen Schaden

[1] https://www.ise.kit.edu/rd_download/SBT/Kolloquium_SBT_2011-11-23_C.Karcher.pdf
[2] https://itzeb.heller-ig.de/leitfaden/

darstellen, ist ein Straßenabschnitt, der zu großen Teilen von Flickstellen überzogen ist, ein möglicher Hinweis auf eine sinnvolle Komplettreparatur.
Gullys und Kanaldeckel kommen in vielen Ausprägungen bezüglich Größe und Form daher. Obwohl diese selbstverständlich begründet angelegt wurden, zeichnen auch sie sich durch charakteristische Höhenunterschiede aus, was einen Vergleich unter anderem mit Schlaglöchern und somit eine genauere Betrachtung wert ist. Ein zuverlässiges Erkennen von Gullys kann außerdem, wie im Unterabschnitt *Uniqueness* sowie im Kapitel *Fazit und Ausblick* erwähnt wird, einen weiteren Vorteil bringen: Aufgrund ihrer beständigen Struktur können mehrere Scans desselben Straßenabschnitts zueinander ausgerichtet werden. Damit könnte etwa die Veränderung des Straßenzustands in einem bestimmten Zeitraum dokumentiert werden.
Eine weitere Auffälligkeit sind besonders dunkle, unregelmäßige und teils sehr große Flecken. Ohne genauen Anblick vor Ort ist nicht zweifelsfrei auszumachen, worum es sich handelt: Bindemittelaustritte sind eine Alternative, aber auch Ölflecken sind möglich und, durch die Nähe zu Parkplätzen, hier auch plausibel. Frische Ölflecken allerdings stellen augrund der geringen Griffigkeit eine große Rutschgefahr dar. Wegen dieser Option werden auch jene Stellen gesondert behandelt; der Einfachheit halber ist in der Arbeit folgend die Rede von Ölflecken.
Von der entgegengesetzten Intensität, aber für den sicheren Straßenverkehr unerlässlich, sind Fahrbahnmarkierungen. Durch ihre Materialzusammensetzung sollen sie sich zu allen Tageszeiten und Wetterbedingungen stark abheben vom Rest der Fahrbahn und so Orientierung bieten. Von Interesse könnte hier insbesondere sein, ob und welche Markierungen sich stellenweise durch Verschleiß nicht mehr ihrem Zweck gemäß genug abheben und einer Auffrischung oder Erneuerung bedürfen.

Andere Arbeiten wie [17] und [3] haben sich vollständig auf Höhenunterschiede vielfältiger Art konzentriert. Dazu zählen nicht nur Schlaglöcher, sondern auch feine Risse, Spurrillen oder Unebenheiten der Straße auf Quer- und Längsseite um wenige Prozent, die unter Umständen über mehrere Meter gemessen werden. Dies waren keine Ziele der Arbeit, entweder aus Mangel an geeigneten und ausreichenden Daten oder, da andere Verfahren besser geeignet erscheinen. Für die Berechnung dieser Quer- wie Längsunebenheit beispielsweise existieren Empfehlungen der * . Diese beschreiben imaginäre Holzlatten, welche in regelmäßigen Abständen quer zur und längs der ebenfalls gescannten Straße gelegt werden. Anhand der Höhendifferenzen der Lattenenden können die gesuchten Werte mathematisch präzise ermittelt werden, ebenso andere Kenngrößen wie die Spurrinnentiefe oder fiktive Wassertiefe. [3] verfolgen ähnliche Ansätze über Funktionsgraphen von Querschnitten der Oberfläche. Solche geometrischen Verfahren sind bereits verhältnismäßig akkurat und zuverlässig.

er vllt. was
on der Stadt
ssen

Die Schäden dieser Art spielen auch eine Rolle bei der Straßenzustandsbewertung. Daher könnten jene Verfahren kombiniert werden mit Ansätzen, die für die Erkennung von unregelmäßigen Schäden der Fahrbahnoberfläche spezialisiert sind, wie es die in dieser Arbeit getesteten sein sollen. Auf diese Weise kann erst ein gesamtheitliches Bild des Straßenzustands geschaffen werden.

3.2 Preprocessing

Die Schadenserkennung wird nicht unmittelbar auf der ursprünglich eingehenden Punktwolke (nachfolgend *Input*) ausgeführt, sondern zuerst auf geeignete Weise vorprozessiert. Die erste Aufgabe besteht darin, den Boden des Inputs zu bestimmen, um die Menge der potenziell interessanten Punkte für diesen Anwendungsfall zu reduzieren. Dazu wird eine Bodenerkennung (*Ground Detection*) eingesetzt, welche unter anderem mit Höhenmodellen arbeitet. Schließlich werden die als Boden erkannten Punkte mit der entsprechenden semantischen Klasse markiert. In der Pipeline dieser Arbeit wird eine Ground Detection eingesetzt, die für den jeweiligen Input optimierte Parameter nutzt. Diese werden im Voraus - als anderes Teilsystem des Projekts - ermittelt anhand der Oberflächeneigenschaften des Inputs. Dadurch soll möglichst der komplette Boden, also insbesondere die vollständige Straße, und möglichst wenig an restlichen Punkten erfasst werden.

Mit diesen Informationen geht es dann an den Schritt, der die Straße selbst erkennen soll. Diese Street Detection soll also außenstehenden Rasen, niedrige Häuserwände und sonstige als Boden klassifizierte, aber nicht zur Fahrbahn selbst gehörenden Bereiche (*Noise*) entfernen. Dabei wird vor allem mit der relativen Dichte der Punkte und einer darauf basierenden Abschätzung der Trajektorie des Scannerfahrzeugs gearbeitet. Dies beruht auf der Annahme, dass durch die Arbeitsweise der Scanner nahegelegene Oberflächen deutlich dichter abgetastet werden als weiter entfernte. Die Street Extraction baut auf einer zumindest größtenteils gelungenen Ground Detection auf. Sobald die nun als Straße erkannten Punkte extrahiert sind, beginnt der jeweilige Ansatz auf seine Weise, die oben genannten Klassen in der verbliebenen Punktwolke zu erkennen.

Zunächst erfolgt jedoch noch eine Wertenormalisierung. Während der Deep-Learning-Ansatz seine eigene Art der Normalisierung vornimmt, geht der Feature-Ansatz folgendermaßen vor: Das Intensitätsattribut besitzt eine hohe Varianz bezüglich verschiedener Laserscanner, die unter anderem von deren unterschiedlicher Technik oder Verarbeitung verursacht wird. Zwar liegt der Wertebereich des Attributs grundsätzlich bei 0 bis 65535, wird aber oftmals nicht ausgenutzt. Der Intensitätswert desselben Punktes, aufgenommen von zwei verschiedenen Scannern, kann sich um mehrere Tausend unterscheiden. Aus diesem Grund werden diese absoluten Werte zunächst in relative umgewandelt, sodass sie über verschiedene Punktwolken hinweg vergleichbar sind.
Ein weiterer Aspekt, der sich negativ auf die Normalisierung auswirken kann, sind einzelne Punkte, deren Intensität übermäßig stark von der Mehrheit der anderen Werte abweicht. Unabhängig davon, ob dies Messungenauigkeiten sind oder tatsächlich einzelne Abweichler, ist ein Entfernen dieser Werte erstrebenswert. Da aber natürlich auch diese Punkte erhalten bleiben und eine Intensität besitzen sollen, werden diese Außenseiter-Punkte noch vor der Normalisierung bestimmt und ihre Intensität auf den niedrigsten bzw. höchsten Wert gesetzt, der nicht zu den abweichenden Werten gezählt wird. Auf diese Weise sollen die Relationen zwischen verschiedenen Punktwolken erhalten bleiben, ohne

durch die Werteanpassung und -normalisierung einen merklichen Informationsverlust zu erleiden bzgl. der hier betrachteten Klassen.
Da bei den 3D-Koordinaten keine so erheblichen Unterschiede zu erwarten sind, werden diese Werte nicht normalisiert und alle letztlich daraus ermittelten Features als vergleichbar über verschiedene Punktwolken hinweg angesehen.

3.3 Ansatz Feature-Extraction

Der Ansatz über die Feature-Extraction in seiner momentanen Form gliedert sich grob in zwei Teilschritte: Zunächst erfolgt die Ermittlung der Pro-Punkt-Features selbst, was den deutlich zeitintensiveren Anteil ausmacht. Anschließend erfolgt die Vorhersage (*Prediction*) der Klassen allein auf Basis dieser berechneten Featurevektoren über einen Random Forest, ggf. reduziert auf eine Menge von in diesen Werten besonders auffälligen Punkten (siehe Abschnitt *Uniqueness*). Dieses Vorgehen entspricht somit dem klassischen Machine-Learning-Prozess.

3.3.1 Features

Bei einer manuellen Feature-Bestimmung stellt sich zuerst die Frage, welche Features überhaupt in Betracht gezogen werden. Daher sollen nun grundsätzliche Features erläutert werden, die gewisse Unterscheidungsmerkmale zwischen den verschiedenen Klassen ausdrücken.

aper?

Die reine Intensität, also ein Maß zur Ermittlung der Stärke des reflektierten Lasers , hebt bereits Fahrbahnmarkierungen durch einen sehr hohen und Ölflecken durch einen sehr niedrigen Wert hervor. [17] nutzen dieses Attribut ebenfalls als Teil ihrer Featurevektoren, wobei dort noch eine Interpolation über Gewichtungen invers zur Distanz vorgenommen wird. Da der eine Wert scale-unabhängig ist und es auch einzelne Punkte mit diesen extremen Ausschlägen gibt (siehe Abbildung Y), die weder zur einen noch zur anderen Klasse gehören, wird an dieser Stelle eine Abhängigkeit vom Scale geschaffen: Beispielsweise kann der Durchschnitt der Nachbarschaftsintensitäten berechnet werden. Im kleinen Scale mögen diese Werte sich auch für Ausreißerstellen ähneln, im größeren Scale aber gleicht sich das für diese aus und bleibt etwa für Fahrbahnmarkierungen charakteristisch hoch.
Ebenfalls auf dem Intensitätsattribut beruhend ist eine Betrachtung der lokalen Intensitätsunterschiede der Nachbarn zum jeweils betrachteten Punkt. Neben Fahrbahnmarkierungen und Ölflecken sind gerade Flickstellen dadurch gekennzeichnet, dass die deutlich dunklere Fugendichtmasse, die Streifen mit wenigen Zentimetern Breite bildet, sich abhebt vom Rest der umliegenden Oberfläche und somit die meist rechteckige Form erkennen lässt. Auch Schlaglöcher besitzen oft, bedingt durch den bei ihnen abgetragenen Asphalt, eine geringere Intensität. Diese Differenzen sollen also dabei helfen, in ihrer Intensität auffällige Stellen einzugrenzen und zu erkennen.

