[redacted]

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Philipp Gehring

Verfahren zur Erstellung von Referenzkarten aus kamerabasierten Punktewolken für die Validierung von Parkassistenzsystemen

Fakultät Technik und Informatik
Department Maschinenbau und Produktion

Faculty of Engineering and Computer Science
Department of Mechanical Engineering and Production Management

Philipp Gehring

Verfahren zur Erstellung von Referenzkarten aus kamerabasierten Punktewolken für die Validierung von Parkassistenzsystemen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Maschinenbau – Entwicklung und Konstruktion
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Robert Bosch GmbH
Chassis Systems Control/Engineering Parking Base Development 3
Burgenlandstraße 33
70469 Stuttgart

Erstprüfer: Prof. Dr. Thomas Frischgesell
Zweitprüfer: M.Eng. Jürgen Schmidt

Abgabedatum: 1. Dezember 2020

Zusammenfassung

Philipp Gehring

Thema der Bachelorthesis

Verfahren zur Erstellung von Referenzkarten aus kamerabasierten Punktewolken für die Validierung von Parkassistenzsystemen

Stichworte

Photogrammetrie, Szenen-Rekonstruktion, Kartierung, Assistenzsystem, Parken

Kurzzusammenfassung

Im Rahmen dieser Arbeit wurden Methoden für ein Verfahren zur Erstellung einer Referenzkarte aus kamerabasierten Punktewolken untersucht. Die Referenzkarte dient der Validierung von Park- und Manövrierassistenzsystemen. Das entwickelte Verfahren wurde prototypisch implementiert und anhand einer LiDAR-Referenzmessung validiert.

Philipp Gehring

Title of the paper

Process for the creation of reference maps from camera-based point clouds for the validation of parking assistance systems

Keywords

photogrammetry, scene reconstruction, mapping, assistance system, parking

Abstract

Within the scope of this work, methods for a procedure for creating a reference map from camera-based point clouds were investigated. The reference card is used to validate parking and maneuvering assistance systems. The developed procedure was implemented as a prototype and validated using a LiDAR reference measurement.

Danksagung

An dieser Stelle möchte ich mich bei der Robert Bosch GmbH für die Möglichkeit der Absolvierung meines Hauptpraktikums, sowie für das spannende Thema der Bachelorarbeit bedanken. Ebenso bedanke ich mich für die freundliche Aufnahme in das Team der Abteilung CC-DA/EPB3 von Tom Reimann.

Des Weiteren möchte ich mich bei meinem Erstprüfer Herrn Professor Frischgesell für die Betreuung der Arbeit bedanken. Ein besonderer Dank gilt meinem Zweitprüfer Herrn Jürgen Schmidt für die stets hilfreichen Diskussionen und Kommentare.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Nomenklatur

Abkürzungen

Abkürzung	**Beschreibung**
3D	dreidimensional
Euro NCAP	European New Car Assessment Programme
fps	frames per second (Bilder pro Sekunde)
KOS	Koordinatensystem
LiDAR	Ligth Detection and Ranging
MP	Megapixel
PKW	Personenkraftwagen
px	Pixel
RANSAC	Random Sample Consensus
RMSD	root mean squared deviation (kleinste quadrierte Abweichung)
SVD	singular value decomposition (Singuläre-Werte-Zerlegung)
VHM	Vehicle Model
2D	zweidimensional

Formelzeichen

(c_x, c_y)	Bild-Hauptpunkt
(u, v)	Bildkoordinate in px
f -	Brennweite
$\vec{e}_{1x}$	Einheitsvektor x Koordinatensystem Boden
$\vec{e}_{1y}$	Einheitsvektor y Koordinatensystem Boden
F_C	Hauptpunkt-Projektion
$\vec{X}_C$	Kamerakoordinatensystem, homogen
$\mathbf{K}$	Kameramatrix
$\mathbf{H}$	Kovarianzmatrix
$\mathbf{U}$	Links-Singulärwerte (hier: Links-Drehmatrix)
μ	Mittelwert
$\vec{n}$	Normalenvektor der Ebene
$\mathbf{P}$	Proektionsmatrix
$\mathbf{P}, \mathbf{Q}$	Punktematrix mit nx3 Elementen
$\mathbf{V}$	Rechts-Singulärwerte (hier: Rechts-Drehmatrix)
$\mathbf{R}\|\mathbf{t}$	Rotationsmatrix mit Translationsvektor in der letzten Spalte
$\mathbf{R}_{01}$	Rotationsmatrix Weltkoordinaten zu Bodenkoordinaten
T	Schwellwert Statistical Outlier Filter
$\mathbf{\Sigma}$	Singulärwerte
s	Skalierungsparameter Sznenprojektion
σ	Standardabweichung
p	Stützpunkt der Ebene
$\vec{X}_W$	Szenen Koordinatensystem
$\vec{X}_{W,P}$	Szenen-Punkt
$\mathbf{T}_{sfm2VHM}$	Transformationsmatrix Bodenkoordinaten zu VHM Koordinaten
$\mathbf{T}_{01}$	Transformationsmatrix Weltkoordinaten zu Bodenkoordinaten
$\vec{r}_{mm}$	Verschiebungsvektor Ebenenpunkte Boden
D	vorzeichenbehafteter Abstand Punkte zu Ebene

1. Einleitung

Die Zahl der zugelassenen Personenkraftwagen (PKW) in Europa nimmt stetig zu. Im Jahr 2013 lag die Zahl der zugelassenen PKW in Europa bei fast 12 Millionen, bis 2018 ist die Zahl auf 16 Millionen PKW um ein Drittel angestiegen [1]. Die Verkehrsdichte nimmt besonders im städtischen Bereich stark zu. Es ist beim Einparken oft unübersichtlich und hektisch, wie beispielsweise auf Supermarktparkplätzen oder an einer befahrenen Straße. Hier müssen alle Bereiche um das Fahrzeug vom Fahrer im Auge behalten werden. Auch dynamische Objekte, wie zum Beispiel Fußgänger, Radfahrer oder andere Verkehrsteilnehmer, erscheinen und verschwinden oft in der Szene um das Fahrzeug. Um einen guten Überblick über die Szene zu behalten, ist viel Vorsicht geboten. Die Unterstützung durch Assistenzsysteme gewinnt daher an Beliebtheit und soll in Verkehrssituationen, die Notbremsungen erfordern, sogar zur Pflicht werden [2].

Auch das *European New Car Assessment Programme* (Euro NCAP) bezieht seit 2014 aktive Sicherheitssysteme mit in ihre unabhängige Bewertung der Fahrzeugsicherheit ein. Seit 2016 werden Notbremsassistenten und Spurhaltesysteme berücksichtigt und ab dem Jahr 2020 sollen auch Parkassistenzsysteme mit in die Bewertung einfließen [3].

Die Anforderung an die Entwicklung dieser Systeme steigt somit stetig. Auch Parkassistenzsysteme, wie der Parkassistent der Robert Bosch GmbH, müssen immer komplexere Manöver sicher beherrschen. Die stetige Verbesserung der Objekterkennung und -lokalisierung ist daher zwingend erforderlich.

1.1. Park- und Manövrierassistenzsystem

Das Park- und Manövrierassistenzsystem ist ein für niedrige Geschwindigkeiten[1], zum Einparken und für langsame Rangierfahrten ausgelegtes Assistenzsystem. Es zeigt dem Fahrer die Distanz zu Objekten an und visualisiert das Fahrzeug aus der Vogelperspektive. Wenn eine passende Parklücke erkannt wurde, kann ein Parkvorgang gestartet werden. Das Fahrzeug übernimmt dann die Regelung der Lenkung und

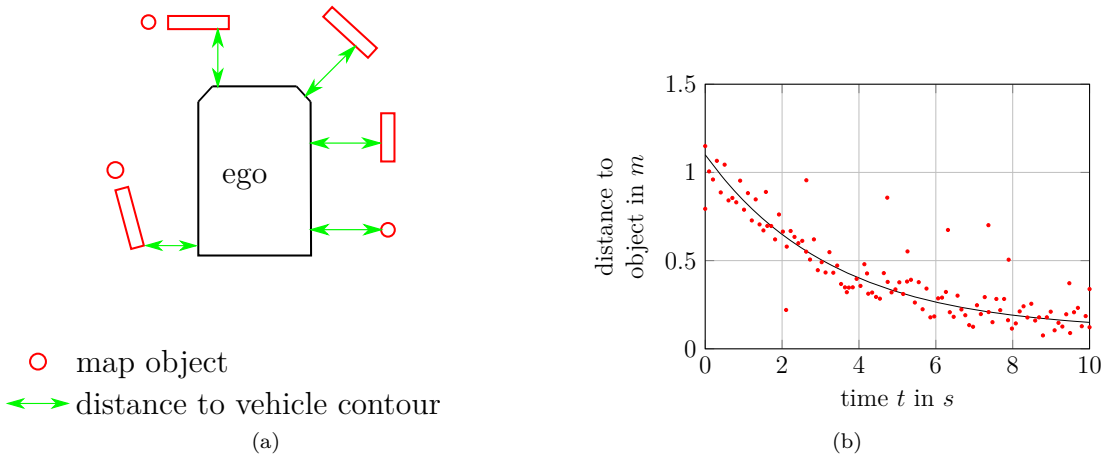
[1] <15 km/h Rangieren und Manövrieren
<60 km/h Parklückensuche

der Geschwindigkeit. Die dafür benötigte Sensorik des Systems basiert hauptsächlich auf Ultraschallsensoren (vgl. Abbildung 1.1), die in der Frontschürze und im Heck des Fahrzeugs integriert sind. Die Anzahl der Sensoren hängt von der Konfiguration des Systems ab und variiert zwischen vier und 16 Sensoren. Mit steigender Zahl der Sensoren ist eine umfangreichere Erfassung der Umgebung möglich, sodass Systeme für ein einfaches Abstandswarnsystem bis hin zu einem autonomen Parkvorgang möglich sind.

Abbildung 1.1.: BOSCH© Ultraschallsensor (Generation 6). Drei verschiedene Befestigungsvarianten des Ultraschallsensors [4].

Für die Objektdetektion werden Schallpulse ausgesendet. Über die gemessene Zeit bis zum Echo wird die Distanz zum Objekt berechnet. Um Interferenzen mit der Umwelt zu minimieren, haben die Sensoren ein eigenes Impulsmuster, dass von anderen Sensoren und Schallquellen der Umwelt unterschieden werden kann. Des Weiteren werden aus den berechneten Abständen zeitliche Verläufe (vgl. Abbildung 1.2b) gebildet, die zur Filterung der Signale genutzt werden. Mithilfe von heuristischen Methoden werden aus den Echos Objekte gebildet. Diese Objekte werden im nächsten Schritt durch die Auswertung der empfangenen Reflexionseigenschaften bezüglich ihrer Kollisionsrelevanz bewertet und als niedrig oder hoch klassifiziert. Hindernisse, die als hoch klassifiziert werden, können nicht überfahren werden. Ein Objekt, welches jedoch niedriger als die Bodenfreiheit des Autos ist, kann passiert werden und wird als niedrig klassifiziert.

Das Steuergerät bildet aus diesen Objekten eine Umgebungskarte, auf der alle erkannten Objekte gespeichert und verfolgt werden. Dabei wird das Fahrzeugumfeld durch linien- und punktförmige Objekte in der Umgebungskarte abstrahiert (vgl. Abbildung 1.2a). Die Karte dient als Grundlage zur Fahrzeugführung oder Warnung des Fahrers. Die Informationen dieser Karte und ihrer Objekte werden in jedem Messzyklus aktualisiert.



(a) (b)

Abbildung 1.2.: Objektkarte mit zeitlichem Verlauf einer Objektannäherung. (a) Schematische Darstellung der Objektkarte des Park- und Manövrierassistenzsystems. [unternehmensinterne Quelle] (b) Annäherung an ein Objekt mit einem Sensor. Distanz zum Objekt über die Zeit aufgetragen. gebildeter Zeitverlauf in Schwarz; Echos in Rot.

1.2. Referenzmesstechnik

Die Genauigkeit der Objektbildung ist entscheidend für die Leistungsfähigkeit des Systems. Das System muss dafür sehr sensitiv für sein. Falschdetektionen von Objekten müssen jedoch vermieden oder konsistent als niedrig klassifiziert werden. Es muss jedoch sichergestellt werden, dass die Umgebung dabei immer noch hinreichend genau abgebildet wird, um eine korrekte Fahrzeugführung zu ermöglichen.

Mithilfe einer Referenzmesstechnik kann in einer Szene die Abdeckung der realen Hindernisse mit Ultraschallechos validiert werden. Für eine genaue Fahrzeugführung ist eine genaue Abbildungstreue der Objekte, sowie deren Höhenklassifikation erforderlich.

Um eine aussagekräftige Validierung des Park- und Manövrierassistenzsystems zu bekommen, ist eine genauere Messmethode als die Messung mit Ultraschallsensoren erforderlich. Heute werden zur Validierung der Objektbildung *Light Detection And Ranging* (LiDAR) Sensoren eingesetzt, die auf einem Referenzfahrzeug montiert sind. Die Szene wird mit einem LiDAR Sensor gescannt und in einer Punktewolke abgebildet. Diese dient dann als Genauigkeitsreferenz. Dieses Referenzfahrzeug ist mit zusätzlicher Messtechnik ausgestattet, die die Fahrzeugsignale simultan aufzeichnet. Anschließend werden dann die in Abbildung 1.3 gezeigten Punktewolken aus den LiDAR Daten generiert und mit den aufgezeichneten Ultraschall Daten verglichen. Die Verfügbarkeit dieser Referenzfahrzeuge ist begrenzt, da kostspielige Messtechnik installiert ist. Diese Messtechnik ist unter anderem auf dem Fahrzeugdach angebracht, was die Einfahrt in Garagen oder Parkhäuser einschränkt.

Mit einer Referenzmesstechnik auf Basis der Photogrammetrie, ist es möglich die

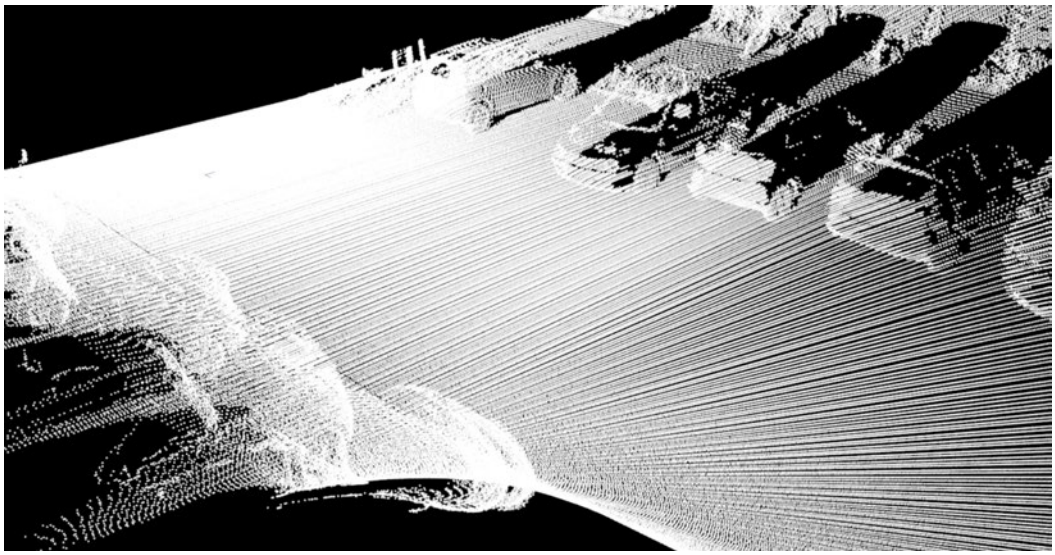


Abbildung 1.3.: LiDAR Punktewolke einer Messfahrt. 451699 gescannte Punkte des LiDAR-Referenzfahrzeugs.

statischen Szenen der Testfahrt aus Videodaten zu rekonstruieren. Hier wird auf den Einsatz teurer Sensoren verzichtet, da dies ein rein fotometrisches Verfahren ist. Die Aufnahmen können beispielsweise mit einer handelsüblichen Smartphone-Kamera aufgenommen werden. Die Szene mit den relevanten Hindernissen wird nach der Testfahrt aus mehreren Perspektiven gefilmt und kann dann mithilfe einer Photogrammetrie Software zu einer 3D-Punktewolke rekonstruiert werden. Das Rekonstruktionsverfahren wird in Abschnitt 2.1 ausführlich behandelt. Die rekonstruierte Szene kann in Form einer zweidimensionalen (2D) Karte mit den Ultraschalldaten überlagert werden. Die Ultraschalldaten können so sehr viel anschaulicher interpretiert werden, da es zusätzliche Szeneninformationen gibt.

1.3. Zielsetzung und Struktur der Arbeit

Um die Vergleichbarkeit zwischen den rekonstruierten Szenen und den Ultraschalldaten herstellen zu können, wird eine Darstellungsmethode für die Photogrammetrie Szenen benötigt. Diese werden dafür in einer 2D-Umgebungskarte aus der Vogelperspektive visualisiert, die sich aufgrund der bestehenden 2D-Darstellung der Ultraschalldaten gut für den Vergleich eignet.

Ziel dieser Arbeit ist die Entwicklung eines Verfahrens zur Kartierung der mittels Photogrammetrie rekonstruierter Punktewolken. Da die generierten Punktewolken eine sehr unterschiedliche Qualität besitzen (vgl. Abbildung 2.4), wird zusätzlich zu

der Visualisierung der Hindernisse, eine Filterung der Karte benötigt. Hier werden irrelevante Bereiche herausgefiltert und die Darstellung verbessert. Die Karte wird als *proof of concept* in der Sprache Python realisiert, um sie im Anwendungsfall zu testen. Mit einem LiDAR Referenzfahrzeug wird die Abbildungstreue dieses Verfahrens qualitativ bewertet.

Das Kapitel 2 beinhaltet einen Überblick über ausgewählte Verfahren und Methoden für die Erstellung einer Referenzkarte. Es wird dafür der Bezug zu dieser Arbeit hergestellt und Anwendungsfälle im Kontext der Arbeit erläutert.

In Kapitel 3 werden die entwickelten Verfahren und angewandten Methoden untersucht und verglichen. Die Realisierung des Verfahrens in der Programmiersprache Python ist in Kapitel 4 beschrieben. Auf eine qualitative Validierung wird in Abschnitt 4.3 eingegangen. Abschließend werden die Ergebnisse in Kapitel 5 zusammengefasst und kritisch bezüglich ihres Nutzens bewertet. Darüber hinaus wird ein Ausblick auf Anwendungsfälle und Erweiterungen gegeben.

2. Stand der Technik

In diesem Kapitel werden ausgewählte Verfahren und Algorithmen behandelt, die zur Erstellung der Referenzkarte benötigt werden. Dazu werden die mathematischen Methoden und Modelle im Kontext der Arbeit erklärt. Darauf aufbauend werden in Kapitel 3 die einzelnen Bestandteile des entwickelten Verfahrens genauer vorgestellt.

2.1. Photogrammetrie

Die Photogrammetrie ist eine Bildmesstechnik, mit der aus zweidimensionalen (2D) Aufnahmen eines Objektes oder einer Szene Informationen gewonnen werden können. Kombiniert mit der Disziplin des maschinellen Sehens (engl.: *Machine Vision* oder *Computer Vision*) wird die Photogrammetrie unter anderem dafür genutzt, aus Aufnahmen einer Szene eine dreidimensionale (3D) Repräsentation zu erstellen [5].

2.1.1. Lochkameramodell

Abbildung 2.1 zeigt die Projektion eines Prismas in die Bildebene der Kamera mithilfe des Lochkameramodells. Dieses Modell basiert auf der Zentralprojektion. Der Punkt $\vec{X}_{w,P}$ der Szene werden als Strahl durch den Hauptpunkt F_c in die Bildebene projiziert. Die Pose der Kamera wird als äußere Orientierung (auch extrinsische Parameter) bezeichnet und bezieht sich auf das Szenen-Koordinatensystem $\vec{X}_w$. Mit der Transformation

$$\vec{X}_{c,P} = \mathbf{R}|\mathbf{t}\,\vec{X}_{W,P} \tag{2.1}$$

$$\vec{X}_c = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ WY \\ Z \\ 1 \end{bmatrix}$$

werden die Koordinaten der Szene $\vec{X}_W$ in Kamerakoordinaten $\vec{X}_c$ transformiert.

Die Bildebene der Kamera ist entlang der positiven z-Achse um die Brennweite f verschoben. Dies kompensiert die Spiegelung des Objekts in der Bildebene. Der Bildhauptpunkt (c_x, c_y) liegt in der Mitte des Bildes im Schnittpunkt mit der z-Achse. Die

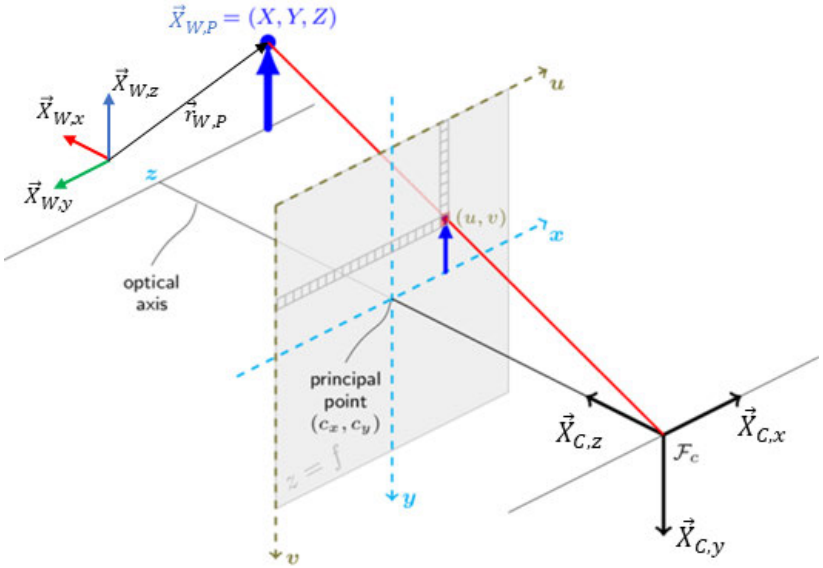


Abbildung 2.1.: Projektion mithilfe des Lochkameramodells. Projektion eines Punktes $\vec{X}_{W,P}$ vom Weltkoordinatensystem $\vec{X}_w$ durch den Hauptpunkt F_C in das Kamerakoordinaten $\vec{X}_c$ [6].

Kameraparameter lassen sich in der Kameramatrix $\mathbf{K}$ zusammenfassen und werden als innere Orientierung (auch intrinsische Parameter) der Kamera bezeichnet.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} W$$

Durch das Zusammenfassen von der Kameramatrix $\mathbf{K}$ und der Transformation $\mathbf{R}|\mathbf{t}$ erhält man die Projektionsmatrix

$$\mathbf{P} = \mathbf{K}\,\mathbf{R}|\mathbf{t} \tag{2.2}$$

Die Abbildung von 3D Szenen Punkten in 2D Bildkoordinaten wird in die Projektionsgleichung

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P}\,\vec{X}_W \tag{2.3}$$

zusammengefasst. Der Parameter s gibt die Skalierung der Projektion an und berücksichtigt damit die Einheit der Szene.

Das Lochkameramodell ist stark vereinfacht im Vergleich zu einer realen Kamera. Reale Linsen erzeugen eine nicht zu vernachlässigende Verzerrung der einfallenden

Lichtstrahlen. Sie verzerren das Bild in der Bildebene hauptsächlich radial [6]. Diese Verzerrung wird durch Koeffizienten in der Kameramatrix **K** berücksichtigt. Diese Koeffizienten finden in dieser Arbeit jedoch keine weitere Anwendung, da sie von dem *Alice Vision Framework* (vgl. Unterabschnitt 2.1.3) berücksichtigt.

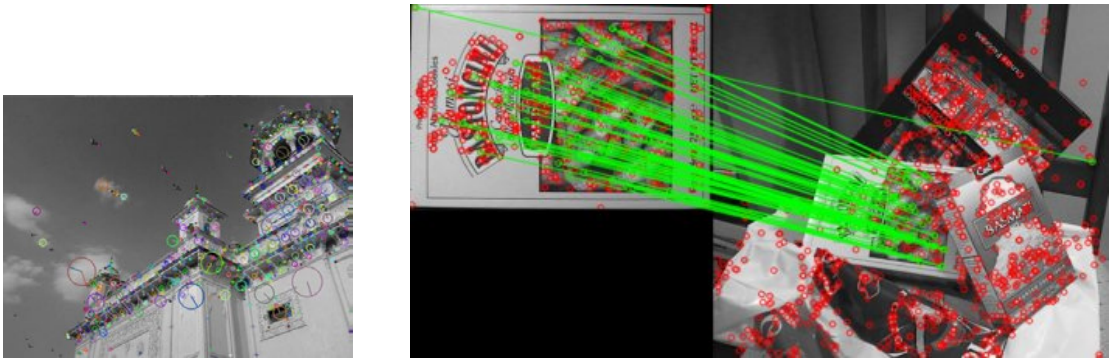
2.1.2. Rekonstruktion

Die Rekonstruktion von Punkten einer Szene wird mithilfe von vier Verfahrensschritten erreicht. Ziel ist es die Projektionsmatrix (Gl. (2.2)), die die Kamerapositionen enthält, aus den Aufnahmen der Szene zu ermitteln. Anhand dieser Projektionsmatrix können dann, Punkte der Aufnahmen in die 3D-Weltkoordinaten projiziert werden.

Die Rekonstruktion erfolgt in den folgendeW vier Schritten:

- Feature-Erkennung
- Image-Matching
- Feature-Matching
- Structure-From-Motion

Um die Bilder einander zuordnen zu können und darin Korrespondenzen festzustellen, werden mit der Feature-Erkennung markante Bereiche (*Features*) in jeder Aufnahme lokalisiert. Diese Bereiche sind unabhängig von Rotation und Skalierung und können damit einfach in Aufnahmen aus einer anderen Perspektive wiedererkannt werden (vgl. Abbildung 2.2a). Die *Features* werden anhand von lokalen Bildmerkmalen in einem 128-dimensionalen Feature-Vektor gespeichert. Dieser Feature-Vektor enthält Informationen über die lokalen Kanten und Farben im Bild [7].



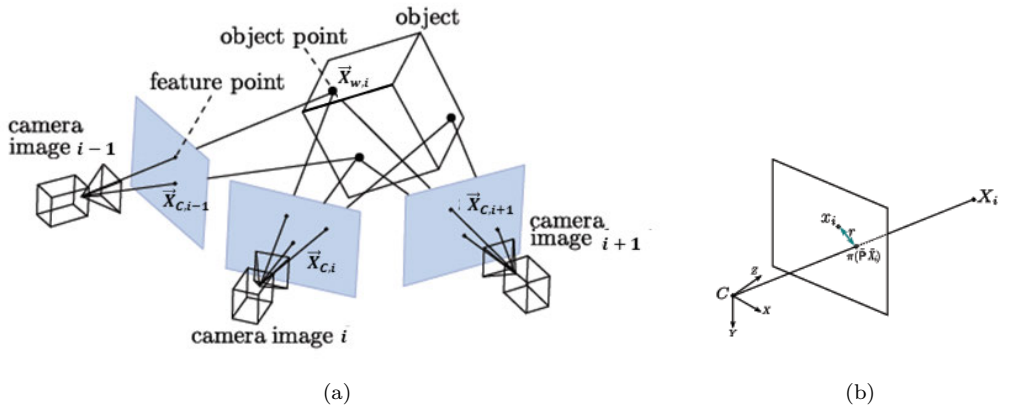
(a) (b)

Abbildung 2.2.: Lokalisierte Features in einer Aufnahme. (a) Feature-Erkennung nach Lowe [7] für eine schwarzweiß Aufnahme [8]. (b) Feature-Matching. Grüne Linien verbinden die einzelnen Feature-Korrespondenzen [9].

Als Vorbereitung für die Korrespondenzanalyse der Features, wird das Image-Matching ausgeführt. Ziel ist es, Aufnahmen von Objekten oder Bereichen in der Szene aus mehreren Perspektiven zu erhalten, sodass Feature-Korrespondenzen nicht global in der gesamten Szene gesucht werden müssen. Um die Aufnahmen effizient vergleichen zu können, wird für jedes Bild ein beschreibendes Merkmal (engl.: *descriptor*) gebildet. Dazu werden die Features jeder Aufnahme in einer Baumstruktur, dessen Blatt Indizes den *descriptor* bilden, gespeichert. Anhand der Übereinstimmung der beschreibenden Merkmale werden die Aufnahmen gruppiert, sodass innerhalb einer Gruppe Aufnahmen mit ähnlichen Features enthalten sind [10].

Die Korrespondenzanalyse der einzelnen Features wird durch das Feature-Matching ausgeführt. Für das Feature-Matching wird die Baumstruktur des Image-Matching Schritts verwendet. Anhand der *descriptors* werden Bildkandidaten für die Korrespondenzanalyse der einzelnen Features gesucht. Die Bildkandidaten lassen sich in der Baumstruktur effizient durch eine Suche ihrer nächsten Nachbarn durchführen. Die Auswahl von Bildkandidaten ist auf maximal zwei begrenzt, um bei der folgenden Feature-Korrespondenz Rechenzeit zu sparen.

Da der Rechenaufwand für die Suche der einzelnen Feature-Korrespondenzen zu hoch ist, wird eine Näherungsmethode verwendet. Die *Fast Approximate Nearest Neighbors Search* sucht diese Feature-Korrespondenzen effizient auf Basis der nächsten Nachbarn. Es werden somit gleiche Punkte in Aufnahmen aus zwei Perspektiven gefunden (vgl. Abbildung 2.3a). Anhand dieser Feature-Korrespondenzen wird die Projektionsmatrix $\mathbf{P}(\vec{X}_{W,i})$ für jedes Bildpaar geschätzt [11].