Neben diesen in unterschiedlichen Formen auf der Intensität basierenden Features werden

natürlich auch die 3D-Koordinaten der Punkte genutzt. Es existieren verschiedene Maße, die für einen einzelnen Punkt die Form und Oberfläche seiner lokalen Nachbarschaft beschreiben. Eine Auswahl dieser grundsätzlich einwertigen Features soll im Folgenden erläutert werden.
Die Basis jener Maße ist die Betrachtung der umgebenden Koordinaten eines Punktes und deren Verteilung. Dies wird repräsentiert durch die Kovarianzmatrix. Grundsätzlich beschreibt die Kovarianz zwischen zwei Variablen einen Grad ihres linearen Zusammenhangs: Ihr Vorzeichen deutet an, ob höhere Werte der einen Variable eher mit höheren oder niedrigeren Werten der anderen Variable einhergehen und umgekehrt. Im konkreten Fall von 3D-Daten sind die Variablen die drei Dimensionen X, Y und Z selbst, die zusammen eine 3×3-Kovarianzmatrix einer Punktmenge bilden, welche in den entsprechenden Einträgen die jeweilige Kovarianz zwischen den Dimensionen beschreibt. Die Durchschnitte der Variablen für die Zentrierung der Koordinaten sind in diesem Fall die Koordinaten des durchschnittlichen Punkts der Menge, dem sogenannten *Centroid*.

Ist diese Matrix für einen Punkt und seine Nachbarschaft einmal berechnet, müssen nun sinnvolle Informationen daraus extrahiert werden. Für die genutzten Maße geschieht das mittels der drei Eigenwerte der Matrix. Diese bemessen die höchsten Varianzen entlang paarweise orthogonaler Achsen. Der zum geringsten Eigenwert zugehörige Eigenvektor repräsentiert dabei die Normale zur Punktmenge, welche senkrecht auf deren approximierter Oberfläche steht.
Diese Eigenwerte λ_1, λ_2 und λ_3 werden absteigend sortiert, sodass $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Anschließend werden sie normalisiert durch Division ihrer Gesamtsumme:

$$e_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3}, i \in \{1, 2, 3\}. \tag{3.1}$$

Mittels dieser drei Werte können nun die gesuchten Maße *Linearity*, *Planarity*, *Scattering* (auch *Spherity*) sowie *Local Curvature* (auch *Change of Curvature*, folgend nur *Curvature*) einfach berechnet werden, wie es in Tabelle 3.1 dargestellt ist. Diese Features werden unter anderem von [16] genutzt, um städtische Objekte von Mobile-Mapping-Punktwolken zu erkennen. Sie beschreiben, grob formuliert, wie linear, eben oder rund die Nachbarschaft eines Punktes sich verhält und wie stark sie sich im Vergleich zu einer Ebene krümmt. Von ihrer Aussagekraft sind sie vergleichbar mit den in [17] genutzten Features *Roughness Index* und *Gaussian Curvature*. Im Gegensatz dazu werden in diesem Ansatz aber keine globalen Features verwendet, die jeweils die gesamte Punktwolke verarbeiten. Während etwa [17] *Triangulated Irregular Networks* und Verfahren der *Object Segmentation* dazu nutzen, um die Umrisse von Objekten wie Schlaglöchern und Rissen zu detektieren, sollen hier tatsächlich nur lokale, also jeweils auf einen Scale beschränkte, Features genutzt werden.

3.3.2 Scales

Um einen Punkt zu charakterisieren, wird seine lokale Nachbarschaft betrachtet. Für die Ermittlung der Nachbarschaft gibt es grundsätzlich zwei Wege: die k nächsten Nach-

Linearity	Planarity	Scattering	Curvature
$\frac{e_1-e_2}{e_1}$	$\frac{e_2-e_3}{e_1}$	$\frac{e_3}{e_1}$	$\frac{e_3}{e_1+e_2+e_3}$

Tabelle 3.1: *Die vier einwertigen, auf den Eigenwerten der Kovarianzmatrix basierenden Features zur Beschreibung der Form der lokalen Nachbarschaft.*

barn zu nehmen oder alle Nachbarn, die im vorgegebenen Radius einer Kugel um den Ursprungspunkt liegen. Der Vorteil an den k Nachbarn ist, dass man sich über die betrachtete Nachbarzahl sicher sein kann. Allerdings hängt der dabei betrachtete Radius erheblich von der lokalen Punktdichte ab: Beim Mobile-Mapping wird für Punkte auf der Trajektorie des Scannerfahrzeugs mit demselben k also ein deutlich geringerer Radius betrachtet als für Punkte näher am Rand, siehe Abbildung X. Deshalb eignet sich die Radiusbetrachtung besser für konsistente und über verschiedene Punkte vergleichbare geometrische Features [15]. Diese Art der Nachbarschaftsbetrachtung - von nun an werden die Radii *Scales* genannt - findet in dem Ansatz entsprechend Verwendung.

Die nächste Frage ist, welche konkreten Scales genutzt werden. Dass überhaupt die Nutzung mehrerer Scales sinnvoll ist, zeigt sich beim Blick auf verschiedene Objekte, die es zu erkennen gilt. Zum einen sind natürlich gerade Schlaglöcher besonders unregelmäßige Strukturen, die in vielerlei Ausprägungen vorkommen können: über mehrere Größen hinweg sowie Formen von sehr rund bis eher länglich. Flickstellen zeichnen sich vorzugsweise in kleinem Scale aus durch ihre hohen Intensitätsunterschiede. Bei größeren Gullys wird der innere, mittige Bereich aber erst mit deutlich größerem Radius auffällig, denn im kleinen Bereich ist dieser sehr planar. Um mit diesen verschiedenen Granularitäten umgehen zu können, werden mehrere, an typische Objektgrößen angepasste, Scales genutzt.

3.3.3 Ein- und mehrwertige Features

Als Features für Punkte kommen zum einen einwertige Daten in Betracht. Das können direkt Attribute sein wie die Intensität, Werte, die sich aus der Punktenachbarschaft ergeben wie Curvature, oder kumulierte Formen davon wie Durchschnitte oder Standardabweichungen.
Daneben gibt es auch die Möglichkeit für mehrwertige Features. Diese können etwa als Histogramme ausgedrückt werden, also Häufigkeitsverteilungen für bestimmte numerische Merkmale. Sie sind zusammengesetzt aus mehreren sogenannten Bins, welche die Anzahl der Observationen zählen, die in den von ihnen aufgespannten Wertebereich fallen. In dieser Arbeit entspricht eine Observation dem Merkmal eines Punktes, etwa einem Curvature-Wert. Histogramme können linear sein, wenn alle Bins einen identisch großen Wertebereich haben, oder entsprechend nicht-linear, wobei in diesem Ansatz nur von linearen Histogrammen Gebrauch gemacht wurde, wie im Kapitel *Implementierung* nachzuvollziehen ist.

Der Vorteil der Histogramme besteht in einer grundsätzlich höheren Aussagekraft durch die genauere Charakterisierung der Nachbarschaft [13]: Was sich etwa in einem Durchschnittswert durch entgegengesetzte Größen aufheben würde, kann nun präziser dargestellt werden. Mehr und damit feinere Bins, d.h. jeweils kleinere Wertebereiche, haben eine potenziell höhere Aussagekraft. Allerdings resultiert dies unter Umständen in einer zu starken Verteilung der Häufigkeiten, sodass charakteristische Ausschläge verloren gehen. Histogramme bieten aber noch weitere Vorteile: Die oben angesprochenen teilweise erheblichen Unterschiede der Dichte an verschiedenen Stellen einer Mobile-Mapping-Punktwolke machen einwertige Features unter Umständen schwierig vergleichbar für einen festen Scale. Bei Histogrammen kann dieser Fakt abgeschwächt werden, da durch das Einsortieren in Bins die Werte normalisiert werden in den Bereich von 0 bis 1, und zwar unabhängig von der Anzahl der Nachbarn. Deswegen werden im Ansatz zum Vergleich zwischen Histogrammen und auch für die Prediction lediglich die relativen Bingrößen (*Frequencies*) betrachtet. Ferner wird auch der Umgang mit Außenseiterpunkten (*Outliern*), die teilweise von Messungenauigkeiten stammen, erleichtert: Während sie im einwertigen Fall ein Feature stark verzerren können, erhöhen sie beim Histogramm nur die Frequency eines Bins leicht und sind somit beinahe wie leere Bins zu behandeln. Ähnliches gilt für Werte mit hoher Varianz, wie es vor allem die gemessene Intensität ist. Schließlich sind die Histogramme, im Gegensatz zu einigen anderen lokalen Deskriptoren [4], unabhängig von Transformationen der Punktwolke wie Rotation oder Translation.
Für diesen Ansatz werden Histogramme genutzt bei den Features zur geometrischen Oberflächenbeschreibung sowie den Intensitätsunterschieden. In den Kapiteln *Implementierung* und *Evaluierung* wird darauf eingegangen, mit welchen Wertebereichen und Binzahlen gearbeitet wird bzw. was dies für Auswirkungen hat auf die Unterscheidbarkeit zwischen den Klassen. Grundsätzlich wird aber eine Kombination aus ein- und mehrwertigen Features eingesetzt. Zusammengehörige einwertige Features sowie einzelne Histogramme eines Attributs werden künftig teilweise als *Featureräume* bezeichnet, etwa als Curvature-Featureraum.

3.3.4 Uniqueness

Die zu klassifizierenden Objekte und deren einzelne Punkte sollten sich - im ein oder anderen Featureraum - merklich unterscheiden von den gewöhnlichen Straßenpunkten, die weder einen Schaden noch ein sonstiges besonderes Objekt darstellen. Dieses Konzept der *uniquen* Punkte, teilweise auch *key feature points*, hat das Ziel eine Punktwolke durch eine geeignete und angemessen große Menge von Punkten zu charakterisieren.
Die Basis für die Ermittlung dieser uniquen Punkte sind, wie vom Prinzip her auch in [13] genutzt, die Berechnung einer Featurerepräsentation des *durchschnittlichen* Punktes (*Mean*) sowie der Distanz der Repräsentation einzelner Punkte zu diesem Durchschnitt. Diejenigen Punkte, die eine spezifizierte ausreichend große Distanz besitzen, werden als unique betrachtet.
Die durchschnittliche Repräsentation kann für einwertige Featureräume durch einfaches Bilden des Durchschnitts ermittelt werden, für mehrwertige Featureräume (Histogramme)

werden die jeweiligen Bins über alle Punkte aufsummiert und entsprechend die Frequencies neu berechnet. Die Distanz von der Punktrepräsentation zur Meanrepräsentation kann dann über ein geeignetes Distanzmaß berechnet werden. Bei Betrachtung aller Distanzen können anschließend die entferntesten Punkte bestimmt werden.