(a) (b)

Abbildung 2.3.: Rekonstruktion von 3D Punkten. (a) Objektpunkt $\vec{X}_{W,i}$ im Weltkoordinatensystem wird aus mehren Perspektiven in Bildpunkte $\vec{X}_{C,i}$ projiziert. (b) Definition des Projektionsresiduum r für eine Projektion des Objektpunktes X_i in Bildpunkte x_i (2.3).

Im letzten Schritt, der Structure-From-Motion, wird die für jedes Bildpaar geschätzte Projektionsmatrix global optimiert. Hierfür wird für alle Features der Szene eine

Bündelblockausgleichung (engl.: *bundle adjustment*) vorgenommen. Diese Optimierung minimiert den Projektionsfehler (Gl. (2.4)) inkrementell, indem es zu einem initialen Bildpaar Weitere hinzufügt [12].

$$min \sum_{i=1}^{n} \vec{X}_{c,i} - \mathbf{P}(\vec{X}_{w,i}) \tag{2.4}$$

Die Projektion aller extrahierten Features mithilfe der optimierten Projektionsmatrix $\mathbf{P}$ in das 3D Weltkoordinatensystem, ergibt die rekonstruierte Punktewolke der Szene in Abbildung 2.4.

Abbildung 2.4.: Rekonstruierte Punkte einer Szene. Aus 104 Aufnahmen rekonstruierte Parkplatz Szene (vgl. Tabelle A.2 in Anhang A) (93466 Punkte).

2.1.3. Alice Vision Framework

Das Open-Source-Projekt *Alice Vision* [1] implementiert das vorgestellte Verfahren (vgl. Abschnitt 2.1) zur Rekonstruktion von Szenen. Die einzelnen Verfahrensschritte sind in einer Pipeline zusammengefasst und werden von der Anwendung *meshroom* [2] aufgerufen.

Alice Vision ist in der Sprache C++ entwickelt, die Anwendung *meshroom* ist jedoch vollständig in Python 3 [3] implementiert. Die Kommunikation erfolgt über die Kommandozeile und kann mithilfe von Parametern verändert werden. *meshroom* eignet sich daher gut für schnelle Anpassungen und Erweiterungen [12], [13].

[1] `https://alicevision.org/`
[2] `https://github.com/alicevision/meshroom`
[3] `https://www.python.org/`

In Vorarbeiten wurden Ein- und Ausgabefunktionen für die Photogrammetrie Pipeline entwickelt, die es ermöglichen, aus Videodaten Szenen zu rekonstruieren.

2.2. Random-Sample-Consensus-Verfahren

Beim Schätzen von Parametermodellen in großen Datensätzen ist es besonders wichtig, ein robustes Verfahren zu haben, da die Datensätze Ausreißer (engl.: *Outlier*) enthalten können. Die Methode der kleinsten Quadrate eignet sich in solchen Fällen nicht, da ihr Ergebnis stark durch Ausreißer beeinflusst wird [14].

Das Random-Sample-Consensus (RANSAC) Verfahren ist ein probabilistischer, modellbasierter Suchalgorithmus, mit dem Daten effizient nach einem Modell durchsucht werden können. Die Daten werden dafür in zufällig ausgewählte Teilmengen eingeteilt, für die die Modellparameter ermittelt werden. Es werden dann die Punkte ermittelt die innerhalb einer festgelegten Fehlertoleranz liegen (eng.: *Inlier*). Da die Anzahl der Iterationen begrenzt ist, können große globale Fehler verbleiben. Der Algorithmus hat für große Eingangsdaten eine konstante zeitliche Komplexität und wird für Daten bevorzugt verwendet [5].

Es ist vorausgesetzt, dass die Anzahl der minimalen Punkte des Modells, die maximale Anzahl der Iterationen und die Methode der Fehlerberechnung bekannt sind. Der Algorithmus führt folgende Operationen aus:

1. zufällige Teilmenge S der Daten D wird ausgewählt (vgl. Abb. 2.5a)
2. Modellparameter werden für S ermittelt (vgl. Abb. 2.5b)
3. globaler Fehler ϵ zu den Daten D wird ermittelt (vgl. Abb. 2.5c)
4. Iteration von 1. - 3. bis Fehlermaximum unterschritten oder maximale Iterationen erreicht

Im Bereich *Computer Vision* werden mit dem Algorithmus zum Beispiel Geometrien in 2D oder 3D Punktewolken gesucht. Im Rahmen dieser Arbeit konzentriert sich die Anwendung auf das Schätzen von Ebenen in 3D Punktewolken. Hierfür wird das mathematische Modell der Ebene in der Koordinatenform (Gl. (2.5)) verwendet.

$$ax + by + cz + d = 0 \tag{2.5}$$

Die Modellparameter a, b, c, d werden für eine Punktewolke ermittelt, in der die Ebene gesucht wird. Als Eingabeparameter werden die minimale Punkteanzahl der

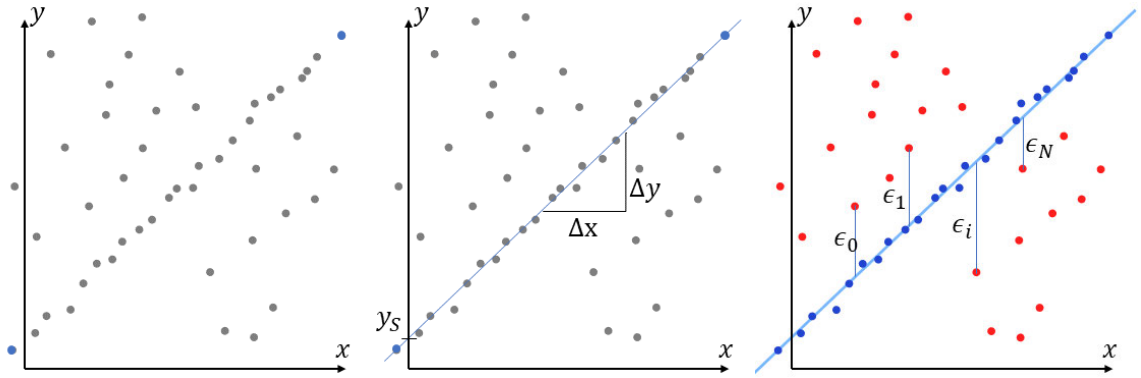


Abbildung 2.5.: Operationen des RANSAC-Algrithmus am Beispiel. Suchen einer Geraden in 2D-Punkten [15]. (a) minimale Punkte für Modellparameter werden ausgewählt (blau).(b) Ermittlung der Modellparameter $\Delta y/\Delta x$ und y_S. (c) Anhand des globalen Fehlers werden *Inlier* ermittelt.

Ebene, der maximale Abstand der Punkte und die maximale Anzahl der Iterationen benötigt.

Die verwendete Methode, mit denen der globale Fehler der Punkte bestimmt wird, kann je nach Implementierung variieren. In dieser Arbeit wird die in der Bibliothek *Open3D* [16] implementierte Methode der kleinsten Quadrate verwendet [17].

2.3. Region-Growing-Segmentierung

Die Segmentierung von Daten zielt darauf ab Regionen mit ähnlichen Eigenschaften in den Daten zu finden. Berücksichtigte Eigenschaften der Daten können zum Beispiel semantischer Art oder quantitative Merkmale von Punkten und ihrer Umgebung sein. Jeder Punkt wird dabei mit einem Label versehen, welches ihn zu einer nicht überlappenden Region zuordnet.

Der Region-Growing-Algorithmus ist eine Methode, um zusammenhängende Regionen zu finden, die definierte ähnliche Eigenschaften haben. Es wird mit einer kleinen Menge zufällig verteilter Punkte begonnen, die dann so lange um ihre Nachbarn erweitert wird, bis keiner mehr der geforderten Definition der Region entspricht. Dann wird eine neue Region anhand eines initialen Punktes erzeugt. Dieser Ablauf wird wiederholt bis alle Punkte einer Region zugeordnet sind [18].

Die von Rabbani et al. [19] vorgestellte Implementierung des Region-Growing-Algorithmus, berücksichtigt die Winkel zwischen den Punkteumgebungsnormalen und deren Residuum. Die Punktenormalen werden durch Ausgleichen des Fehlers einer

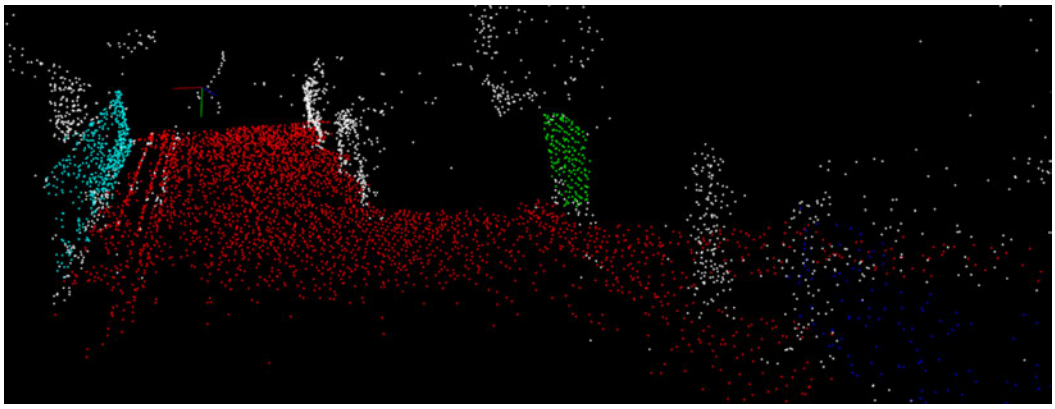


Abbildung 2.6.: Segmentierte Regionen mit Region Growing Algorithmus. Farblich gekennzeichnete Regionen einer Punktewolke mit gleichen Ebenheitseigenschaften und mindestens 50 Punkten. Punkte, die nicht einer Region zugeordnet sind, sind weiß. Die Bodenebene ist Rot eingefärbt.

Ebene in der Punktenachbarschaft ermittelt. Das Residuum der Normale wird durch die Summe der Punkteabstände zur gefitteten Ebene ermittelt.

Die vorgestellte Bedingung kann als Gleichmäßigkeit der Oberfläche interpretiert werden (vgl. Abbildung 2.6). Die Regionen werden so lange vergrößert, bis das Residuum und der Winkel der Normalenvektoren nicht mehr im definierten Grenzwert liegen.

Das Residuum und die Normalenvektoren der Punkte werden vorab berechnet und werden als Eingabeparameter benötigt. Außerdem sind das maximale Residuum der Punktenormalen und der maximale Winkel zwischen den Punktenormalen vorher festzulegen. Da keine Implementierung zur Verfügung steht, die in der Programmiersprache Python verwendbar ist, wurde der Pseudo-Code von Rabbani et al. [19] in Python implementiert.

2.4. Kabsch-Algorithmus

In Problemstellungen im Bereich *Computer Vision* sind häufig Transformationen zwischen zwei oder 3D Datenpunkten gesucht. Der Algorithmus, benannt nach Wolfgang Kabsch (1947-2019), findet die Rotation mit dem kleinsten Fehler [4] zwischen zwei Sets $\mathbf{P}$ und $\mathbf{Q}$ mit korrespondierenden Punktepaaren [20].

Die Varianz zweier Matrizen wird durch die Kovarianzmatrix beschrieben. Für die zwei Matrizen $\mathbf{P}$ und $\mathbf{Q}$ ergibt die Kovarianzmatrix $\mathbf{H}$ sich durch die Gleichung

$$\mathbf{H} = \mathbf{P}^{\mathbf{T}}\mathbf{Q} \tag{2.6}$$

[4] Hier: RMSD (*root mean squared deviation*)

Es ist möglich mit dem algebraischen Ansatz in Gleichung (2.7) die Rotation direkt zu bestimmen. Diese Lösung ist jedoch nicht numerisch stabil, da sie beispielsweise die Inverse der Matrix **H** enthält.

$$\mathbf{R} = (\mathbf{H^T H})^{\frac{1}{2}} \mathbf{H^{-1}} \tag{2.7}$$

Der Kabsch-Algorithmus umgeht diese numerische Problemstellung mit einem allgemeinen Ansatz für die Rotationsmatrix. Der Algorithmus kann dafür in drei Operationen unterteilt werden:

1. Translation der Punkte **P** und **Q**
2. Berechnung der Kovarianzmatrix **H**
3. Berechnung der optimalen Rotation **R**

Um die Punkte **P** und **Q** bestmöglich auszurichten, muss mit einer Translation der Mittelwert der Datensätze in den Ursprung verschoben werden. Hierfür wird der Mittelwert der Punkte von jedem Punkt abgezogen.

Die Rotation wird mithilfe einer Singuläre-Werte-Zerlegung (engl.: *singular value decomposition*, kurz: SVD) ermittelt, da diese numerisch stabiler ist, als die algebraische Lösung

Die Kovarianzmatrix

$$\mathbf{H} = \mathbf{P^T Q} \tag{2.8}$$

der Datenpunkte ist der Ausgangspunkt für die optimale Rotation **R**. Sie wird mithilfe der SVD in die orthogonalen Matrizen **U**, **V** und die Diagonalmatrix **Σ** zerlegt (vgl. (2.9)). Dabei können die orthogonalen Matrizen **U** und **V** als Drehung der Punkte interpretiert werden. Die Diagonalmatrix Σ verzerrt die Koordinaten.

$$\mathbf{H} = \mathbf{U \Sigma V^T} \tag{2.9}$$

Für die Bestimmung der Rotationsmatrix **R** werden die Orthogonalmatrizen **U** und **V** benötigt. Die Rotationsmatrix ergibt sich durch die Multiplikation mit einer parametrierten Einheitsmatrix.

$$\mathbf{R} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} \mathbf{U^T} \tag{2.10}$$

wobei der Korrekturfaktor d sicherstellt, dass ein rechtshändiges Koordinatensystem zugrunde liegt. Laut *Euler's Rotation Theorem (1776)* ist die Determinante einer nicht verzerrenden Rotationsmatrix gleich 1 und einer verzerrenden Matrix gleich -1.

Ist die Determinante der orthogonalen Matrizen -1, so wird das Vorzeichen mit dem Korrekturfaktor d korrigiert und die durch Gleichung (2.10) erhaltene Rotationsmatrix enthält keine Verzerrungen.

$$d = sign(det(\mathbf{VU^T})) \tag{2.11}$$

Der Algorithmus ist auf n-dimensionale Probleme erweiterbar. In dieser Arbeit wird der Algorithmus ausschließlich für 3D-Problemstellungen im Kontext von 3D-Punktewolken verwendet.