Die Gründe für eine solche Darstellung der Punktwolke durch eine geringere Zahl an Punkten sind vielfältig. Eine Motivation kann - gerade für Objekte, für die eher nicht erwartet wird, sich zu verändern oder zu bewegen - die Registration sein. Dabei sind die Ausgangslage zwei oder mehr Punktwolken, die durch mehrere Scans entstanden sind, etwa zu verschiedenen Zeitpunkten oder aus unterschiedlichen Perspektiven. Durch geeignete Algorithmen wie dem *Sample Consensus Initial Alignment* [12] können für unique Punkte des einen Scans sogenannte Korrespondenz-Punkte im anderen Scan gesucht werden, welche die jeweils ähnlichste Featurerepräsentation besitzen. Aus diesen Paaren kann die nötige Transformation zwischen den Punktwolken bestimmt und durchgeführt werden. Durch ein entsprechendes Fehlermaß, etwa über die paarweisen Distanzen, kann die Qualität dieser Annäherung ermittelt und nach dem geringsten Fehler optimiert werden, bevor eine abschließende Verfeinerung dieses *Alignments* zum Beispiel über nicht-lineare Verfahren erfolgt. Der Vorteil der Nutzung von uniquen Punkten liegt hierbei darin, dass die Ausgangsmenge der zu betrachtenden Punkte für die Korrespondenzsuche drastisch, aber zugleich sinnvoll reduziert wird. Die tiefgreifendere Verwendung des Uniqueness-Konzepts wird in dieser Arbeit nicht näher behandelt, doch im Kapitel *Fazit und Ausblick* wird noch einmal angerissen, inwiefern damit potenziell Änderungen des Straßenzustands im Verlaufe der Zeit ermittelt werden könnten.
Ein zweiter und pragmatischer Grund für die Ermittlung von uniquen Punkten ist die resultierende Datenreduktion. Wie im Kapitel *Implementierung* genauer erläutert wird, ist in der momentanen Architektur dieses Ansatzes eine Übertragung der Featurevektoren mit temporärer Festplattenspeicherung nötig. Da weiterhin die Nutzung von mehrwertigen Features wie Histogrammen inhärent zusätzlichen Speicheraufwand bedeutet, soll das Uniqueness-Konzept Abhilfe schaffen. Eine Bedingung für ordentliche Ergebnisse der Uniqueness ist in diesem Anwendungsfall, dass die Objekte von Interesse eine kleine Minderheit der Punkte repräsentieren und so in der Gesamtpunktwolke entsprechend auffallen. Dies ist bei gewöhnlichen Straßenpunktwolken der Fall, siehe Kapitel *Evaluierung* für Relationen der Klassen. Für die nicht-uniquen Punkte wird die implizite Annahme getroffen, dass diese den gewöhnlichen Straßenzustand repräsentieren und ihnen somit direkt die Klasse *Straße* zugewiesen wird.

3.3.5 Approximation für größere Scales

Die Zahl der Nachbarn eines Punktes erhöht sich mit linear steigendem Scale deutlich stärker als linear. Entsprechend verlängert sich auch die Prozessierungsdauer pro Scale erheblich. Da unter Umständen solche größeren Scales aber nicht einfach ausgelassen werden können, beinhaltet der Feature-Ansatz die Möglichkeit einer Approximation der Featurevektoren aus reduzierten Originaldaten.

Dazu wird, noch vor der Extraktion der Features, eine Dichtereduktion (*Density Reduction*) ausgeführt. Diese Operation kann die Punktemenge senken, indem alle Punkte etwa innerhalb eines Würfels zu einem einzigen Punkt zusammengefasst werden. Für Feinheiten sind kleinere Scales gedacht, die jeden Punkt berücksichtigen. Groberen Betrachtungen genügt auch eine regelmäßig gesampelte Teilmenge aller Punkte.
Die Density Reduction verringert die Nachbarzahl jedes Punktes und somit insbesondere die Dauer der Featuregewinnung. Allerdings sind anschließend nur Features berechnet worden für eine Teilmenge der ursprünglichen Punkte. Da aber für jeden Punkt ein vollständiger Featurevektor erwartet wird für die letztliche Prediction, muss dieser jeweils aus den bisher ermittelten approximiert werden. Dazu werden für jeden Originalpunkt und pro Scale alle reduzierten Punkte in kleiner Nachbarschaft gesammelt, ihre Featurevektoren gewichtet nach der inversen Distanz zum Originalpunkt und so zu einem einzelnen Featurevektor verrechnet. Dieser Approximationsschritt soll also die Prozessierungszeit senken und die einzelnen Featurevektoren dabei trotzdem hinreichend genau darstellen.

3.3.6 Prediction per Random Forest

Anhand der zuvor ermittelten Featurevektoren, bestehend aus mehreren Scales und ggf. reduziert auf die Menge der uniquen Punkte, soll nun jeweils die Klasse jedes Punktes bestimmt werden. Wie bei [17] wird im Feature-Ansatz dafür ein Random Forest verwendet, der in dortigen Experimenten einen probabilistischen Ansatz sowie ein Support-Vector-Machine-Modell übertroffen hat. Ein Random Forest zeichnet sich grundsätzlich durch eine hohe Robustheit gegenüber kleinen Abweichungen oder Anomalien aus, die besonders bei feinen Features, wie sie hier vorkommen, wünschenswert ist.
Die Grundlage eines Random Forest ist der Decision Tree. Ein solcher einzelner Baum ist steuerbar über eine Menge von Parametern. Im Allgemeinen erhält er nur eine Teilmenge der Trainingsdaten und auch nur eine Teilmenge der Features zur Betrachtung. Anschließend wird er daraufhin trainiert, durch eine Reihe von Entscheidungen - im rein numerischen Fall basierend auf geeigneten Schwellwerten - die Klasse des Inputs anhand der ihm zur Verfügung gestellten Features zu bestimmen. Um *Overfitting*, also eine zu starke Spezialisierung auf die Trainingsdaten und damit zu geringe Generalisierung, zu vermindern, kann auch die maximale Anzahl an Entscheidungen (die Tiefe des Baums) festgelegt werden.
Für sich genommen ist ein einzelner Baum nur bei einigen Features zu einer sinnvollen Entscheidung für eine Klasse geeignet, bei anderen Features ist diese eher zufällig. Dies wird im Random Forest dadurch ausgenutzt, dass jener aus teils Hundert oder mehr unkorrelierten Decision Trees besteht. Jeder von diesen gibt seine Entscheidung preis, und die endgültige Entscheidung des gesamten Modells ergibt sich als Mehrheitsmeinung (*Majority Vote*) dieser Einzelvorhersagen. Dieses Zusammenspiel aus “Expertenmeinungen“ jeweils spezialisierter Bäume und eher zufälligen Entscheidungen der restlichen Bäume ist eine der wesentlichen Gründe für die Stabilität dieser Modellart.
Genaueres zum genutzten System und den verwendeten Random-Forest-Parametern finden sich im entsprechenden Abschnitt des Kapitels *Implementierung*.

3.4 Ansatz Deep Learning

PointNet [1] ist eine 2016 erstmals vorgestellte Deep-Learning-Architektur. Das neuronale Netz ist in der Lage eine Punktmenge zu verarbeiten, ohne von der Reihenfolge der Punkte abhängig zu sein. Dabei kann es sowohl einer gesamten Punktwolke eine Klasse zuweisen als auch die Punktwolke nach semantischen Klassen pro Punkt segmentieren. Die Fähigkeit mit der inhärenten Unstrukturiertheit von Punktwolken direkt umzugehen, machte PointNet so neuartig. Verwandte Arbeiten zuvor basierten, wie schon gleichnamigen Kapitel aufgeführt, häufig auf einer Einteilung der Punktwolke in ein regelmäßiges 2D- oder 3D-Gatter oder eine Menge von Bildern, um etablierte bildbasierte Verfahren oder *Convolutional Neural Networks* einzusetzen. Da der Fokus dieser Arbeit nicht eine Implementierung von PointNet war, soll dessen Architektur nur grob zusammengefasst werden. Als wichtige Merkmale des Netzes werden genannt:

- das Nutzen einer symmetrischen Funktion für die permutationsunabhängige Prozessierung der Punkte
- die gleichzeitige interne Verwendung und Konkatenation von globalen Features und lokalen Features für die Berechnung der Gesamtklasse der Punktwolke oder der Einzelklassen der Punkte
- die automatische Angleichung der Koordinaten sowie Pro-Punkt-Features über Punktwolken hinweg, um eine Unabhängigkeit von Transformationen wie Rotation und Translation zu erreichen

Standardmäßig besteht der Input für PointNet lediglich aus den dreidimensionalen Punktkoordinaten. Allerdings lassen sich beliebige weitere Pro-Punkt-Features anhängen, was insbesondere für solche Attribute sinnvoll ist, die sich nicht indirekt aus den Koordinaten allein herleiten lassen. In dieser Arbeit gilt das für die Intensität, die auch im Feature-Extraction-Ansatz genutzt wird. Entsprechend werden für einen fairen Vergleich diese Werte den Koordinaten angehängt.
Auch die Arbeitsweise der beiden Ansätze bei der Prediction unterscheidet sich: Statt sich Punkte einzeln anzuschauen, Features zu berechnen und eine Klasse vorherzusagen, wird bei der Nutzung von PointNet eine (durch Parameter in der Größe beeinflussbare) Menge von Samplepunkten aus der gesamten Punktwolke gezogen. Für jeden dieser Punkte wird eine feste Zahl an Nachbarn gezogen in ebenfalls fest eingestelltem (maximalen) Scale. Bei zu wenig Nachbarn werden Punkte mehrmals in die Menge aufgenommen. Nach Durchlaufen des Netzes wird jedem dieser Punkte auf einmal eine eigene Klasse zugewiesen, die das Modell für am wahrscheinlichsten hält.
Bei der Prozessierung ermittelt PointNet weiterhin von sich aus eine Menge sogenannter *Critical Points*, welche die lokale Nachbarschaft hauptsächlich charakterisieren. Dieses Resultat ähnelt damit der Idee des im Feature-Ansatz genutzten Uniqueness-Konzepts.
Insgesamt lässt sich sagen, dass sich dieser modernere Ansatz - wie andere Deep-Learning-Architekturen - dadurch auszeichnet, automatisch aussagekräftige Features aus dem Input zu extrahieren. Dadurch wird ein grundsätzlich generischer Ablauf geschaffen, der

nicht mehr auf manuell erarbeiteten und meist auf einen Anwendungsfall spezialisierten Features basiert, sondern lediglich ausreichend Trainingsdaten benötigt.