2.5. Occupancy-Grid-Karte

Informationen über Hindernisse in einer Karte festzuhalten, ist entscheidend für die Pfadplanung von Robotern oder Fahrzeugen. Ein Ansatz, um eine statische Umwelt in einer Karte abzubilden, ist eine Occupancy-Grid-Karte. In einem *Occupancy Grid* (auch: *Occupancy Grid Map*) wird die Umgebung des Roboters in Zellen einer festen Größe aufgeteilt. Diese Zellen haben einen Zustandswert, der angibt, ob dieser Bereich der Umwelt frei (weiß) oder belegt (schwarz) ist. Zellen, die noch nicht von den Sensoren erfasst wurden, werden als unbekannt (grau) dargestellt.

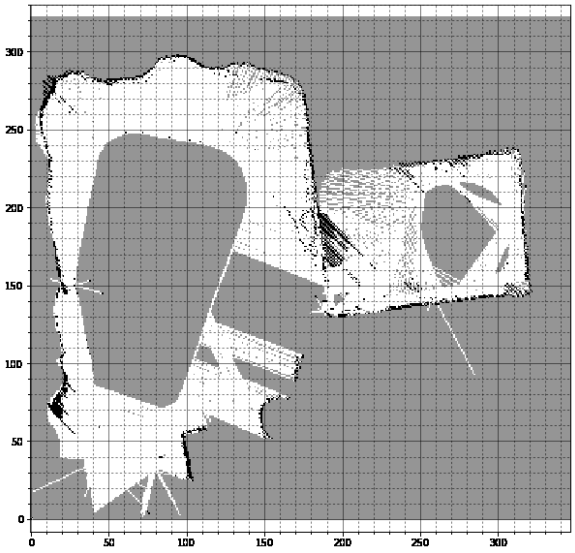


Abbildung 2.7.: Occupancy Grid Map mehrerer Räume. Karte mit stochastischen Zellen in einem 2D-Koordinatensystem, die ein *Occupancy Grid* bilden. Graue Zellen sind unbekannte Bereiche, weiße Zellen sind freie Bereiche und schwarze Zellen sind belegte Bereiche [21].

Da *Occupancy Grids* üblicherweise eine Anzahl von Zellen enthalten, die im Bereich von 10^4 oder mehr besitzen (vgl. Beispiel in Abbildung 2.7), ist es zeitlich nicht

vertretbar, die Zustandswerte aller Zellen zu berechnen. Der Zustandswert stellt daher eine stochastische Größe dar, deren Wert mithilfe der Sensorinformationen geschätzt wird und über mehrere Messzyklen ermittelt wird.

3. Bestandteile des Verfahrens

Dieses Kapitel beschreibt die Konzeptfindung der Verfahrensschritte und zeigt Untersuchungen zu verschiedenen Methoden für die Erstellung einer Referenzkarte. Das Verfahren wurde dafür in die Hauptbestandteile Datenaufbereitung, Bodenschätzung, Höhensegmentierung und Kartierung aufgeteilt, welche einzeln untersucht werden.

3.1. Datenaufbereitung

Die Daten der Szenenrekonstruktion sind aufgrund der Verfahrensungenauigkeit sehr unstrukturiert und inhomogen. Es entstehen viele Ausreißer aus fehlerhaft rekonstruierten Punkten. Andererseits sind Bereiche mit gut strukturierten Texturen mit einer hohen Punktedichte abgebildet.

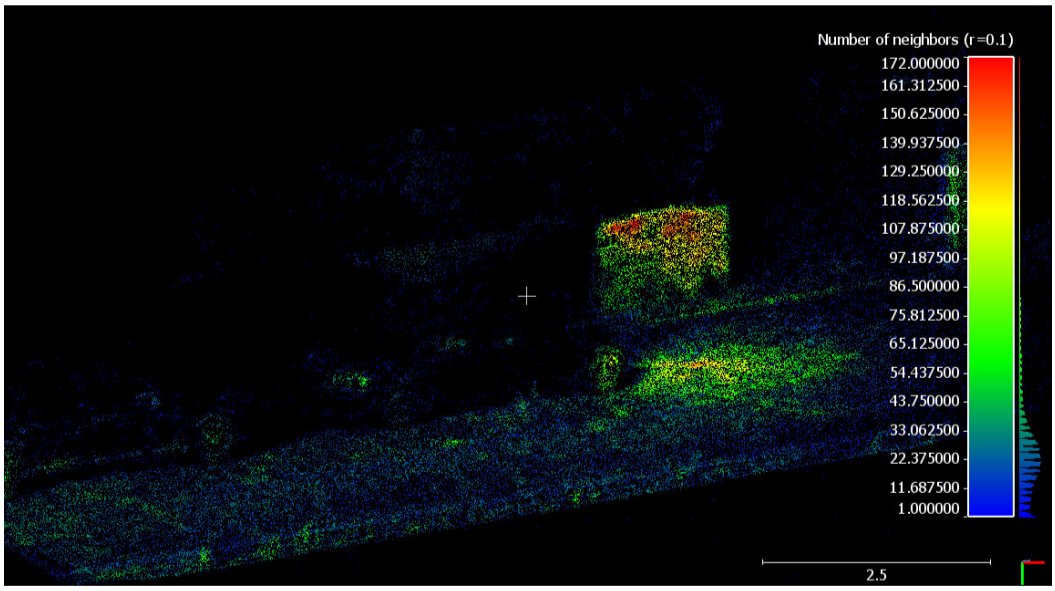


Abbildung 3.1.: Dichteverteilung einer skalierten Punktewolke. Punktewolke eingefärbt anhand der Anzahl der Nachbarn eines Punktes im Radius von 0,1 m. Erstellt mit CloudCompare [22].

In Abbildung 3.1 ist die Punktedichte einer skalierten, aber ungefilterten Punktewolke gezeigt. Die Dichte wurde anhand der Anzahl von Nachbarpunkten in einem Kugelradius

von 10 cm ermittelt. Der Rechenaufwand ist in den Bereichen mit hoher Punktedichte unnötig erhöht, der Informationsgehalt für die Kartendarstellung wird jedoch nicht gesteigert.

Die Punktewolken sind zudem nicht in einer Einheit skaliert, sodass die Daten nicht intuitiv interpretiert werden können. Dies soll mit Skalierung vermieden werden. Im Folgenden wird eine pragmatische Methode zur Skalierung dieser Punktewolken gezeigt und es werden verschiedene Punktefilter verglichen.

Skalierung

Ein erster Ansatz ist, die Punktewolke anhand einer Benutzereingabe zu skalieren. Der Benutzer markiert dafür rekonstruierte Punkte in einer interaktiven Ansicht und gibt die Distanz der Punkte in der gewünschten Einheit ein. Der Skalierungsfaktor kann dann durch die gegebenen Abstände errechnet werden. Ein großer Vorteil ist die einfache Implementierung dieser Methode, da eine interaktive Oberfläche bereits in *Open3D* [16] vorhanden ist.

Ein Nachteil dieses Ansatzes ist, dass die Genauigkeit der Skalierung von der Auflösung der Punktewolke abhängt. Es können nur diskrete Punkte ausgewählt werden. Wenn dies bei der Aufnahme der Szene bereits beachtet wird, sodass die Skalierungspunkte gut rekonstruiert werden können, ist der Einfluss jedoch gering. Die Validierung in Abschnitt 4.3 zeigte, dass dieser Einfluss nicht nennenswert ist, da die Streuung der Punkte aus der Photogrammetrie erheblich größer ist.

Als weiterer Ansatz kann die Punktewolke anhand von Landmarken und bekannten Objekten skaliert werden. Hier werden bekannte Objekte in der Punktewolke lokalisiert und deren Abmaße ermittelt. Dies kann anhand von Merkmalen, wie Ecken oder der größten Abmaße erfolgen. Auch das Skalieren über Marker, wie zum Beispiel QR-Codes ist möglich. Da der Fokus der Zielsetzung dieser Arbeit auf der robusten Bodenschätzung und der Kartierung der Szene liegt, wird von diesem Ansatz abgesehen. Die Problemstellungen würden den Rahmen der Arbeit überschreiten, da die Projektion von Landmarken in die Punktewolken umfangreiche Untersuchungen und Algorithmen erfordern.

Ausreißer-Filterung

Nachdem die Punktewolke skaliert wurde, wird die lokale Qualität der Punktewolke aufbereitet. Fehlerhaft rekonstruierte Punkte liegen oft weit von dem Großteil der

Punkte entfernt und erzeugen in einer späteren Kartendarstellung mehr Rechenaufwand. Diese Punkte werden daher anhand der statistischen Verteilung des Abstandes zueinander herausgefiltert.

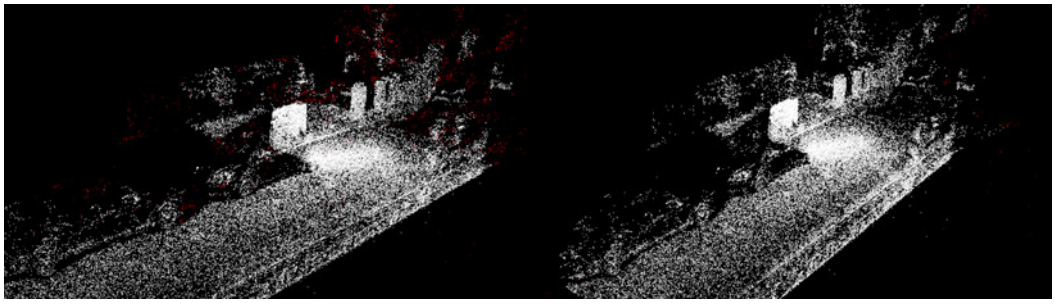
Der *Statistical Outlier Filter* [23] ermittelt Ausreißer anhand von zwei Parametern, dem Faktor k der Standardabweichung und der Anzahl der Nachbarpunkte n_points. Hierfür wird für jeden Punkt die Distanz d zu den n Nachbarpunkten berechnet. Aus dem Mittelwert μ und der Standardabweichung σ der Punktedistanzen des Punkte-Clusters wird dann der Schwellwert

$$T = \mu + k\,\sigma \tag{3.1}$$

bestimmt. Punkte für die $d > T$ gilt, werden entfernt [23].

Der *Radius Outlier Filter* sucht dagegen Nachbarpunkte innerhalb eines festen Suchradius um den aktuellen Punkt. Es muss die minimale Punkteanzahl n und der Suchradius r vorab festgelegt werden. Der Punkt wird entfernt, wenn weniger als die angegebene Anzahl von Nachbarn gefunden wurde.

Der Vorteil des statistischen Filters gegenüber eines Radius Filters ist, dass auch ganze Punkte-Cluster, die weit entfernt von der eigentlichen Punktewolke sind, herausgefiltert werden. Abbildung 3.2 zeigt einen Vergleich der vorgestellten Filter für die Szene Nr. 1 in Tabelle A.1 (vgl. Anhang A). Der Radius Filter entfernt nur kleinere Punktemengen (vgl. Abbildung 3.2a).



(a) (b)

Abbildung 3.2.: Vergleich der Ausreißer Filterung. Ausreißer in Rot eingefärbt, ursprüngliche Punktewolke in Weiß. (a) statistischer Ausreißer Filter mit $n\_points = 100$ und $k = 1$. (b) Radius Filter mit $r = 0,1$ und $n\_points = 1$.

Dichte-Homogenisierung

Die signifikanten Dichteunterschiede in der Punktewolke sind für die folgenden Verfahrensschritte nicht erforderlich, da der Informationsgehalt für die spätere Kartierung

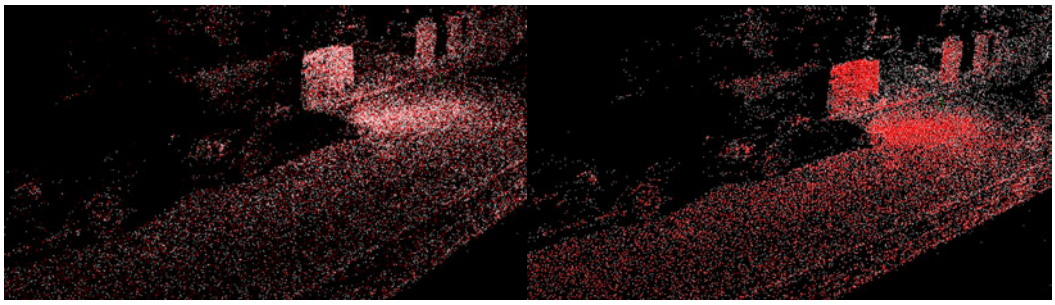
nicht erhöht wird. In Abschnitt 3.2 wird gezeigt, dass mit einer geringeren Punktedichte die Normalen deutlich robuster geschätzt werden können, während Details in den Punktewolken nicht verloren gehen. Durch die verringerte Punkteanzahl wird zusätzlich die Rechenzeit der folgenden Verfahren reduziert.

Um die Dichte zu homogenisieren, wird ein *Voxel Grid Filter* verwendet. Dieser Filter teilt die Begrenzungsbox der Punktewolke (auch *Bounding Box*) in gleich große Quader (auch *Voxel*). Die Kantenlänge dieser *Voxel* ist konstant und als Parameter gegeben. Jeder *Voxel*, der einen oder mehr Punkte enthält, wird mit dem Mittelwert dieser Punkte approximiert. Dies verringert die Auflösung der Punktewolke gleichmäßig [24].

Als alternativer Ansatz wird das *Uniform Downsampling* angewendet. Dieser Filter verringert die Auflösung einer Punktewolke auf jeden k-ten Punkt. Mit der Schrittweite $k = 2$ würde die Anzahl der Punkte somit halbiert werden.

In Abbildung 3.3 ist ein Vergleich der beiden Algorithmen zu sehen. Der *Voxel Grid Filter* (vgl. Abbildung 3.3b) zeigt hier das bessere Verhalten, da er an den dichteren Stellen mehr Punkte filtert, jedoch an den dünn abgebildeten Stellen die Punkte nicht noch weiter ausdünnt. Das *Uniform Downsampling* (vgl. Abbildung 3.3a) erzeugt erwartungsgemäß eine gleichmäßige Ausdünnung der Punktewolke. Daher gehen in dünn abgebildeten Bereichen zu viele Informationen verloren.

Der *Voxel Grid Filter* wird daher für die weiteren Untersuchungen verwendet. In Abschnitt 3.4 wird näher erläutert, wie die Voxel-Größe in Abhängigkeit der erzeugten Karte gewählt wird.



(a) (b)

Abbildung 3.3.: Vergleich Voxel Grid Downsampling und Uniform Downsampling. Reduktion der Punktedichte durch die vorgestellten Filter (rot) entfernte Punkte (weiß) verbleibende Punkte. (a) *Uniform Downsampling* (b) *Voxel Grid Filter*.

3.2. Bodenschätzung

Um die Referenzkarte in der Vogelperspektive darstellen zu können, wird ein Koordinatensystem im Boden benötigt. Da das Koordinatensystem $\mathbf{E}_0$ der rekonstruierten Punktewolken nicht immer gleich definiert ist, muss der Boden robust erkannt werden und es kann keine Annahme bezüglich der Lage im Koordinatensystem getroffen werden. Das Koordinatensystem $\mathbf{E}_0$ wird anschließend, wie in Abbildung 3.4 gezeigt, mit der positiven z-Richtung normal zum Boden transformiert.

Im Folgenden wird eine Bodenhypothese entwickelt und es werden Konzepte für die Bodenschätzung verglichen. Aus dem robustesten Konzept wird der Algorithmus *align_ground* (vgl. Algorithmus 1) abgeleitet und erläutert.

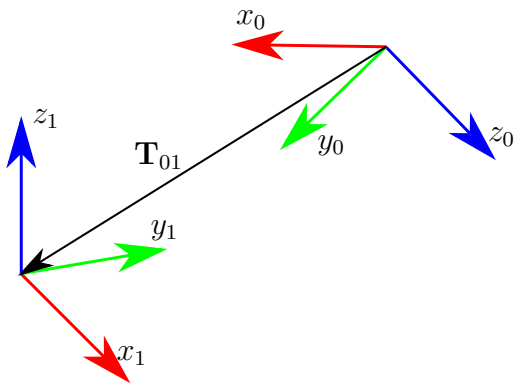


Abbildung 3.4.: Koordinatentransformation der Punktewolke. Transformation des Koordinatensytems $\mathbf{E}_0$ mit x_0, y_0, z_0 in das konstruierte Bodenkoordinatensystem $\mathbf{E}_1$ mit x_1, y_1, z_1.

Das Modell für die Lokalisierung des Bodens (Bodenhypothese), muss den Boden in einer Punktewolke mit großen Ausreißern und undeutlichen Strukturen hinreichend gut beschreiben, um robust gegen störende Strukturen in der Nähe des Bodens zu sein. Da nur Oberflächen von der Rekonstruktion erfasst werden, kann angenommen werden, dass bis auf Ausreißer alle Punkte oberhalb der Bodenebene liegen sollten, ansonsten ist die erkannte Ebene keine Bodenebene. Die Ebene wird mithilfe des RANSAC-Algorithmus geschätzt, da dieser robust gegen Ausreißer ist.

Prüfen lässt sich die Punktebedingung, indem für jeden Punkt, der nicht zu der geschätzten Ebene gehört, der vorzeichenbehaftete Abstand

$$D = \vec{n} \cdot \vec{x} + p \tag{3.2}$$

zur Ebene (vgl. Abbildung 3.5) berechnet wird. Wobei $\vec{n}$ der normierte Normalenvektor

$\vec{N} = (a, b, c)^T$ und

$$p = \frac{d}{|\vec{N}|} \tag{3.3}$$

der Stützpunkt der Koordinatenform einer Ebene sind [25, S. 541].

Gellert et al. [25, S. 540] leiten Gleichung (3.2) aus der Hesse'schen Normalform der Ebene

$$\vec{n} \cdot \vec{x} = -p \tag{3.4}$$

ab.

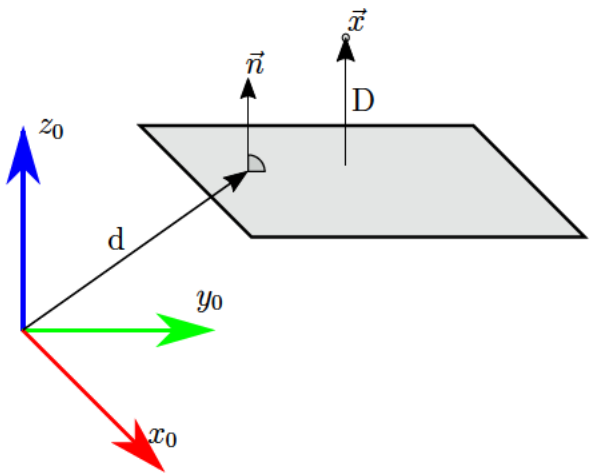


Abbildung 3.5.: Abstand des Punktes $\vec{x}$ zur Bodenebene. Vorzeichenbehafteter Abstand D eines Punktes $\vec{x}$ zu der geschätzten Ebene.

Die Punkte mit dem Abstand $D > 0$ liegen auf der Seite des Normalenvektors $\vec{n}$ der Ebene, Punkte mit dem Abstand $D < 0$ liegen auf der gegenüberliegenden Seite des Normalenvektors $\vec{n}$. Da die Ebene des RANSAC-Algorithmus eine definierte Streuung ϵ hat, werden nur Punkte außerhalb dieser Streuung durch die Bedingung

$$|D| \geq \epsilon$$

berücksichtigt.

Aus den beiden gebildeten Punktemengen wird ein Verhältnis zur gesamten Punktewolke gebildet. Dieses Vorgehen ist im folgenden Abschnitt unter Algorithmus 1 näher erläutert.

Konzepte

Um den Boden mit dem zuvor entwickelten Modell zu lokalisieren, wird zunächst ein sehr einfacher Ansatz gewählt. Die erforderlichen Ebenenparameter für die Bodenhypothese, werden mithilfe des RANSAC-Algorithmus (vgl. Abschnitt 2.2) ermittelt. Störende Strukturen, die im Hintergrund rekonstruiert wurden, haben auf die Ebene keinen Einfluss. Dies ist vorteilhaft, da die Punktewolken der Photogrammetrie oft Hintergrundobjekte oder Ausreißer enthalten. Problematisch ist, dass es viele andere flache Objekte, wie beispielsweise Hauswände, Dächer oder Bürgersteige, in Punktewolken gibt. Je nach Ausprägung, ist die Wahrscheinlichkeit größer, diese Strukturen zu finden. Robust funktioniert dieser Ansatz in Szene Nr. 1 der Tabelle A.1 (s. Anhang A), da der Untergrund als eine große, flache Ebene ausgeprägt ist.

Als alternativer Ansatz zum RANSAC Verfahren, wird die Punktewolke mit dem Region-Growing-Verfahren segmentiert. Mit der von Rabbani et al. [19] vorgestellten Ebenheitsbedingung (vgl. Abschnitt 2.3) werden zusammenhängende und ebene Regionen gefunden. Da der Region-Growing-Algorithmus alle Punkte der Punktewolke in Regionen unterteilt, werden mit der Segmentierung möglicherweise mehrere Kandidaten für eine Bodenebene gefunden. Alle Regionen, die größer als die Mindestgröße *min_size* sind, werden ausgegeben und es wird mithilfe des RANSAC Algorithmus *estimate_ground* eine Ebene in dem Kandidaten-Cluster geschätzt. Diese Kandidaten werden dann einzeln mit der Bodenhypothese verglichen und bewertet.

Die Bewertung wird anhand der Punkteanzahl auf der Seite des Normalenvektors und der Punkteanzahl des Kandidaten-Clusters vorgenommen. Beide Größen werden auf die Gesamtanzahl der Punkte normiert und zu einem *Score* (*weighted_score*) verrechnet. Der Algorithmus 1 zeigt den Ablauf der Bewertung. Jeder Normalenvektor der Kandidaten-Cluster wird in der Richtung umgekehrt, um alle Möglichkeiten der Ebenenorientierung zu bewerten.

Koordinatentransformation

Wenn der Boden erfolgreich lokalisiert ist, sind dessen Modellparameter a, b, c, d bekannt. Die Ebene des Bodens kann somit durch ihren Normalenvektor $\vec{N}$ und den Stützpunkt d beschrieben werden (vgl. Abbildung 3.5). Das Koordinatensystem ($\mathbf{E_0}$) der Punktewolke soll mit der in Algorithmus 1 verwendeten Funktion $find\_ground$ so transformiert werden, dass die x-y Ebene in der Bodenebene liegt und die z-Achse senkrecht nach oben zeigt. Die Richtung der x-Achse und der y-Achse wird zufällig gewählt.

Algorithm 1 Bodenausrichtung

```
1: procedure ALIGN_GROUND(point_cloud)
2:     plane_candidates ← {}
3:
4:     (normals, residuals) ← estimate_normals(point_cloud, knn)
5:     regions ← segment_by_region(point_cloud, normals, residuals, min_size)
6:
7:     for region in regions do
8:         [a, b, c, d] ← estimate_ground(region, plane_eps)
9:
10:        plane_ratio ← get_cluster_ratio(point_cloud, [a, b, c, d])
11:        weighted_score ← 2 · sizeof(region)/sizeof(point_cloud) + plane_ratio
12:        plane_candidates ← (weighted_score, region)
13:
14:        plane_ratio ← get_cluster_ratio(point_cloud, −1 · [a, b, c, d])
15:        weighted_score ← 2 · sizeof(region)/sizeof(point_cloud) + plane_ratio
16:        plane_candidates ← (weighted_score, region)
17:
18:    end for
19:
20:    plane ← max_score(plane_candidates)
21:    T ← find_ground_plane(plane, [a, b, c, d])
22: return transform(point_cloud, T)
23: end procedure
```

Um die gewünschte Ausrichtung zu erreichen, muss das Koordinatensystem $\mathbf{E}_1$ in der Ebene konstruiert werden. Dazu werden zwei Punkte aus der Ebene der Punktewolke gewählt. Damit diese für gleiche Boden-Cluster ebenfalls deterministisch sind, wird der betragsmäßig kleinste Ortsvektor $\vec{r}_{min}$ und größte Ortsvektor $\vec{r}_{max}$ gewählt. Anhand dieser zwei Punkte wird der Verschiebungsvektor

$$\vec{r}_{mm} = \vec{r}_{max} - \vec{r}_{min} \tag{3.5}$$

gebildet. Dieser Vektor dient als Referenz für das Koordinatensystem $\mathbf{E}_1$ in der Ebene. Aus dem Vektor $\vec{r}_{mm}$ und dem Normalenvektor $\vec{n}$ wird dann der Vektor $\vec{e}_{1y}$ gebildet, der die neue y-Achse der Koordinatenbasis darstellt.

$$\vec{e}_{1y} = \vec{r}_{mm} \times \vec{n} \tag{3.6}$$

Der in Gleichung (3.6) gebildete Vektor steht senkrecht auf dem Normalenvektor

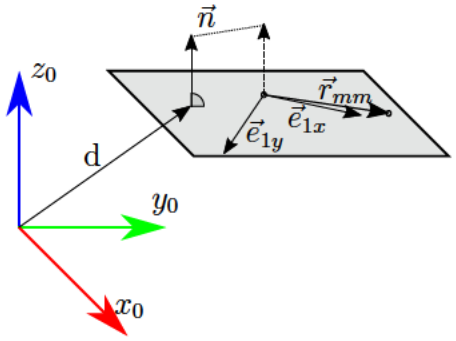


Abbildung 3.6.: Konstruktion des Bodenkoordinatensystems. Bodenkoordinatensystem wird durch Vektoren der Koordinatenbasis $\mathbf{E}_1 = [\vec{e}_{1x}, \vec{e}_{1y}, \vec{n}]$ definiert.

und dem Ortsvektor. Da der Ortsvektor anhand von diskreten Punkten in der Ebene gebildet wurde, liegt dieser nicht genau in der Ebene.

$$\vec{e}_{1x} = \vec{e}_{1y} \times \vec{n} \tag{3.7}$$

Es kann somit die Koordinatenbasis

$$\mathbf{E}_1 = \left[\begin{array}{c|c|c} \vec{e}_{1x} & \vec{e}_{1y} & \vec{n} \end{array}\right] \tag{3.8}$$

konstruiert werden. Sie dient als Referenz für die Koordinatentransformation der Punktewolke.

Um die Punktewolke transformieren zu können, wird die Rotation des Ursprungs zur Koordinatenbasis des Bodens (Gl. (3.8)) gesucht. Eine optimale Rotationsmatrix $\mathbf{R}_{01}$ zwischen Vektoren kann mit dem Kabsch-Algorithmus (vgl. Abschnitt 2.4) effizient gefunden werden [20].
Zuletzt wird der Ortsvektor der Basis $\mathbf{E_1}$ in der Bodenebene gesucht. Hier kann der erste Punkt $\vec{r}_{max}$ verwendet werden, um die Punktewolke zu transformieren.
Mit der Transformationsmatrix

$$\mathbf{T}_{01} = \begin{bmatrix} \mathbf{R}_{01} & \vec{r}_{max} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.9}$$

werden die Punkte der rekonstruierten Punktewolke starr transformiert.

Abbildung 3.7 zeigt eine erfolgreiche Ausrichtung der Straßenszene in Anhang A (vgl. Tabelle A.1 Nr. 1). Dieser Ansatz erweist sich als robust und effizient, da er nicht nur

wenige Operationen benötigt, sondern diese auch effizient durch Matrixmultiplikationen durchführbar ist. Die Genauigkeit ist abhängig von der zulässigen Streuung ϵ der Ebene. Da jedoch die geschätzten Ebenen in den Regionen des Region-Growing-Algorithmus sehr genau [1] sein können, ist der Einfluss nicht nennenswert.

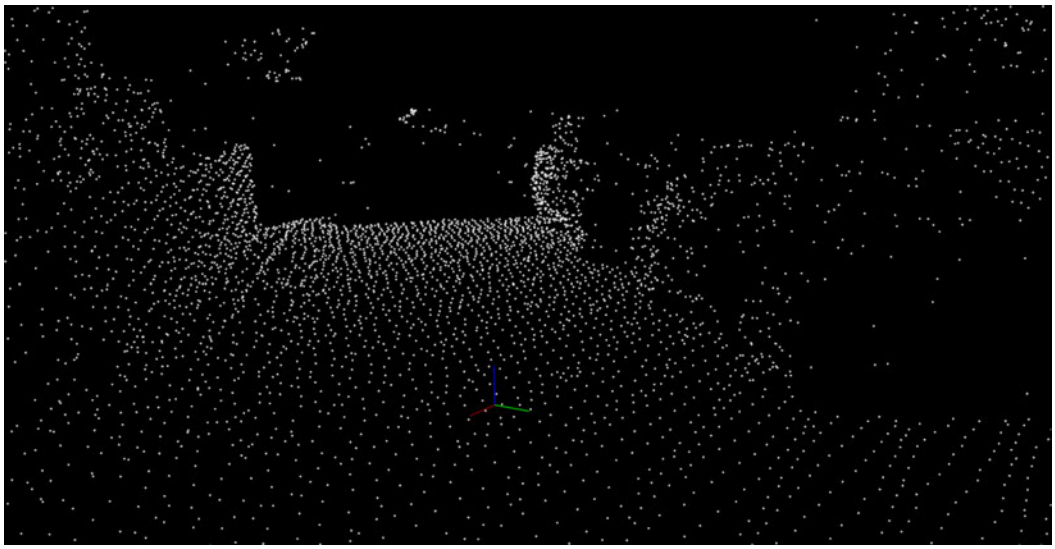
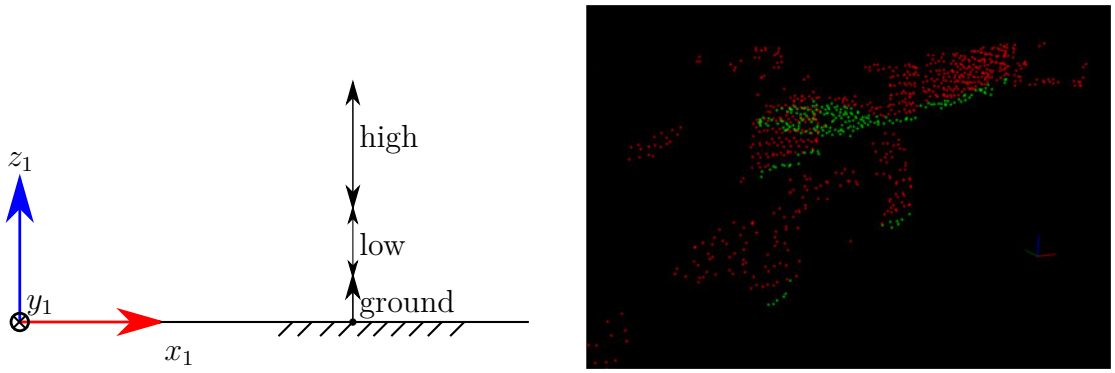


Abbildung 3.7.: Am Boden ausgerichtetes Koordinatensystem. Koordinatensystem der Strapenszene Nr. 1 in Tabelle A.1 (vgl. Anhang A) am Bodenboden ausgerichtet.