3.5 Postprocessing

Die Nachprozessierung findet statt, nachdem vom jeweiligen Ansatz alle Punkte einer Klasse zugewiesen wurden. Die Grundidee ist, Klassen von denjenigen Punkten noch einmal zu ändern, bei denen eine andere Klasse aufgrund der unmittelbaren Nachbarklassen sehr wahrscheinlich ist. Dies geschieht in zwei Richtungen:
Zum einen werden Klassen-Außenseiter als Standardklasse “Straße“ eingestuft. Das sind Punkte, die momentan nicht “Straße“ sind, aber nicht genügend Punkte ihrer Klasse in unmittelbarer Umgebung haben. Für jede der betrachteten Objektklassen wird angenommen, aus mehr als einigen wenigen Punkten zu bestehen. Solche Klassifizierungen werden daher als Noise interpretiert und nicht weiter als bemerkenswert behandelt.
Die andere Richtung behandelt Straßen-Punkte, die inmitten von Nicht-Straßen-Punkten liegen. Bei genügend großem Anteil von letzterer Gruppe nimmt der Punkt die häufigste Nicht-Straßen-Klasse seiner Nachbarschaft an. Auf diese Weise werden zum Beispiel Schlaglöcher innen aufgefüllt oder die Mitte von Gullys ergänzt, wenn sie etwa wegen eines zu geringen Scales nicht als eben diese Klassen erkannt wurden.
Die beiden Schritte werden mit verhältnismäßig vorsichtigen Werten durchgeführt, um die begründeten initialen Klassifizierungen nicht zu stark zu verändern auf Basis einer simplen Heuristik.

Kapitel 4

Implementierung

4.1 Systemüberblick

Genutzte Features bzw. Binanzahl und -grenzen werden nach letzten Experimenten eingefügt! ('XXX')

Noch vor neuer Einleitung geschrieben; wird hier evtl. verkürzt

Beide Ansätze bauen auf ein System von bestehenden und abgeschlossenen Tools auf. Diese wurden teilweise ergänzt um nötige Erweiterungen für den Feature-Extraction-Ansatz.

Das in dieser Arbeit bedeutsamste Werkzeug für die Prozessierung von Punktwolken ist das *PCTool*, einer in C++ geschriebenen Qt-Anwendung. Dieses Framework bietet eine Vielzahl an Möglichkeiten zur effizienten Verarbeitung von Punktwolken. Die Grundbausteine sind sogenannte Nodes, die jeweils eine Operation zu Eingabe, Ausgabe oder Prozessierung implementieren. Dazu gehören zum Beispiel Reader und Writer von Punktwolken verschiedener Dateiformate. Auch die bereits erwähnten Funktionalitäten der Density Reduction und Ground Detection finden sich dort als separate Nodes wieder neben den für den eigenen Ansatz implementierten. Um die Nodes zu nützlichen Aktionen zu verknüpfen, können sie zu Pipelines zusammengelegt werden. Ein Beispiel dazu findet sich in Abbildung X, wo der beschriebene Ablauf des ersten Teils vom Feature-Ansatz zu sehen ist: vom Einlesen der Punktwolke, der Ground Detection und Straßenextraktion bis zum Ermitteln der Featurevektoren der einzelnen Punkte zur Schadenserkennung und Zurückschreiben der Punktwolke. (Anmerkung: Die einzelne Ground-Detection-Node steht an dieser Stelle nur stellvertretend für die im Projekt genutzte elaboriertere Bodenerkennung, siehe [8].) Auf die Implementierungen von Teilen jener Nodes wird in diesem Kapitel näher eingegangen.

Der zweite Schritt des Feature-Extraction-Ansatzes, den Predictions der einzelnen Punktklassen anhand der Featurevektoren, findet sich in einer Reihe von Python-Skripten. Diese sind Teil des sogenannten *Modelzoo*, der eine Menge von Machine-Learning-basierten Funktionalitäten für die Webplattform [14] bietet. Unter anderem ist dort auch die Klassifizierung von Punktwolken anhand ihres sogenannten *PCFP*s gelegen, einer Art globaler Fingerabdruck [6]. Für den hier besprochenen Ansatz werden vor allem Funktionalitäten zum Trainieren und Testen von Random Forests für die Predictions bereitgestellt.

Das ebenfalls in Python geschriebene Framework *PCNN* implementiert Architekturen

von neuronalen Netzen, die direkt auf Punktwolken operieren, wie etwa das vorgestellte PointNet. Außerdem bietet es Schnittstellen zum Vorverarbeiten von Punktwolken sowie zum Trainieren von bzw. Predicten mit solchen Modellen. Vorausgesetzt werden lediglich entsprechend annotierte Punktwolken. Auch diverse Parameter der Netze zum Steuern des Trainings- und Predictionprozesses lassen sich hier einstellen. Dazu können Pipelines mit jeweiligen Schlüssel-Werte-Paaren definiert und anschließend aufgerufen werden. Mithilfe dieses Tools wurde der Deep-Learning-Ansatz über PointNet getestet.

Um sich die Punktwolken selbst und deren annotierte Klassen anzusehen, wurde der *PCViewer* genutzt. Dieser visualisiert dabei die einzelnen Punkte und bietet eine Vielzahl von weiteren Darstellungsoptionen. Beispielhaft können die Punkte nach ihrer Intensität, ihrem Farbwert oder ihrer Klassenzugehörigkeit eingefärbt werden. Eine solche Visualisierung ist unter anderem für eine qualitative Analyse von Predictions der einzelnen Ansätze nötig, wie sie im Kapitel *Evaluierung* vorgenommen wird. Alle in dieser Arbeit anzutreffenden Bilder von Punktwolken wurden ebenfalls mit dem *PCViewer* aufgenommen.

Das letzte im Zuge dieser Arbeit genutzte Programm ist das *TrainingTool*. Dieses kann, ähnlich wie der Viewer, eine Punktwolke mit all ihren Punkten darstellen. Der Zweck ist hierbei vordergründig nicht die Visualisierung, sondern die Möglichkeit zur Klassenänderung von einzelnen Punkten, das sogenannte Annotieren oder Labeln. Als Hilfestellung können dennoch andere Anzeigeformen dienen, die etwa die Punkthöhe berücksichtigen. Um sich eine Grundlage für die Experimente dieser Arbeit zu schaffen und mangels geeigneter vorhandener Datensätze, wurden mit dem Tool die Schlaglöcher, Gullys und alle weiteren erwähnten Klassen manuell nach bestem Wissen und Gewissen markiert. Insbesondere wegen der teils feinen Objekte sowie der begrenzten Auflösung der Punktwolken ist damit aber immer eine Unsicherheit behaftet. Gerade bzgl. der Grenzen von Flickstellen und Ränder von Schlaglöchern ist diese Einschränkung in der qualitativen und quantitativen Analyse der Predictions zu berücksichtigen.

4.2 Preprocessing

Auf die Hintergründe und Implementierung der ausgereifteren Ground Detection wird hier nicht näher eingegangen, diese sind in [8] zu finden.

4.2.1 Straßenextraktion

Die Straßenextraktion ist in der separaten *StreetExtractor*-Node im PCTool umgesetzt. Dort wird jeweils eine Punktwolke eingelesen, von der erwartet wird, nur noch die zuvor als Boden klassifizierten Punkte zu enthalten. Neben der reinen Straße sind nun auch noch Rasen, Einfahrten und sonstige niedrig gelegene Oberflächen vorhanden. Diese unterscheiden sich dahingehend von der vom Scannerfahrzeug gefahrenen Strecke (der *Trajektorie*), dass sie deutlich weniger dicht abgetastet wurden und entsprechend durch

weniger Punkte repräsentiert werden.

Abbildung X verdeutlicht dies: Die Punkte sind eingefärbt nach ihrer relativen Dichte. Dazu wurden in einem Scale von y cm jeweils die Anzahl an Nachbarn gezählt. Diese Liste wurde anschließend min-max-normalisiert, d.h. von jedem einzelnen Wert wird das Minimum der Liste subtrahiert und geteilt durch die Differenz von Maximum und Minimum. Auf diese Weise ist in der verarbeiteten Liste immer der kleinste Wert die 0 und der größte Wert die 1. Diese relativen Dichten wurden schließlich in Farben des HSV-Farbraums umgewandelt. Somit erscheinen die Punkte mit den dichtesten Nachbarschaften in orangenen und roten Farbtönen, während es nach außen hin blauer und grüner wird. Wie bereits hier zu erkennen ist, lässt sich die Trajektorie anhand dieses Maßes abschätzen. Diese verläuft für die hier dargestellte Punktwolke auf einer Straßenseite sowie dem Gehweg. Demzufolge ist das Fahrzeug wohl genau auf dieser Seite entlanggefahren. Das Ziel ist es allerdings, die vollständige Straße zu extrahieren. Die Ermittlung der anderen Straßenhälfte spielt daher im Folgenden auch eine Rolle, um letztlich die gesamte Straße und alle Schäden auf dieser mit nur einer Scannerfahrt erfassen zu können.