3.3. Höhensegmentierung

Die in Abschnitt 3.2 vorgestellte Transformation ermöglicht die Interpretation der terrestrischen Szenen. Die z-Koordinate des neuen Koordinatensystems kennzeichnet die Höhe der Punkte über dem Boden. Es wird eine genauere Filterung der Punkte mit der Berücksichtigung der Fahrzeuggeometrie ermöglicht. Im Folgenden ein Konzept zur Bestimmung von hohen und niedrigen Bereichen genauer betrachtet.

[1] 2 cm Streuung für $\epsilon = 0,01$



(a) (b)

Abbildung 3.8.: Höhensegmentierung anhand der z-Koordinate. (a) schmematische Höhenbereiche in der Punktewolke (b) Punktewolke (Szene Nr. 1 in Tabelle A.1 Anhang A) eingefärbt nach ihrer Höhe.

Für den Ansatz soll pro Punkt allein die z-Koordinate für die Höhenbestimmung verwendet werden. Wenn der Boden hinreichend genau geschätzt wurde, ist es möglich die irrelevanten Bereiche, anhand ihrer Höhe zu eliminieren. Dafür wird die Punktewolke in einen hohen und einen niedrigen Bereich aufgeteilt. In Abbildung 3.8a ist schematisch dargestellt, in welche Bereiche die Punktewolke aufgeteilt wird. Das Ergebnis einer gefilterten Punktewolke mit eingefärbten Höhen ist in Abbildung 3.8b gezeigt. Da die Bodenschätzung in Abschnitt 3.2 sehr robust ist, hat sich diese Methode der Höhenbestimmung als hinreichend genau herausgestellt. Außerdem sind die Bereiche einfach auf verschiedene Fahrzeugtypen anpassbar.

3.4. Kartierung

Die Punktewolke wurde in den vorangegangenen Verfahrensschritten so aufbereitet, dass sie wenig Ausreißer enthält und den für das Park- und Manövrierassistenzsystem relevanten Höhenbereich abbildet. Da die Szene im Bodenkoordinatensystem $\mathbf{E_1}$ beschrieben ist, kann eine Kartendarstellung in zweidimensionalen (2D) Koordinaten über eine Projektion der dreidimensionalen (3D) Punkte erfolgen.

Die im Folgende verglichenen Karten haben zur Vereinfachung ihren Ursprung ebenfalls im Bodenkoordinatensystem $\mathbf{E_1}$.

Geometrieprojektion

Um die 3D-Punktewolke in einer Karte abzubilden, muss diese um eine Dimension reduziert werden. Da die Karte im vorigen Verfahrensschritt bereits auf den wesentlichen Höhenbereich begrenzt wurde, können die Punkte senkrecht in die Bodenebene projiziert

werden. Damit lässt sich jeder Punkt der 3D-Punktewolke nur noch mit seiner x und y Koordinate in der Karte darstellen.

Ein Nachteil dieser Methode ist, dass Informationen zu überhängenden Hindernissen verloren gehen. Da diese jedoch selten im Bereich des Sensorsichtfeldes sind, ist dies eine vertretbare Einschränkung.

Kartendarstellung

Die 2D-Darstellung der Punkte soll im Folgenden so visualisiert werden, dass die Höheninformation klar erkennbar ist und die Geometrie von Objekten interpretiert werden kann. Ziel ist es, mit der Interpretation der Geometrie die Ultraschalldaten plausibilisieren zu können.

Hierfür werden zwei Konzepte der Punktedarstellung näher erläutert. Zuerst wird auf die Darstellung anhand einer 2D-Datenpunktekarte eingegangen. Außerdem wird eine 2.5D-Karte in Anlehnung an das *Occupancy Grid Mapping* von Moravec und Elfes [26] gezeigt.

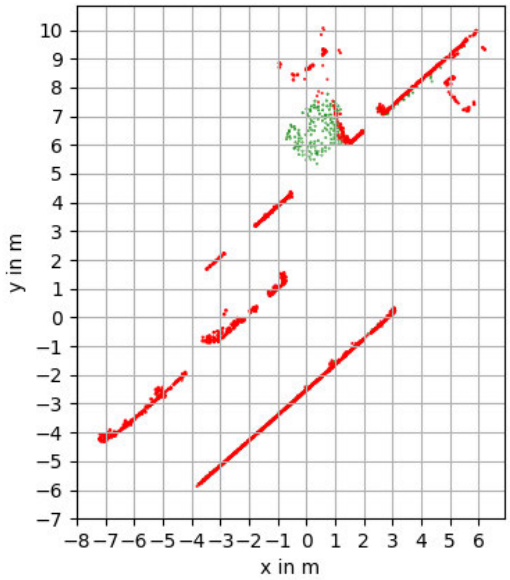


Abbildung 3.9.: Umweltrepräsentation als 2D-Punktekarte. Punktekarte mit hohen (rot) und niedrigen (grün) Punkten der projizierten Punktewolke.

Die in Abbildung 3.9 abgebildete Karte aus einzelnen Punkten stellt jede einzelne Geometrieinformation dar, die vorhanden ist. Punkte die sehr dicht aneinander liegen werden trotzdem separat dargestellt. Diese Darstellung ist nicht optimal für große Punktewolken, da die Punkte zu dicht abgebildet werden. Die Karte wird schnell

unübersichtlich.

Mit dem Konzept der *Occupancy Grid Map* (vgl. Abschnitt 2.5) wird die Punktewolke in Zellen unterteilt (vgl. Abbildung 3.10). Diese Zellen haben eine feste Auflösung und werden, im Gegensatz zu der von Moravec und Elfes [26] vorgestellten Karte, vollständig für die ganze Karte erstellt bevor sie mit ihrem Zustand versehen werden. Die Punkte der Punktewolke werden dann anhand ihrer Koordinaten in die Zellen einsortiert. Dieser Ansatz beansprucht mehr Laufzeit, jedoch ist ein probabilistischer Ansatz, wie in [26], nicht möglich, da die Zellen nicht wiederholt gemessen werden. Es kann somit keine Wahrscheinlichkeit für den Status der Zelle auf Basis der Messungen gebildet werden.

Ein Vorteil der Darstellung der *Occupancy Grid Map* ist, dass Punkte einer Höhe, die dicht beieinander liegen, in einer Zelle gruppiert werden. Es werden außerdem keine Punkte mit verschiedenen Höhen überlagert, da es nur eine Zelle gibt, dessen Status anhand einer Bedingung angepasst wird.

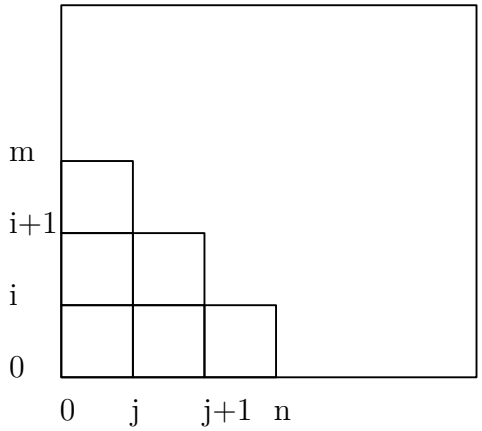


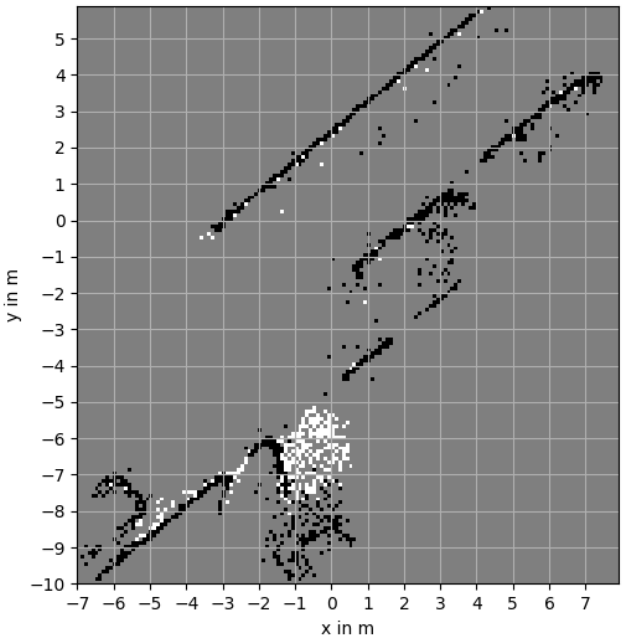
Abbildung 3.10.: Koordinaten der Zellen in der Referenzkarte. Ursprung der Koordinaten und Referenzpunkt der Zellen unten links. Zellen werden von 0 bis (m,n) mit Indizes versehen.

Die Abstraktion der *Occupancy Grid Map* eignet sich sehr gut zum Filtern der Punkte in den Zellen. Beispielsweise können Zellen Informationen ihrer Nachbarn kennen und somit ihre Eigenschaften anpassen. Im folgenden Abschnitt werden Methoden zur Filterung der Karte vorgestellt.

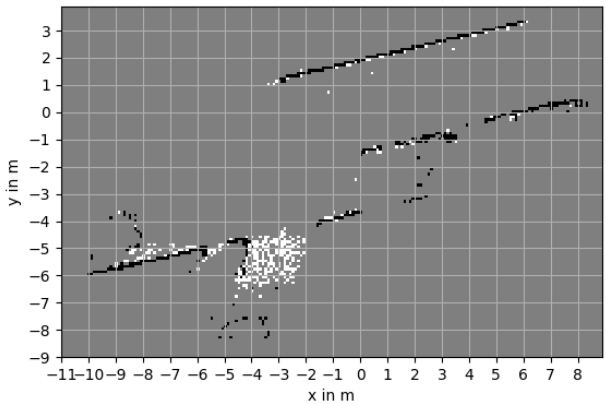
Filterung der Karte

Die in Abbildung 3.11a gezeigte Karte enthält noch vereinzelte Zellen. Diese Zellen werden im Folgenden durch eine Punkteschwelle herausgefiltert. Diese Bedingung setzt

den Status der Zelle erst auf den geforderten Wert, wenn eine definierte Anzahl von Punkten innerhalb der Zelle erreicht wurde. Dies kann für hohe und für niedrige Punkte separat festgelegt werden, da diese Objekte unterschiedlich gut abgebildet sind.



(a)



(b)

Abbildung 3.11.: Vergleich des Zellenfilters der *Occupancy Grid Map* anhand einer Punkteschwelle. (a) Ungefilterte Umgebungskarte der Szene Nr. 1 in Tabelle A.1 (vgl. Anhang A). (b) Durch Punkteschwelle gefilterte Umgebungskarte der Szene Nr. 1 in Tabelle A.1 (vgl. Anhang A).

4. Realisierung

Ziel dieser Arbeit ist eine prototypische Realisierung der Referenzkarte. Dafür werden in diesem Kapitel, die in Kapitel 3 ausgewählten Verfahrensschritte, systematisch implementiert und mit ausgewählten Testszenen validiert.

4.1. Entwicklungsumgebung

Da es keine Hardware-Einschränkungen durch das Verfahrenskonzept gibt, wird die Entwicklung auf einem handelsüblichen Entwicklungslaptop der Firma HP umgesetzt. Dieser Laptop ist mit Microsoft Windows 10 Enterprise (Version 1809, 64 bit) ausgestattet und verfügt über die folgenden Hardware-Spezifikationen:

- Intel Xeon CPU E3-1505M @ 2.8 GHz, x86 Architektur
- Nvidia Quadro GPU M2000M
- 32 GB RAM

Das in Kapitel 3 gezeigte Verfahren ist überwiegend in der Sprache Python 3 (Version 3.6) entwickelt. Da für die Visualisierung auf die Open-Source-Bibliothek *python-pcl* [1] zurückgegriffen wird, kann keine aktuellere Version der Python-Umgebung verwendet werden. Für alle nicht selbst implementierten Funktionen für die Handhabung von Punktewolken, wird die Open-Source-Bibliothek *Open3D* (Version 0.10) [16] verwendet.

Für die von der Robert Bosch GmbH bereitgestellte Messdaten Verarbeitung (vgl. Abbildung 4.3) muss die Python Version 3.8 verwendet werden. Die Anpassung und Integration dieser Komponente ist aufgrund der geringen Verwendung in dieser Arbeit nicht nötig, daher wird sie separat verwendet.

Alle Videodaten werden mit einer Smartphone-Kamera des Herstellers *OnePlus* aufgenommen. Das Modell *OnePlus 6* hat eine Brennweite von 4,25 mm und eine Auflösung von 4608x2176 Pixeln und damit 10 Megapixeln (MP). Die Videoaufnahmen werden mit einer Auflösung von 3840x2160 Pixeln und damit 8,29 MP aufgenommen. Die Bildrate der Kamera beträgt 30 *frames per second* (fps). Alle Angaben werden mit

[1] `https://github.com/strawlab/python-pcl`

dem *ExifTool by Phil Harvey* [2] aus den Video- und Bildmetadaten ausgelesen, da in der Hardware-Spezifikation der zwei Sensoren keine eindeutigen Angaben gemacht werden.

4.2. Implementierung

Um das Verfahren in mehreren Schritten implementieren und diese Schritte separat untersuchen zu können, wurde die Implementierung in die Bestandteile des Verfahrens aufgeteilt. Es ergeben sich daraus die vier Komponenten Datenaufbereitung, Bodenschätzung, Höhensegmentierung und Kartierung (vgl. Abbildung 4.3), die in Abhängigkeit zueinander stehen. Die Kombination der Verfahrensschritte und die Datenein- und ausgabe wurden in dem Skript *Cloud2Grid* implementiert.

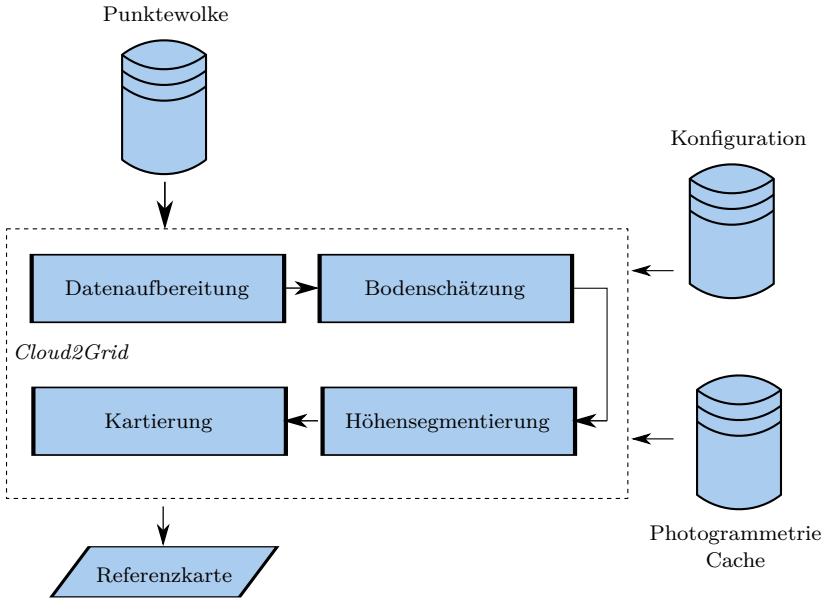


Abbildung 4.1.: Implementierungsschema des entwickelten Verfahrens. Eingabedaten, Ausgabedaten und schematischer Aufbau des Cloud2Grid-Skripts.

Diese Komponenten greifen alle auf eine Konfigurationsdatei zu, in der die Parameter des Verfahrens gespeichert sind.

Ein- und Ausgabe

In dem Skript *Cloud2Grid* erfolgt die Eingabe der Daten und Optionen über eine Kommandozeilen-Schnittstelle. Diese Schnittstelle eignet sich durch ihre Flexibilität und Einfachheit gut für prototypische Anwendungen. Es können schnell Optionen hinzugefügt oder geändert werden.

[2] `https://exiftool.org/`

Tabelle 4.1.: Optionen des Cloud2Grid-Skripts. .

Option	Beschreibung
scale-dist	Räumlicher Abstand, der zu wählenden Skalierungspunkte
aligned	Punktewolke ist bereits ausgerichtet, Verfahrenschritt wird übersprungen
vhm	Punktewolke ist bereits im Fahrzeugkoordinatensystem abgebildet
show-cameras	Kamerastandpunkte in der Referenzkarte anzeigen

Der Aufruf des Cloud2Grid-Skripts erfordert nur die Punktewolke als Parameter. Diese muss als Polygon-Datei (*.ply) oder als Komma-separierte-Werte (*.csv) vorliegen. Statt eines Dateipfades zu der Punktewolke, kann auch ein Pfad zu dem Archiv Ordner der Photogrammterie Pipeline eingegeben werden. Die Ausgabe der Kamerastandpunkte mithilfe der von *meshroom* zwischengespeicherten Daten, wird dann möglich. Neben der Punktewolke können die in Tabelle 4.1 aufgeführten Optionen angegeben werden.

Die Referenzkarte wird mithilfe des Python-Pakets *matplotlib* [3] [27] erstellt und ausgegeben. Anschließend kann die Visualisierung in verschiedenen, gängigen Bildformaten abgespeichert werden.

Vorverarbeitung

Um die Punktewolke verarbeiten zu können, wird ein Open3D-Geometry-Objekt der Punktewolke erstellt. Dieses Objekt wird in der Laufzeit in jedem Verfahrensschritt geändert und speichert die aktuellste Repräsentation. Die Open3D-Geometry-Klasse bietet zudem Funktionen zur Filterung und Verarbeitung des Punktewolken-Objektes.

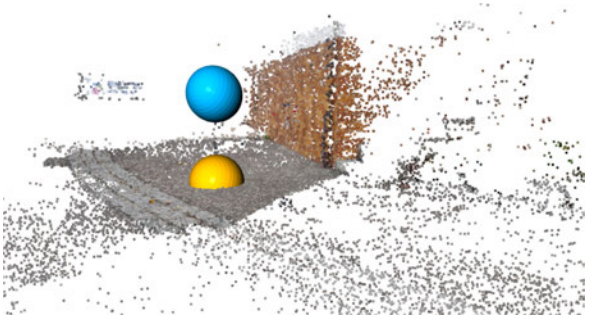


Abbildung 4.2.: Skalierungsmarker der Punktewolke. Zwei vom Benutzer gewählte Punkte an einem Pfosten in der Szene (vgl. Szene Nr. 3 Tabelle A.1 in Anhang A). Der Posten ist mit wenig Punkten zwischen der blauen und gelben Kugel rekonstruiert.

Da die Skalierung so viele Punkte wie möglich benötigt, um den Skalierungsfehler

[3] `https://matplotlib.org/`

durch die Punkteauswahl klein zu halten, wird die Punktewolke nach dem Einlesen skaliert. Hierfür muss die Option *scale-dist* gesetzt sein.

Die Punktewolke wird in dem interaktiven Fenster von *Open3D* [16] angezeigt und der Benutzer wählt die zwei Punkte zu der angegebenen Distanz. Die Punkte werden durch Kugeln markiert und von der Visualisierung an die Funktion zur Skalierung gegeben (vgl. Abbildung 4.2).

Die Datenbereinigung (vgl. Abschnitt 3.1) fährt dann mit dem *Statistical Outlier Filter* und der Homogenisierung der Dichte mit dem *Voxel Grid Filter* fort. Diese Funktionen sind bereits in *Open3D* implementiert und können mithilfe des Geometrie-Objektes aufgerufen werden.

Bodenschätzung

Für die Bodenausrichtung ist der in Abschnitt 3.2 vorgestellte Algorithmus (vgl. Algorithmus 1) implementiert. Die Punktewolke wird der Funktion zur Verfügung gestellt und die Parameter werden aus der Konfigurationsdatei geladen. *Open3D* schätzt für eine geladene Punktewolke die Punktenormalen für jeden Punkt. Da weder die Methode *estimate_normals* noch das Attribut *normals* die Residuen beinhaltet, werden diese mit einer Normalenschätzung ermittelt.

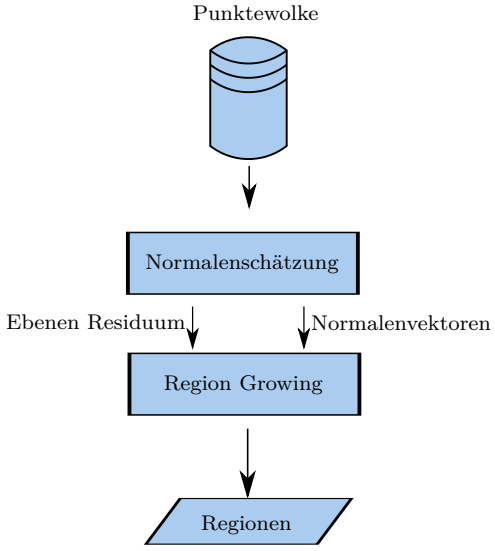


Abbildung 4.3.: Implementierungsschema der Bodenschätzung. Normalenvektoren der Punkte in der Punktewolke werden geschätzt. Ebenen Residuum und Normalenvektoren werden an den *Region Growing* Algorithmus weitergegeben. Regionen in der Punktewolke werden segmentiert.

Da der Region-Growing-Algorithmus nicht für Python zur Verfügung steht, wird dieser in der Funktion *segment_by_region* entwickelt. Es wird der in Abschnitt 2.3

vorgestellte Algorithmus implementiert.

Aufgrund der ausführlichen Erläuterung der Koordinatentransformation in Abschnitt 3.2 wird an dieser Stelle auf eine erneute Beschreibung des Ablaufes verzichtet. Für die Transformation der Punktewolke wurden die in *Open3D* implementierte Funktion des Geometry-Objektes zurückgegriffen.

Nach der Transformation der Punktewolke wird mit der Höhensegmentierung fortgefahren.

Höhensegmentierung

Die ausgerichtete Punktewolke kann jetzt mit dem in Abschnitt 3.3 vorgestellten Ansatz in Bereiche eingeteilt werden, die für das Fahrzeug hoch oder niedrig sind. Außerdem sind Bereiche oberhalb der Sensorreichweite und die Bodenpunkte nicht relevant, daher können diese Bereiche entfernt werden.

Die Punktewolke wird dafür mittels einer Begrenzungsbox (*Bounding Box*) in gestapelte Quader unterteilt, deren Kanten am Bodenkoordinatensystem ausgerichtet sind. Die Höhe der Quader kann je nach Fahrzeugabmaßen gewählt werden. Von einem Quader eingeschlossene Punkte werden anhand ihres Index mit der Funktion *select_by_index* [4] ausgewählt.

Dieser Verarbeitungsschritt gibt die geladene Punktewolke als zwei Punkte-Arrays zurück, die jeweils hohe oder niedrige Punkte enthalten. Die Punkte-Arrays werden im folgenden Schritt weiter verarbeitet.

Karten Initialisierung

Im Folgenden wird das Konzept und die Datenstruktur der implementierten Karte erläutert. Das untersuchte *Occupancy Grid* hat eine konstante Auflösung der Zellen und diese sind unabhängig voneinander, da ihr Status nur auf der von ihr belegten Umgebung beruht. Dies ermöglicht die Umsetzung als ein kartesisches Gitter in zwei Dimensionen. Ein kartesisches Gitter zeichnet sich durch gleichmäßig aufgelöste Zellen in mehreren Dimensionen aus.

Für die Umsetzung des *Occupancy Grids* in Python wurde ein objektorientierter Ansatz gewählt, da dieser Programmieraufwand bei sich oft wiederholenden Strukturen spart. Die Karte wird daher in zwei Klassen entwickelt (vgl. Abbildung 4.4). Die Klasse *Map* speichert alle Informationen der Karte, die zur Zeit der Initialisierung bekannt sind. Es werden die Auflösung und die Abmessungen der Karte als Parameter

[4] bereits in *Open3D* implementierte Funktion

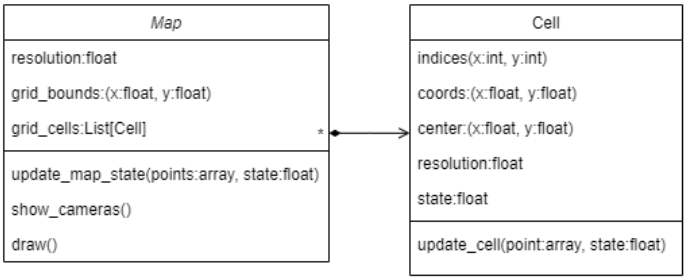


Abbildung 4.4.: Klassenkonzept des *Occupancy Grid*. Klassen Komposition der Klassen *Map* und *Cell* zur Abbildung der Datenstruktur eines *Occupancy Grids*.

benötigt. Aus diesen Parametern werden die Zellen der Karte berechnet. Diese werde als Instanzen der Klasse *Cell* gespeichert. Jede Instanz benötigt ihre Indizes in der x und y Dimension, die Auflösung der Zelle und die Koordinaten des Zellenurspungs. Aus diesen Parametern wird der Mittelpunkt der Zelle in den Koordinaten der Karte berechnet. Dieser Mittelpunkt wird für eine spätere Filterung benötigt. Ein weiteres Attribut jeder Zelle ist der Zustand. Dieser ist in einer Gleitkomma-Zahl zwischen 0 und 1 festgehalten. In dem implementierten *Grid* sind drei Zustandswerte (vgl. Tabelle 4.2) definiert.

Tabelle 4.2.: Zellenzustände der *Occupancy Grid Map*. Drei mögliche Zellenzustände als Gleitkomma-Zahlen.

Zustand	Bedeutung
0	freie Zelle
0,5	unbekannte Zelle
1	belegte Zelle

Funktionen und Filterung der Karte

Nach der Initialisierung der Karte werden die Punkte in die Karte einsortiert. Hierfür wurde die Methode *udpate_map_state* implementiert. Sie lädt die Punkte in einen *PointStore*. Der *PointStore* ist eine Datenstruktur aus der die Punkte einzeln gewählt und einsortiert werden können. Die Punkte werden nach dem Einsortieren aus dem *PointStore* gelöscht, um die Anzahl der nötigen Iterationen zu minimieren.

Die Punkte-Arrays mit den segmentierten Punkten aus dem Abschnitt Höhensegmentierung in die Zellen einsortiert und der Zellenzustand wird gesetzt. Hierfür steht die

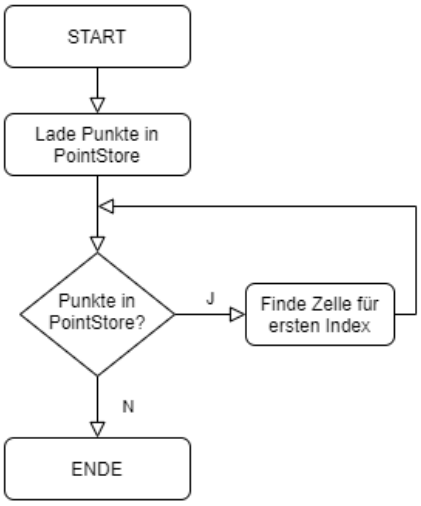


Abbildung 4.5.: Flussdiagramm zur *update_map_state* Methode. Die Methode *update_map_state* lädt die Punkte aus dem *PointStore* bis keine mehr vorhanden sind.

Methode *udpate_cell* des Zellen Objektes zur Verfügung. Die hinzugefügten Punkte werden in der Zelle gespeichert, um eine spätere Filterung zu ermöglichen.