Mit Hilfe des arithmetischen Mittels und der Standardabweichung der relativen Dichten wird die Punktwolke in zwei Teile getrennt: einen, der die grundsätzliche Trajektorie darstellt, und einen den Rest beinhaltenden. Dabei zählen all jene Punkte, deren relative Dichte größer als das Mittel minus einer halben Standardabweichung ist, zum Trajektorieteil. Der Faktor wurde empirisch und nach visuellem Empfinden bestimmt.
Auf den beiden Teilen wird nun parallel der bereits im PCTool integrierte *PointCloud-Clusterer* eingesetzt: Dieser ermittelt auf Punktdistanzen basierende zusammenhängende Bereiche, die *Cluster*. Spezifiziert werden kann die maximale Distanz eines Punktes zur Noch-Zugehörigkeit zu einem Cluster sowie die minimale Clustergröße gegen das Verwerfen von jenem. Die maximale Distanz wurde hier auf ein Zehntel der durchschnittlichen relativen Dichte gesetzt, was bei den getesteten Punktwolken einen guten Tradeoff zwischen Noiseentfernung und Straßenbeibehaltung bedeutete. Die nötige Clustergröße beträgt 30% der ursprünglichen Größe der Teilpunktwolke. Dies beruht auf der Annahme, dass - in beiden Teilen und trotz allen Noises - die Straße den jeweils größten Anteil der Punkte ausmacht, da sie durchgängig verläuft im Gegensatz zu Rasenflächen oder sonstigen nahegelegenen Bodenstellen. Andererseits wird nicht zwangsläufig nur das größte Cluster behalten, da es Fälle geben kann, in denen die Straße getrennt voneinander ist - etwa durch parkende Autos, die durch die vorherige Bodenerkennung entfernt worden sind. Diese 30% sollen also erneut einen Tradeoff herstellen zwischen einer vollständigen Straßenerfassung und möglichst hohen Noiseentfernung. All jene Cluster, die bis dahin übrig geblieben sind - im Trajektorie-Teil wie im Nicht-Trajektorie-Teil - werden zur Straße gezählt und zum Schluss wieder zu einer einzelnen Punktwolke zusammengeführt (*gemerged*).
Wie durch die Erklärungen deutlich wird, handelt es sich bei der momentanen Straßenextraktion um einfache Heuristiken. Ferner beinhaltet sie Parameter, die bisher nur nach reinem Empfinden an wenigen Testpunktwolken bestimmt wurden. Dieser Teilaspekt

der gesamten Pipeline soll keinen Fokus der Arbeit einnehmen, sondern war zunächst rein funktional angelegt. Für künftige Verbesserungen sind sowohl Erweiterungen des hier genutzten Ansatzes denkbar, etwa eine elaboriertere Abschätzung der Parameter. Es sind allerdings auch komplett andere Herangehensweisen möglich, zum Beispiel die Erkennung von Bordsteinkanten an den Rändern der Straße, die sich durch rapide, aber regelmäßige Höhenunterschiede auszeichnen. Die Straße könnte dann als all jene Punkte definiert werden, die zwischen den beiden Kanten liegen. Diese Methode hätte außerdem zur Konsequenz, nur die Fahrbahnoberfläche zu enthalten und keine Gehwege.

lt. Ablaufdia-
ramm zum bes-
eren Verständ-
s?

4.2.2 Intensitätsnormalisierung

Für die Normalisierung der Intensitätswerte wird ebenfalls die Min-Max-Normalisierung verwendet, wie sie im Abschnitt *Straßenextraktion* erläutert wurde. Dadurch hat letztlich die geringste Intensität der Punktwolke den Wert 0 und die höchste den Wert 1. Wie in *Konzepte* beschrieben, findet aber zuvor noch eine sogenannte *Outlier Detection* statt, welche diejenigen Werte finden soll, die auffällig stark vom Rest der Daten abweichen. Der hier genutzte Prozess basiert auf der *Interquartile Range, IQR* [11]. Dabei werden die Werte an einem Viertel sowie drei Viertel (zur Ganzzahl abgerundet bzw. aufgerundet) der sortierten Liste angeschaut: das erste bzw. dritte Quartil Q_1, Q_3. Mit $IQR = Q_3 - Q_1$ können nun die angesprochenen Schwellwerte nach oben und nach unten definiert werden als

$$T_{lo} = Q_1 - \alpha * IQR \quad \text{und} \quad T_{hi} = Q_3 + \beta * IQR \tag{4.1}$$

Alle Werte kleiner als T_{lo} werden schließlich auf T_{lo} gesetzt; alle Werte größer als T_{hi} werden auf T_{hi} gesetzt. Die einzigen Parameter, die dafür noch festgelegt werden müssen, sind α und β. Sie bestimmen, wie vorsichtig oder aggressiv die Outlier Detection verläuft. Häufig nehmen sie Werte um $1,5$ an, in diesem Fall wurden sie - auch bedingt durch die große Spannbreite des Wertebereichs mit starker Konzentration im mittleren Bereich - empirisch mit 4 bzw. 6 belegt. Konkrete Zahlen für die Auswirkungen auf Trainings- und Testpunktwolke finden sich im Kapitel *Evaluierung*.

4.3 Ansatz Feature-Extraction

Der grundsätzliche Ablauf stellt sich wie folgt dar: Nach dem zuvor erläuterten Preprocessing der Intensität findet im *PavementDistressDetector*-Node des PCTools die Ermittlung der Featurevektoren jedes Punktes statt. Pro genutztem Scale werden dabei - in hochparalleler Verarbeitung - die genannten Features berechnet und teilweise in Histogramme verpackt. Falls Uniqueness betrachtet werden soll, wird pro Punkt und Scale geprüft, ob ersterer unique ist und die Information entsprechend hinterlegt. Sind alle Scales durchlaufen, wird die sogenannte *Featuresdatei* geschrieben. Diese im Sinne der Speichereffizienz binäre Datei enthält die vollständigen Featurevektoren aller Punkte. Im Falle der Uniqueness sind nur die Vektoren der uniquen Punkte geschrieben inklusive ihrer Indizes. Zusätzlich stehen in beiden Fällen zu Beginn die Gesamtzahl der folgenden Featurevektoren, die Anzahl der genutzten Scales sowie die Featurevektorgröße pro Scale.

Die Featuresdatei wird, wie im Abschnitt *Prediction per Random Forest* beschrieben, anschließend von den Python-Skripten eingelesen und für die abschließende Vorhersage der Klassen genutzt.
Der Grund für die Trennung der Featuregewinnung und Prediction in zwei Teilsysteme, was einen verhältnismäßig hohen Zusatzaufwand bedeutet durch Übertragung und Zwischenspeicherung der Featurevektoren, liegt in der Handhabbarkeit. Python ist im Bereich des Machine Learning eine etablierte Wahl und bietet insbesondere einfach zu nutzende Schnittstellen. Langfristig ist daher für eine Effizienzsteigerung die Integration des Prediction-Teilsystems in C++ bzw. im Speziellen das PCTool anzustreben.

Pseudocode des Ablaufs? Anschauungsbild folgt noch

4.3.1 Features

Die genutzten Features werden gewonnen aus den in der Punktwolke gespeicherten 3D-Koordinaten sowie Intensitätswerten.
Für die intensitätsbasierten Features werden pro Punkt und Scale zum einen der Durchschnitt der normalisierten Nachbarintensitäten berechnet. Zum anderen wird ein Histogramm erstellt aus den absoluten Differenzen von der Intensität des Ursprungspunkts zu denen der Nachbarn. Dieses Histogramm umfasst 12 Bins mit einem Wertebereich von 0 bis 0.3. Die Obergrenze schließt bereits - trotz theoretischem Maximum von 1 - den größten Teil der vorkommenden Differenzen ein, darunter diejenigen zwischen Flickstellen und angrenzender gewöhnlicher Fahrbahnoberfläche.
Die positionsabhängigen Features müssen aufwändiger berechnet werden. Die für Linearity, Planarity, Scattering und Curvature nötigen Eigenwerte müssen dazu zunächst ermittelt werden. Zu diesem Zweck wurde der `EigenvalueCalculator` im PCTool implementiert. Dieser baut zu großem Teil auf dem zuvor existierenden `NormalsEstimator` auf, welcher für alle Punkte parallel ihre abgeschätzte Normale berechnet. Diese Normale ergibt sich, wie bereits erwähnt, als der Eigenvektor der lokalen Kovarianzmatrix mit dem kleinsten Eigenwert. Da in diesem Schritt also die Eigenwerte ohnehin berechnet werden, soll im EigenvalueCalculator beides kombiniert werden. Mit einstellbarem Scale werden die Normalen berechnet und jeweils drei Eigenwerte für alle Punkte zurückgegeben. Anschließend werden diese entsprechend den Erläuterungen normalisiert durch Division der Gesamtsumme. Zum Schluss können die eigentlichen Features aus den normalisierten Werten ermittelt werden, wozu Objekte der jeweils zuständigen Klasse genutzt werden (`LinearityCollector` für Linearity, usw.).
Aus den grundsätzlich einwertigen Features werden dann mehrwertige durch Histogrammerstellung gebildet. Hierbei sind die Grenzen zusätzlich vom Scale abhängig, da die Werte im Allgemeinen eine höhere Varianz im kleinen Scale besitzen. So können im Radius von wenigen Zentimetern kleine Unterschiede einen großen Einfluss auf den Wert haben, was auch an der geringen Zahl an Nachbarn liegt (siehe Tabelle X). Für Scales von 10, 20 und mehr Zentimetern gleicht sich dies für die allermeisten Punkte wieder aus und die Werte nähern sich dem Durchschnitt der gesamten Punktwolke an. Für einwertige Features mit eher niedrigen Werten (Linearity, Scattering, Curvature) wird jeweils die Obergrenze der Histogramme angepasst, für Planarity mit eher hohen Werten

Feature	Binzahl	Feste Grenze	$3cm$	$6cm$	$9cm$	$12cm$	$15cm$
Linearity	XXX	0.0	XXX	XXX	XXX	XXX	XXX
Planarity	XXX	1.0	XXX	XXX	XXX	XXX	XXX
Scattering	XXX	0.0	XXX	XXX	XXX	XXX	XXX
Curvature	XXX	0.0	XXX	XXX	XXX	XXX	XXX

Tabelle 4.1: *Die Kennzahlen der linearen Histogramme von den Eigenwert-basierten Features. Ein Eintrag von 0.0 in der Spalte Feste Grenze bedeutet, dass in den folgenden Spalten die Obergrenze angepasst wird. Ein Eintrag von 1.0 bedeutet entsprechend Gegenteiliges.*

entsprechend die Untergrenze. Die genauen Kennzahlen der Histogramme, die vor allem durch manuelle Inspektion für eine gute Unterscheidung zwischen den Klassen festgelegt wurden, finden sich in Tabelle 4.1.

4.3.2 Scales

Um trotz der oftmals Millionen von Punkten einer Punktwolke schnell positionelle Beziehungen herzustellen, wozu etwa das Finden der Nachbarschaft eines Punktes gehört, bedarf es dafür geeigneter Datenstrukturen. Eine solche ist auch der *k-d-Baum*, der einen effizienten Zugriff auf multidimensionale Daten erlaubt, wie es die 3D-Punktkoordinaten sind. Das PCTool bietet dafür die Klasse *SpatialSearch* an, die intern einen k-d-Baum implementiert und eine einfach zu nutzende Schnittstelle bietet. So können mit jeweils einem Funktionsaufruf sowohl die k nächsten Nachbarn einer 3D-Position oder eines Punktindizes gefunden werden als auch alle Nachbarn in einem vorgegebenen Scale. Um einen Eindruck von den durchschnittlichen Nachbarzahlen verschiedener Scales zu erhalten, sind diese in Tabelle X gerundet aufgelistet für die Trainingspunktwolke, jeweils auch separat für den Trajektorie-Teil und den Rest. Wie den Daten zu entnehmen ist und wie bereits angesprochen, steigen die Zahlen immer schneller an für höhere Scales. Die teils erheblich unterschiedlichen Nachbarzahlen auch im selben Scale machen ferner deutlich, warum nur die Frequencies der Histogramme - also relative Werte - Bedeutung für die letztlichen Featurevektoren haben. Mit Blick auf die für diesen Ansatz relevanten Klassen stellt sich die Frage, welche Scales sinnvoll und nötig sind.

Für die unteren Scales wird hier die Grenze bei $3cm$ angesetzt. Dies ist gering genug für die Feinheiten der Flickstellen oder potenziell sehr kleine Schlaglöcher. Um aussagekräfige Features aus Nachbarschaften zu extrahieren und nicht zu stark von Messungenauigkeiten beeinflusst zu werden, benötigt es eine gewisse Grundanzahl an Nachbarn. Die noch kleineren Scales erfüllen dies nicht und werden daher nicht verwendet. Bei den größeren Scales muss abgewogen werden zwischen ihrem zusätzlichen Informationsgehalt und der Verarbeitungsdauer. Den wohl meisten Nutzen aus jenen Scales ziehen die mittigen Punkte der größeren Gullys: Durch ihre lokale Ebenheit wird die Zugehörigkeit zu einer anderen Klasse erst bei Betrachtung weiter entfernter Nachbarn potenziell erkennbar. Allerdings genügt auch für diese Gullys ein Scale von etwa $25cm$ von der Mitte bis zum

charakteristischen Innenring, weshalb dies die naheliegende obere Schranke darstellt. Da zumindest diese Arbeit einen stärkeren Fokus auf die Schadenserkennung legt als auf die vollständige Identifikation von Gullys, liegt die obere Schranke für die standardmäßigen Versuche bei nur $15cm$. Im Kapitel *Evaluierung* findet sich auch ein Vergleich zu einem Experiment mit zusätzlichem $20cm$-Scale. In der Range von 3 bis $15cm$ wird nun jeder Scale im Abstand von wiederum $3cm$ für die Featuregewinnung genutzt. Dies soll einerseits eine angemessene Menge von Scales darstellen, andererseits typische Größen von anzufindenden Schlaglöchern repräsentieren.

4.3.3 Ein- und mehrwertige Features

diese Klassenstruktur aktuel noch nicht umgesetzt, aber einfach und die Tage kommend

Im Zuge dieser Arbeit wurde im PCTool eine eigene Histogramm-Datenstruktur implementiert. Das geschah zum einen aus Mangel an Alternativen, zum anderen, um für den Feature-Extraction-Ansatz nötige Methoden wie das Distanzmaß oder die Berechnung einer durchschnittlichen Repräsentation (siehe Abschnitt *Uniqueness*) direkt einzubauen. Für letztere werden die Counts der einzelnen Bins entsprechend erhöht (ggf. mit Gewichtung) und die Frequencies aktualisiert. Ansonsten werden erwartbare Funktionalitäten bereitgestellt wie die Rückgabe eben dieser Frequencies.
Zum Zwecke der Erweiterbarkeit und optimierter Speichernutzung ist `Histogram` eine abstrakte Klasse, die nur die Counts speichert. Wie bereits erwähnt, kann ein Histogramm linear oder nicht-linear sein, je nachdem, ob dessen Bins einen kontinuierlichen und gleichmäßig aufgeteilten Wertebereich aufspannen. Diese Arten werden in den beiden Unterklassen `LinearHistogram` und `NonLinearHistogram` umgesetzt mit ihren entsprechenden internen Arbeitsweisen. In diesem Ansatz wurden keine nicht-linearen Histogramme genutzt, was insbesondere an ihrer verhältnismäßig aufwändigen Berechnung liegt: Statt die Bins direkt in einer Iteration der Daten zu befüllen, ist eine vorherige Sortierung dieser nötig. In Verbindung mit größeren Scales macht sich die $nlog(n)$-Komplexität in der Laufzeit bemerkbar.
Die dritte Unterklasse ist das `AbsoluteHistogram`. Dieses speichert (mehrere) einwertige Features anstatt echter Bins, wobei keine Counts, sondern nur die Frequencies existieren. Es ist also der grundlegenden Bedeutung nach kein Histogramm, seine Frequencies müssen summiert auch nicht 1 ergeben. Für diesen Ansatz liegen die einwertigen Features aber alle in der Range von 0 bis 1. Es ist vornehmlich eingeführt worden für den einheitlichen Umgang mit ein- und mehrwertigen Features bezüglich der Speicherung und des Distanzmaßes, obwohl die Semantik eine andere ist.
Bedingt durch die unterschiedlichen Klassen können, wie weiter oben geschildert, die Werte wie etwa Curvature in einem Scale deutlich zwischen den Punkten variieren. Da dennoch Minima und Maxima in den Histogrammen abgebildet werden sollen, hat dies - auch wegen der Nutzung linearer Histogramme - teilweise viele leere Bins zur Folge, d.h. Bins mit einem Count und demzufolge einer Frequency von 0. Um während der Featuregewinnung in der PCTool-Node etwas Speicher zu sparen, werden solche Folgen von leeren Bins verkürzt abgespeichert: Statt k ($k \geq 2$) konsekutiver 0.0-Einträge wird ein Paar geschrieben aus Erkennungssymbol für diese Verkürzung (etwa `infinity`) und dem

k selbst. Beim abschließenden Schreiben der Featuresdatei können die Featurevektoren von Interesse dann wieder leicht in ihre ursprüngliche Größe überführt werden. Um ferner etwas Noise zu reduzieren, der zum Beispiel durch einen einzelnen Nachbarn verursacht wurde, wird ein Schwellwert von 0.005 genutzt statt des festen Werts von 0.0, um als "leerer" Bin zu gelten. Zwar sorgt dies dafür, dass die Summe aller Frequencies eines Histogramms in den Featurevektoren nicht mehr unbedingt gleich 1 sein muss; dies führte allerdings zu keinen signifikanten Änderungen der Predictions.

4.3.4 Uniqueness

Die Meanrepräsentation eines Scales, also der Durchschnitt der Featurerepräsentationen im jeweiligen Scale von allen Punkten, wird ungewichtet berechnet durch die Meanmethode der Histogram-Klasse. Das genutzte Distanzmaß zwischen den Featurerepräsentationen ist die *Kullback-Leibler-Divergenz (KL-Divergenz)* . Dieses ist originär ein nicht-kommutatives Maß über zwei Wahrscheinlichkeitsverteilungen P und Q und drückt den Grad ihrer Verschiedenheit aus. In diesem Ansatz werden keine Wahrscheinlichkeitsverteilungen in jenem Sinne verwendet, doch die mehrwertigen Featureräume in Form von Histogrammen erfüllen deren nötige Eigenschaften: Alle Einträge, in diesem Fall die Frequencies der Bins, liegen zwischen 0 und 1 und ihre Gesamtsumme beträgt genau 1. Für einwertige Featureräume ist die zweite Eigenschaft nicht erfüllt, sie machen aber nur einen kleinen Teil der vollständigen Featurevektoren aus und werden aus diesem Grund dennoch identisch behandelt. [13] nutzen die KL-Divergenz ebenfalls als eines von mehreren getesteten Distanzmaßen. P steht hierbei jeweils für die Featurerepräsentation eines Punktes, Q immer für die Meanrepräsentation eines Scales. Die Formel zur Berechnung der Distanz zwischen diesen Featurerepräsentationen, wobei $P(i)$ und $Q(i)$ die Frequencies der i-ten Bins und n die Binzahl bezeichnen, lautet:

$$D(P||Q) = \sum_{i=1}^{n} (P(i) - Q(i)) * \log_2 \frac{P(i)}{Q(i)} \tag{4.2}$$

Die KL-Divergenz ist nur definiert, wenn für alle Einträge i mit $Q(i) = 0$ auch $P(i) = 0$ gilt. Dies ist hier der Fall, da ein Eintrag von 0 im Mean immer bedeutet, dass der entsprechende Eintrag auch bei der Repräsentation jedes Punktes gleich 0 ist. Für $P(i) = 0$ ist der Summand als 0 definiert. Von der Liste aller Distanzen werden daraufhin der Durchschnitt sowie die Standardabweichung berechnet. Alle Punkte mit einer Distanz größer-gleich dem Durchschnitt plus `alpha`-Mal der Standardabweichung gelten als unique. Dabei ist `alpha` ein einstellbarer Parameter und je größer dieser ist, desto vorsichtiger und eindeutiger verläuft die Einstufung als unique. In den Experimenten hat sich generell ein Wert von 1.0 als annehmbar dargestellt.
Damit bei sehr unterschiedlichen Binzahlen der verschiedenen Featureräume - wenn etwa Curvature doppelt so viele Bins hätte wie die Intensitätsdifferenzen - kein Featureraum einen zu starken Einfluss auf die Entscheidung zur Uniqueness hat, wird letztere mit obigem Verfahren separat für jeden Featureraum berechnet. Sobald ein Punkt in mindestens einem Featureraum als unique gilt, tut er dies für den gesamten Scale. Dieses Prinzip ist

vor allem mit Blick auf potenzielle zukünftige Erweiterungen eingeführt worden, sollten zum Beispiel neue Features hinzukommen oder sich die Kennzahlen der Histogramme ändern.

4.3.5 Approximation für größere Scales

Bei zu großen Scales wird auf der Punktwolke eine Density Reduction durchgeführt. Dazu wird intern eine Kopie der Originalpunktwolke angelegt, die sogenannte *Half Edge Length* bestimmt, welche die Stärke der Dichtereduktion festlegt, und letztere entsprechend vorgenommen. Auf dieser reduzierten Punktwolke werden nun ebenfalls ein k-d-Baum für den effizienten Zugriff erzeugt sowie die Features, wie gehabt, berechnet. Der Schwellwert für die Operation liegt bei $20cm$. Für kleinere Scales ist der Mehraufwand mit der Kopie, der Dichtereduktion und der gleich erklärten Approximation zu groß im Vergleich zur standardmäßigen Prozessierungszeit ohne Density Reduction.
Die für die reduzierte Anzahl an Punkten berechneten Features müssen schließlich auf die originale Menge zurück überführt werden. Der Ansatz ist hierbei der folgende: Für jeden Originalpunkt P wird dessen Nachbarschaft $\mathfrak{N}$ in der reduzierten Punktwolke bestimmt. Der Scale für die Suche beträgt $3cm$, um keine zu weit entfernten Punkte und somit Featurevektoren einzubeziehen. Für den Ausnahmefall, dass sich kein einziger Punkt in diesem Scale befindet, wird lediglich der nächste Punkt (kNN = 1) herangezogen. Da nähere Punkte eher den tatsächlichen Featurevektor von P für diesen Scale repräsentieren, sollen sie auch stärker bei der Approximation gewichtet werden. Dies geschieht über die inverse Distanz zu P, wobei dieser Wert - insbesondere, damit die einwertigen Features im Wertebereich von 0 bis 1 verbleiben - noch durch die Summe aller inversen Distanzen normalisiert wird:

$$\omega_i = \frac{1}{d(P,\mathfrak{N}_i)} / \sum_{j=1}^{|\mathfrak{N}|} \frac{1}{d(P,\mathfrak{N}_j)}, i \in \{1, ..., |\mathfrak{N}|\} \tag{4.3}$$

Dabei entspricht d der Distanzfunktion im euklidischen Raum. Mithilfe der Gewichtungen und der Featurevektoren $\mathfrak{F}$ der Nachbarschaft kann schließlich der Featurevektor F von P als gewichteter Durchschnitt berechnet werden [12], umgesetzt durch die *Mean*-Methode der *Histogram*-Klasse:

$$F_P = \sum_{i=1}^{|\mathfrak{N}|} \omega_i * \mathfrak{F}_{\mathfrak{N}_i} \tag{4.4}$$

4.3.6 Prediction per Random Forest

Für die Predictions mittels Random Forest wird das in Python wohlbekannte Paket *scikit-learn* verwendet. Dieses bietet vielseitige, einfach zu bedienende Funktionalitäten im Bereich Machine Learning. Dazu zählen neben Modellen verschiedener Art auch Funktionen zum Berechnen von Qualitätsmetriken solcher trainierter Modelle.
Die Python-Skripte des Ansatzes lassen sich über ein Kommandozeilen-Interface bedienen:

- Es kann ein Random Forest erstellt und trainiert werden. Dazu muss eine Featuresdatei eingelesen werden sowie eine Punktwolke mit den Ground-Truth-Klassen. Darauf kann das Modell dann *gefittet* werden.

- Es kann ein Random Forest getestet werden auf seine Qualität. Dazu muss neben dem trainierten Modell erneut eine Featuresdatei eingelesen werden sowie eine Punktwolke mit den Ground-Truth-Klassen. Anschließend predictet der Random Forest die Klassen der entsprechenden Punkte, was mit den tatsächlichen abgeglichen und in Metriken pro Klasse ausgedrückt wird. Zu den hier verwendeten Metriken siehe Kapitel *Evaluierung*.

- Es können die Klassen einer noch nicht annotierten Punktwolke predictet werden. Dazu muss neben dem gewünschten Modell die entsprechende Featuresdatei eingelesen werden sowie die Punktwolke zum Schreiben der Klassen. Dieser Teil ist für den regelmäßigen Einsatz auf der Plattform der bedeutsamste.

Wie im zugehörigen Abschnitt des Kapitels *Konzepte* geschildert, lässt sich ein Random Forest an einigen Stellen parametrisieren. Die einzigen beiden Parameter, die für diesen Ansatz manuell gesetzt wurden, sind die Gesamtzahl der Decision Trees, aus denen der Random Forest besteht, sowie die maximale Tiefe pro Baum. Die Anzahl an *Estimators*, wie es bei scikit-learn heißt, liegt für die Experimente bei 100. Es wurden auch höhere Zahlen getestet, die aber keine signifikanten Steigerungen bei der Qualität der Predictions brachten. Während dieser Wert dem von [17] gleicht, liegt die maximale Tiefe eines Baums hier bei 20 statt bei 10. Dies kann damit begründet werden, dass im vorliegenden Ansatz deutlich größere Featurevektoren genutzt werden: statt 48 beinhalten sie hier XXX Werte. Eine noch größere Tiefe brachte bei Versuchen allerdings keine Verbesserungen mehr - im Gegenteil scheint dann der Random Forest zu stark zu overfitten auf die Trainingsdaten und die individuellen Entscheidungen pro Baum zu spezialisiert zu treffen. Generell ist aber festzuhalten, dass der Fokus nicht auf einer detaillierten Evaluierung des Random Forests und dessen Parameter lag, sondern eher auf dem Finden von aussagekräftigen Features.

4.4 Ansatz Deep Learning

Da auf die Funktionsweise von PointNet hier kein Einfluss geübt werden kann, beschränken sich die Experimente, deren Ergebnisse im Kapitel *Evaluierung* vorgestellt werden, auf das Nutzen verschiedener Parameterkombinationen. Die letztlich für die Klassifizierung relevanten Parameter des Modells bzw. für dessen Einsatz durch PCNN sind[1]:

1. der *Query Radius*. Dieser Wert gibt an, wie weit ein Punkt der Nachbarschaft maximal entfernt sein darf vom Ursprungspunkt. Standardmäßig auf $3m$ gesetzt für größere Objekte wie Häuser und Bäume, sollte er für den Anwendungsfall dieser Arbeit entsprechend reduziert werden. Da nur ein fester Wert einstellbar ist, werden Experimente durchgeführt für jeden der im Feature-Extraction-Ansatz genutzten Scales. In der Evaluierung wird dabei auch auf die Unterschiede in der jeweiligen Klassifikationsleistung eingegangen.

[1] https://gitlab.hpi3d.de/pcr/pcnn-docs/-/blob/main/latex/pcnn.pdf

2. die *Neighborhood Size*. Hiermit wird bestimmt, wie groß jede betrachtete Nachbarschaft ist. Weil auch dieser Wert für alle Punkte gleich bleibt, aber nicht jede Nachbarschaft dieselbe Punktanzahl enthält, muss ggf. aufgefüllt werden durch Mehrfachnutzung einzelner Punkte. Damit diese Ausnahmebehandlung nicht allzu häufig Anwendung findet, orientiert sich die genutzte Nachbarschaftsgröße in den Experimenten am jeweiligen Query Radius. Genauer bedeutet dies, dass der Wert nahe der durchschnittlichen Nachbarzahl für den entsprechenden Scale liegt.

3. der *Length Divider*. Dieser PCNN-eigene Parameter hat einen Einfluss darauf, wie viele Samplepunkte aus der Punktwolke gezogen werden. Je kleiner der Wert, desto mehr Punkte werden betrachtet und umgekehrt. Der für die Experimente genutzte Wert XXX

4. die Anzahl an Epochen. Eine Epoche ist dann beendet, wenn alle Trainingsdaten das Modell einmal durchlaufen haben. Mit mehr Epochen steigt grundsätzlich die Genauigkeit des Modells, die mit wenigen Epochen noch eher zufällig ist. Allerdings muss dabei beachtet werden, dass zu viele Epochen Overfitting verursachen können. Bemerkbar macht sich dies dann in geringeren Genauigkeiten bei neuen Daten und letztlich schlechteren Predictions. Für die Experimente wurde eine Anzahl von XXX

4.5 Postprocessing

Das Postprocessing wird in zwei Iterationen ausgeführt, wobei in beiden jeweils eine Punktenachbarschaft im Scale von 7,5cm betrachtet wird. Auch hier wird dazu ein zuvor für die Punktwolke ersteller k-d-Baum genutzt, der an dieser Stelle entsprechend eine Python-Implementierung besitzt.
Zunächst wird über die als einfache Straße klassifizierten Punkte iteriert und der Prozentsatz an Punkten im Scale ermittelt, der nicht Straße ist. Übersteigt dieser die Marke von 50%, wird dem Ursprungspunkt die häufigste Nicht-Straßenklasse seiner Nachbarschaft zugeteilt. Dies soll Punkte etwa im Inneren von Schlaglöchern oder Gullys ausbessern, die fälschlicherweise als Straße klassifiziert wurden, ggf. weil nicht als unique markiert. Für eine weitere Zeitersparnis wird diese Iteration nur auf den Straßenpunkten ausgeführt, die im obigen Scale mindestens eines Nicht-Straßenpunktes liegen, da auch nur jene für diese Erweiterung interessant sind. Die Ermittlung dieser Straßenpunkte kann zuvor erfolgen und ist deshalb effizient, da die Menge der Straßenpunkte die der Nicht-Straßenpunkte in mehreren Größenordnungen übersteigt.
Die zweite Iteration zielt darauf ab Außenseiterpunkte, die nicht als Straße klassifiziert wurden, zu entfernen, d.h. ihnen die Klasse Straße zu geben. Dies wird genau dann durchgeführt, wenn im Scale nicht mindestens 10% der Punkte dieselbe Klasse wie der Ursprungspunkt besitzen. Dies funktioniert gut für Klassen, die wie der Scale auch eher rund sind, darunter Gullys und Schlaglöcher. Für Flickstellen kann dies problematischer sein, da diese eher dünn und linienartig verlaufen. Außerdem ist die Erkennung von Flickstellen - wie im Kapitel *Evaluierung* nachzulesen - teilweise unstet, was dieser Methode

der Nachverarbeitung nicht zugutekommt. Daher werden die als Flickstelle klassifizierten Punkte in dieser Iteration übersprungen.
Wie im zugehörigen *Konzepte*-Abschnitt erläutert, ist dieses heuristische Postprocessing eher vorsichtig gewählt und soll vermeintlich eindeutige Fehlklassifzierungen ausbessern. Die ursprünglichen Klassifizierungen haben jedoch, ob beim Feature-Extraction-Ansatz oder bei PointNet, eine tiefergehende Begründung und letztlich Legitimation.

Kapitel 5

Evaluierung

5.1 Trainings- und Testdaten

5.2 Metriken

Um die Predictions von Modellen quantitativ evaluieren zu können, existieren verschiedene Maße. Zwei für Klassifizierungen häufig genutzte sind Precision und *Recall*, die sowohl für eine gesamte Prediction als auch pro Klasse berechnet werden können. Auf eine Ausgangsklasse $\mathfrak{K}$ und Predictions von Punktwolken bezogen, drückt der Precision-Wert P den Prozentsatz der Punkte aus, die als $\mathfrak{K}$ klassifiziert wurden und tatsächlich $\mathfrak{K}$ sind. Der Recall-Wert R hingegen gibt den Prozentsatz aller $\mathfrak{K}$-Punkte an, die "gefunden" und korrekt als $\mathfrak{K}$ klassifiziert wurden. Je nach Anwendungsfall kann der eine oder der andere Wert bedeutsamer sein und höher gewichtet werden. Für die in dieser Arbeit behandelte Schadenserkennung sollen beide gleich stark gewichtet werden. Um beide Werte zu einem einzelnen zusammenzufassen und dabei besonders niedrige Ausschläge in einem der beiden zu bestrafen, kommt oft der F_1-Score zum Einsatz:

Paper?

$$F_1 = \frac{2 * P * R}{P + R} \tag{5.1}$$

Für die kommenden Experimente werden jeweils Tabellen aufgeführt mit diesen drei Metriken. Dabei werden sie nur pro Klasse berechnet, da sie bezogen auf die Gesamtklassifzierung - aufgrund der um ein Vielfaches höheren Zahl an gewöhnlichen Straßenpunkten gegenüber dem Rest - wenig Aussagekraft besitzen bezüglich der Qualität des Ergebnisses.

5.3 Preprocessing

5.4 Ansatz Feature-Extraction

5.4.1 Ergebnisse

5.4.2 Vergleich mit nur einwertigen Features

5.4.3 Vergleich mit größeren Scales

5.4.4 Vergleich ohne Uniqueness

5.5 Ansatz Deep Learning - Ergebnisse

5.6 Postprocessing

Kapitel 6

Fazit und Ausblick

Literaturverzeic
korrigieren bzg
Article/Inproce
dings, Editors
und Roberts
Titel ergänzen

Reihenfolge des
Literaturverzec
nis ordnen

Fazit schreiben
(inkl. kurzer Zu
sammenfassung

Für den Feature-Extraction-Ansatz bieten sich verschiedene Verbesserungen und Erweiterungen an. Zum einen könnte die Integration in ein einziges abgeschlossenes System vollzogen werden, vorzugsweise das PCTool, um den zusätzlichen Overhead von Datentransfers zu vermeiden. Für die Straßenextraktion, einer notwendigen Vorstufe für die Schadenserkennung, könnten elaboriertere Verfahren entwickelt werden, die den Anteil der reinen Straße gegenüber Gehweg und Noise weiter erhöhen. Diese können neben der bisherigen Punktdichtebetrachtung etwa auch auf der Erkennung von Bordsteinkanten als seitliche Grenzen basieren.
Ein großer Fokus könnte auf der Nutzung von noch aussagekräftigeren Features liegen, die ja die Basis aller Predictions sind. Das können sowohl Variationen der bisher genutzten Features sein - darunter fallen auch nicht-lineare Histogramme, welche die vorkommenden Werte besser abbilden können - als auch komplett andere Maße. Globale Features mit der gesamten Punktwolke als Ausgangslage können ebenfalls eingearbeitet werden und zusätzliche Informationen in die Featurevektoren bringen, die mit lokalen Nachbarschaften allein nicht auszudrücken sind.
Insbesondere für die Berechnung der uniquen Punkte dürfte eine initiale Ermittlung des groben Straßenzustands hilfreich sein. Dies könnte durch einen weiteren Fingerabdruck der Straßenpunktwolke repräsentiert werden und beispielsweise ausdrücken, ob es sich eher um eine Straße in der Innenstadt in sehr gutem Zustand handelt oder um eine marode Landstraße. Diese Information würde schließlich genutzt werden, um Parameter des Uniqueness-Konzepts sowie Ober- und Untergrenzen der Histogramme besser abzuschätzen und letztlich sowohl die ermittelte Uniqueness als auch die Predictions für die konkrete Punktwolke zu verbessern. Mit einer so verfeinerten Uniqueness könnte außerdem die Geschwindigkeit und Genauigkeit einer Registration gesteigert werden, die vor allem auf den beständigen Gullys basiert. In Verbindung damit ist auch eine Dokumentation der Zustandsentwicklung einer Straße denkbar: Anhand der Änderungen der uniquen Punkte, zuvorderst der Schadensklassen, über einen gewissen Zeitraum könnten Problemstellen ausgemacht und eventuell sogar prognostiziert werden.
Eine Fortführung der Schadenserkennung mittels Deep-Learning-Ansätzen ist ebenfalls denkbar. Dabei könnten zum einen noch mehr Konfigurationen von PointNet auf ihre Qualität getestet werden. Zum anderen wären auch gänzlich andere Architekturen interessant, die auf direkter Punktbasis arbeiten, wozu auch der Nachfolger von PointNet zählt. Allen Ansätzen gemein ist jedoch ein ausgereifteres Postprocessing, das seinen

Fokus je nach Anwendungsfall anders legen könnte. Falls zum Beispiel Flickstellen besser und vollständiger erkannt werden sollen, können die einzelnen erkannten Cluster zu annähernd Rechtecken verbunden werden. Wenn hingegen vor allem die Detektion der Gullys relevant ist, kann mehr Aufwand betrieben werden, die Umrisse und das planare Innere präzise zu erkennen.
Wie generell für Machine-Learning-basierte Ansätze wären auch hier mehr und vor allem unterschiedlichere Trainingsdaten wichtig, um auf vielfältigen realen Daten gute Ergebnisse zu liefern. Dazu zählen etwa Schlaglöcher mit größeren Tiefen oder länglicheren Umrissen. Auch Gullys mit anderen Größen und weitere Klassen wie Risse oder sonstige Straßenobjekte könnten von Interesse sein. Außerdem würden bei weiteren Daten auch mehr verschiedene Scanner zum Einsatz kommen, wofür sich das Preprocessing der Intensitäten auszahlen sollte. Auf diese Weise könnte der bisherige *Proof of Concept* erweitert werden, um die Vielfalt und Komplexität der Realität besser zu verarbeiten.
Schließlich könnte mit der Erkennung von Schäden und Unzulänglichkeiten einer Straße auch eine Bewertung ihres Zustands erfolgen. Dabei sind verschiedene Verfahren denkbar, etwa eine einfache Kumulation der Punkte pro Schadensklasse in der gesamten Punktwolke. Solche Schritte könnten jedoch auch in kleinerem Maßstab vorgenommen werden, indem mit gängigen Geoinformationssystemem gleichmäßige Raster in der Punktwolke erzeugt und einzeln bewertet werden. Je nach notwendiger Genauigkeit der Zustandsbewertung könnte die Klassifikation in feineren Abstufungen geschehen: So würden zum Beispiel Schlaglöcher anhand ihrer Größe und Tiefe eingeteilt werden in *kleiner*, *mittlerer* oder *großer* Schaden. In jedem Fall wäre eine solche Bewertung kleinerer Abschnitte dazu geeignet, einfach Priorisierungen für Reparaturen zu schaffen und Straßenschäden somit schneller und effizienter zu beheben.

Literaturverzeichnis

[1] R. Qi Charles, Hao Su, Mo Kaichun und Leonidas J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Hrsg. von XXX. 2017.

[2] R. T. H. Collis. “Lidar”. In: *Applied Optics* 9.8 (1970).

[3] Afshin Famili, Alireza Shams und William J. Davis. “Application of Mobile Terrestrial LiDAR Scanning Systems for Identification of Potential Pavement Rutting Locations”. In: *Transportation Research Record Journal of the Transportation Research Board* XXX.XXX (2021).

[4] Xian-Feng Han, Jesse S. Jin, Juan Xie, Mingjie Wang und Wei Jiang. “A comprehensive review of 3D point cloud descriptors”. In: *ArXiv* abs/1802.02297 (2018).

[5] Huy Tho Ho und Danny Gibbins. “Curvature-based approach for multi-scale feature extraction from 3D meshes and unstructured point clouds”. In: *IET Computer Vision* 3.4 (2009).

[6] Kai Robert Kirsten. “XXX”. Magisterarb. Hasso-Plattner-Institut, Universität Potsdam, 2021.

[7] Wilfried Linder. *Digital Photogrammetry, A Practical Course*. Hrsg. von XXX. Springer, 2009.

[8] Paul Mattes. “Analyse und Evaluation von Heuristiken für die optimierte Anwendung von Deep-Learning-Modellen auf Punktwolken”. Magisterarb. Hasso-Plattner-Institut, Universität Potsdam, 2021.

[9] Niloy J. Mitra und An Nguyen. “Estimating surface normals in noisy point cloud data”. In: *Proceedings of the nineteenth annual symposium on Computational geometry*. 2003.

[10] Fabio Remondino und Sabry El-Hakim. “Image-based 3D modelling: a review”. In: *The Photogrammetric Record* 21.115 (2006).

[11] Peter J. Rousseeuw und Mia Hubert. “Robust statistics for outlier detection”. In: *WIREs Data Mining Knowledge Discovery* 1.73 (2011).

[12] Radu Bogdan Rusu, Nico Blodow und Michael Beetz. “Fast Point Feature Histograms (FPFH) for 3D Registration”. In: *In Proceedings of the International Conference on Robotics and Automation (ICRA)*. Hrsg. von XXX. 2009.

[13] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow und Michael Beetz. “Persistent Point Feature Histograms for 3D Point Clouds”. In: *In Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10)*. Hrsg. von XXX. 2008.

[14] Martin Schilling. “Konzeption und Implementierung einer webbasierten Plattform zur Verarbeitung and Analyse von 3D-Punktwolken”. Magisterarb. Hasso-Plattner-Institut, Universität Potsdam, 2021.

[15] Hugues Thomas, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette und Yann Le Gall. “Semantic Classification of 3D Point Clouds with Multiscale Spherical Neighborhoods”. In: *International Conference on 3D Vision (3DV)* 10.1109 (2018).

[16] M. Zaboli, H. Rastiveis, A. Shams, B. Hosseiny und W. A. Sarasua. “Classification of Mobile Terrestrial LiDAR Point Cloud in Urban Area Using Local Descriptors”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-4.W18 (2019).

[17] Li Zhiqiang, Cheng Chengqi, Kwan Mei-Po, Tong Xiaochong und Tian Shaohong. “Identifying Asphalt Pavement Distress Using UAV LiDAR Point Cloud Data and Random Forest Classification”. In: *International Journal of Geo-Information* 8.39 (2019).

Todo list

- hier Zmsf. des Vergleichs der Ansätze bzgl. Qualität und Performance iii
- Terminologie beenden 1
- Was anderes als Fußnoten nutzen? 9
- hier vllt. was von der Stadt Essen 10
- Paper? 12
- Genutzte Features bzw. Binanzahl und -grenzen werden nach letzten Experimenten eingefügt! ('XXX') 21
- Noch vor neuer Einleitung geschrieben; wird hier evtl. verkürzt 21
- vllt. Ablaufdiagramm zum besseren Verständnis? 24
- Pseudocode des Ablaufs? Anschauungsbild folgt noch 25
- diese Klassenstruktur aktuell noch nicht umgesetzt, aber einfach und die Tage kommend 27
- Paper? 28
- Paper? 33
- Literaturverzeichnis korrigieren bzgl. Article/Inproceedings, Editors und Roberts Titel ergänzen 35
- Reihenfolge des Literaturverzechnis ordnen 35
- Fazit schreiben (inkl. kurzer Zusammenfassung) 35

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe des Literaturzitats gekennzeichnet. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Potsdam, 21. Juli 2021	
(Ort, Datum)	(Unterschrift)