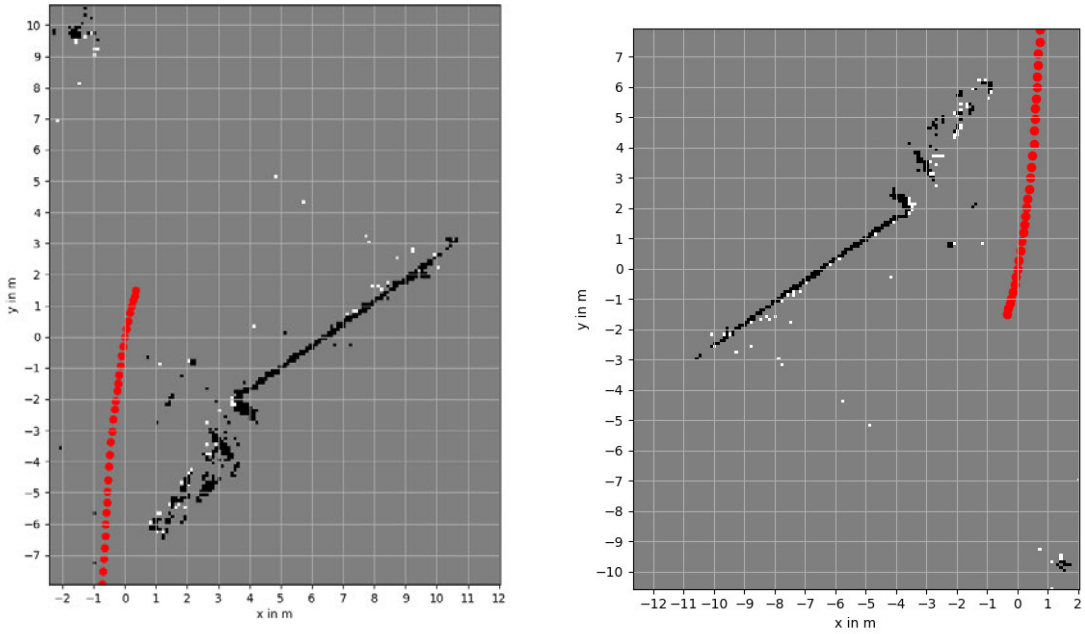
Die Filterung vereinzelter Zellen in der Karte wird über eine Punkteschwelle realisiert (vgl. Abschnitt 3.4). Für die Implementierung der Punkteschwelle wurde in der Methode *udpate_cell* eine Bedingung integriert, die bereits vorhandene Punkte berücksichtigt. Der Zellen Zustand wird erst auf den geforderten Wert gesetzt, wenn eine bestimmte Anzahl von hohen oder niedrigen Punkten erreicht wurde.

Abbildung 4.6b zeigt die gefilterte Karte einer Szene mit einem Pfosten (vgl. Szene Nr. 3 Tabelle A.1 in Anhang A). Es wurde eine Schwelle von zwei Punkten pro Zelle für hohe Punkte gesetzt. Es ist zu sehen, dass die Zellen um den Pfosten (Koordinate (-2, 1) in Abbildung 4.6b) überwiegend unbekannt sind, wogegen vorher vereinzelte Fehlrekonstruktionen zusehen waren.

Um eine bessere Orientierung in der Karte zu bekommen, wurde die Methode *show_cameras* implementiert, die Kamerastandpunkte anzeigen zu können. Sie wird mit der Option *vhm* aus Tabelle 4.1 aufgerufen und erfordert das Verzeichnis des Photogrammetrie Caches als Eingabewert. Die Kamerastandpunkte werden aus dem in Abbildung 4.3 dargestellten Photogrammetrie Cache [5] extrahiert. Sie erscheinen als rote Punkte in der Karte (vgl. Abbildung 4.6). Da die Standpunkte jedoch in

[5] von *meshroom* abgelegter Cache mit allen Zwischenschritten und Log-Dateien zur Rekonstruktion

dem Photogrammetrie Koordinatensystem (vgl. Abschnitt 3.2) definiert sind, wird die Bodentransformation auf die Koordinaten angewendet und die Punkte werden senkrecht in die Bodenebene projiziert.



(a) (b)

Abbildung 4.6.: gefilterte *Occupancy Grid Map* mit Kamerastandpunkten. Karte der Szene Nr. 3 mit eingeblendeten Kamerastandpunkten (rot) (vgl. Tabelle A.1 in Anhang A) (a) ohne Punkteschwelle. (b) Punkteschwelle für hohe Punkte. Mindestens zwei hohe Punkte in einer Zelle für den hohen Zustand..

4.3. Validierung

Im folgenden Abschnitt wird das in dieser Arbeit entwickelte und implementierte Verfahren anhand von Referenzdaten validiert. Ziel ist es, die entwickelte Karte, anhand einer Referenzmessung qualitativ zu bewerten. Um eine aussagekräftige Referenzmessung zu erhalten, wird mit einem LiDAR Sensor eine Straßenszene vermessen. Am Ende der LiDAR Messung, wird die Szene gefilmt und mithilfe der Photogrammetrie rekonstruiert.

Die Videoaufnahmen der Szenen wurden wie in Unterabschnitt 2.1.3 beschrieben aufgenommen, um eine gute Vergleichbarkeit mit den bisherigen Szenen der Arbeit zu erhalten.

Im Abschnitt Referenzfahrzeug wird die Messtechnik und das Koordinatensystem (KOS) des Fahrzeugs erläutert. Anschließend wird die Datenverarbeitung nach der Messung mit dem Fahrzeug erläutert. Zuletzt werden die Karten der Szenen erstellt und verglichen.

Referenzfahrzeug

Das Referenzfahrzeug ist eine mit zusätzlicher Messtechnik ausgestattete Mercedes-Benz E-Klasse (S213). Serienmäßig ist diese mit einem Kamerasystem, Radar und Ultraschallsensorsystem ausgestattet. In der Dachbox in Abbildung 4.7a ist unter anderem ein differentielles, globales Positionierungssystem (DGPS) und ein LiDAR Linien-Scanner verbaut. Der LiDAR Sensor der Firma SICK ist ein 2D-Sensor, der am Heck des Fahrzeugs mit einer Neigung zum Boden ausgerichtet ist. Durch den 2D-Sensorbereich wird die Umwelt in einer Linie quer zur Fahrtrichtung abgetastet.

(a) (b)

Abbildung 4.7.: LiDAR Referenzfahrzeug. (a) Referenzfahrzeug mit LiDAR und DGPS System (b) Fahrzeugkoordinatensystem des vehicle model.

Das DGPS dient zur Erfassung der Fahrzeugbewegung (Odometrie) und wird für die Fusion der aufgezeichneten Punktewolken verwendet. Die Antennen (grün) sind auf dem Dach des Fahrzeugs angebracht.

Die Ultraschallsensoren erfassen die Umgebung, welche vom Steuergerät in einer Umgebungskarte verarbeitet und aufgezeichnet wird (vgl. Abschnitt 1.1).

Alle Sensorsysteme messen dabei in eigenen Koordinatensystemen und müssen später auf ein gemeinsames Koordinatensystem bezogen werden. Für die Fahrzeugbewegung ist im Fahrzeugmodell (englisch: *vehicle model*, kurz: VHM) das in Abbildung 4.7b gezeigte, rechtshändige Koordinatensystem festgelegt, dass sich in der Mitte der Fahrzeughinterachse befindet. Die Translation des Fahrzeugs ist dabei mit den Koordinaten (x, y) und die Rotation des Fahrzeugs (engl.: *yaw*) mit dem Winkel ψ beschrieben. Dieses Koordinatensystem wird initialisiert, wenn das Referenzfahrzeug gestartet wird.

Nach dem Start der Messung zeichnen alle Sensoren Daten auf und leiten diese an einen zentralen Messdaten-Rechner. Dieser Rechner erstellt die Daten-Dateien und lädt sie auf einen Cloud-Server hoch. Die Verarbeitung dieser Daten ist im Folgenden beschrieben.

Datenverarbeitung

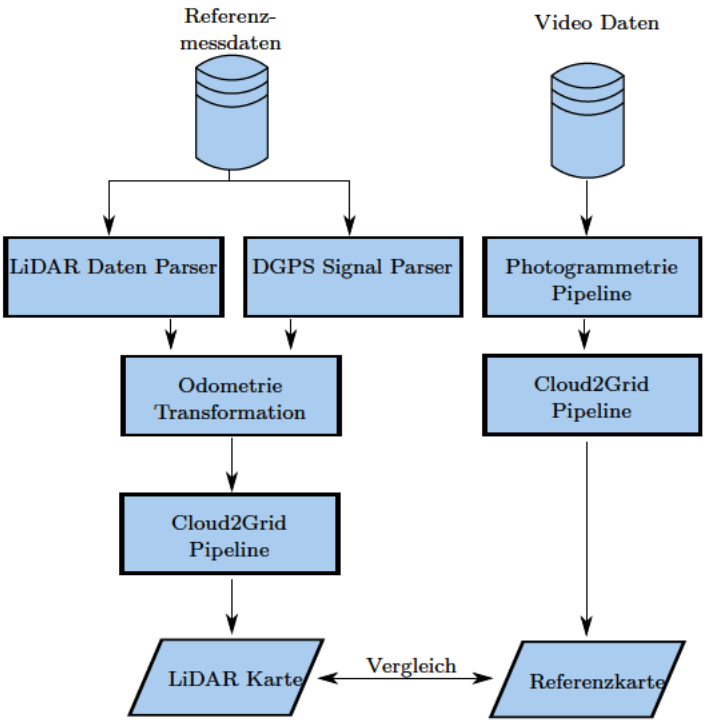


Abbildung 4.8.: Validierungsprozess des *Cloud2Grid* Verfahrens. (links) Verarbeitung der Referenzmessdaten mithilfe der bestehenden *Parser*. (rechts) Verarbeitung der Video Daten mithilfe der Photogrammetriepipeline.

Aus der Fahrzeugmessung werden die LiDAR-Daten und die DGPS-Daten benötigt, um die Punktewolke zu erzeugen. Diese Daten werden nach dem in Abbildung 4.8 abgebildeten Schema verarbeitet. Sie liegen zuerst als Binär-Daten vor und werden mit vorhandenen Dekodierungsanwendungen (engl.: *Parser* oder Decoder) in csv-Daten konvertiert.

Die extrahierte LiDAR Punktewolke hat ihren Ursprung am Anfang der Messung. Um sie mit den Photogrammetrie Daten überlagern zu können, muss das Koordinatensystem an das Ende der Messung transformiert werden. An dieser Position ist das Fahrzeug in der Szene abgebildet. Die Transformation erfolgt anhand der Odometriedaten des Fahrzeugs in Abbildung 4.9. Die aufgenommenen Odometriedaten werden an

den Anfang der Messung transformiert. Anschließend wird die LiDAR Szene anhand der Odometrie in der x-y-Ebene verschoben.

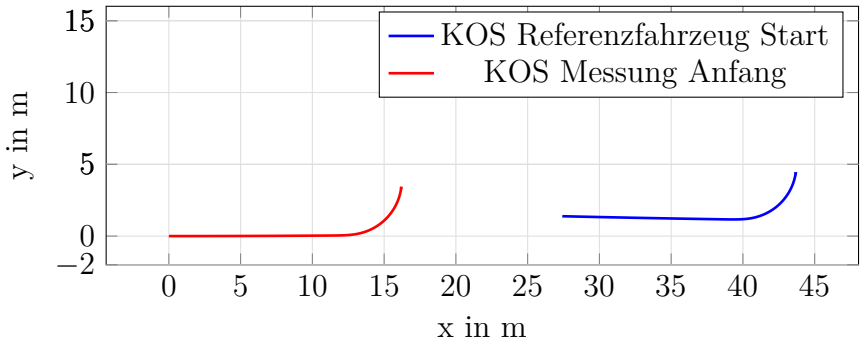


Abbildung 4.9.: Transformation der VHM Odometrie zum Start der Messung. Das KOS Referenzmessung Start wird an den Anfangspunkt der Messung transformiert.

Die transformierte Punktewolke wird anschließend als Polygon-Datei exportiert. Das Cloud2Grid-Skript liest diese Punktewolke ein und erzeugt die Referenzkarte der LiDAR-Szene.

Die Referenz in der Photogrammetrie Punktewolke wird anhand von zwei Punkten $\{\vec{p}_r, \vec{p}_l\}$ in einer interaktiven Ansicht festgelegt. Diese werden an den Naben der Hinterachse definiert. Somit ist, wie in Abbildung 4.10 gezeigt, der Mittelpunkt $\vec{r}_{vhm}$ des Verbindungsvektors $\vec{r}_{rl}$, der Ursprung des VHM Koordinatensystems.

Die Orientierung der x-y Ebene des Koordinatensystems $\mathbf{E}_{vhm}$ wird mit der Rotationsmatrix $\mathbf{R}_{vhm}$ an dem Bodenkoordinatensystem $\mathbf{E}_1$ ausgerichtet. Die Rotation $\mathbf{R}_{vhm}$ wird erneut mit dem Kabsch-Algorithmus ermittelt. Die Punktewolke wird dann anhand der Transformation

$$\mathbf{T}_{sfm2vhm} = \begin{bmatrix} \mathbf{R}_{vhm} & \vec{r}_{vhm} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.1}$$

ausgerichtet. Abbildung 4.11 zeigt die überlagerten Punktewolken. Die LiDAR Punktewolke ist in Weiß und die Photogrammetrie Rekonstruktion in Rot gezeigt.

Ergebnis

Um ein aussagekräftiges Ergebnis zu erhalten, werden Referenzkarten von zwei verschiedene Szenen mit dem in Abbildung 4.8 abgebildeten Prozess, erstellt. Da sich die Szene des Parkplatzes aufgrund fehlender Punkte an den hinteren Naben nicht hinreichend genau ausrichten lässt, wird stattdessen die Szene Nr. 1 in Tabelle A.2 (vgl. Anhang A)

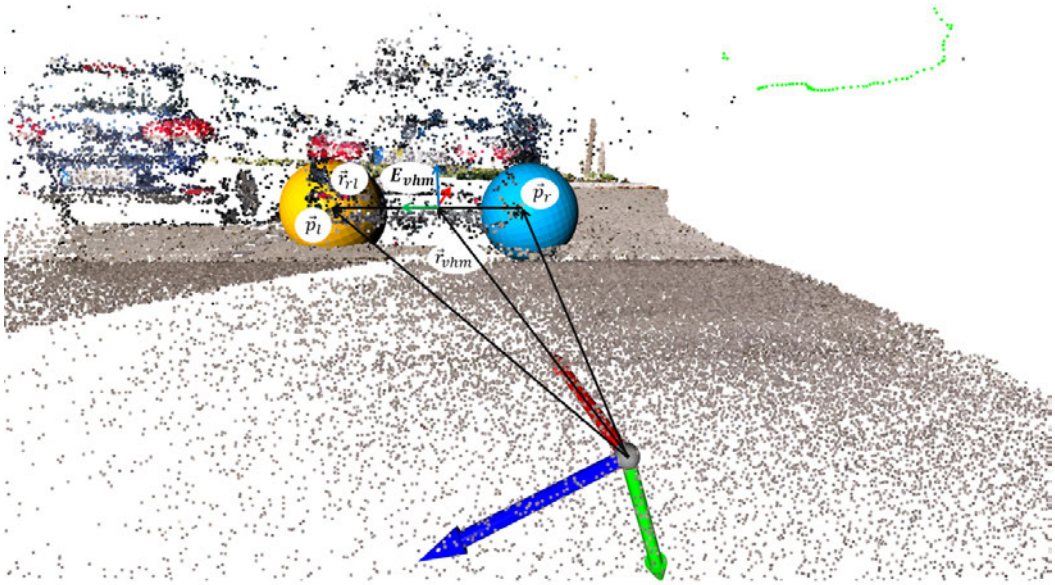


Abbildung 4.10.: Konstruktion des Referenzpunktes in der Photogrammetrie Punktewolke. Konstruktion des Referenzkoordinatensystems $\mathbf{E}_{vhm}$ im Fahrzeug durch die Auswahl von zwei Punkten (Kugeln).

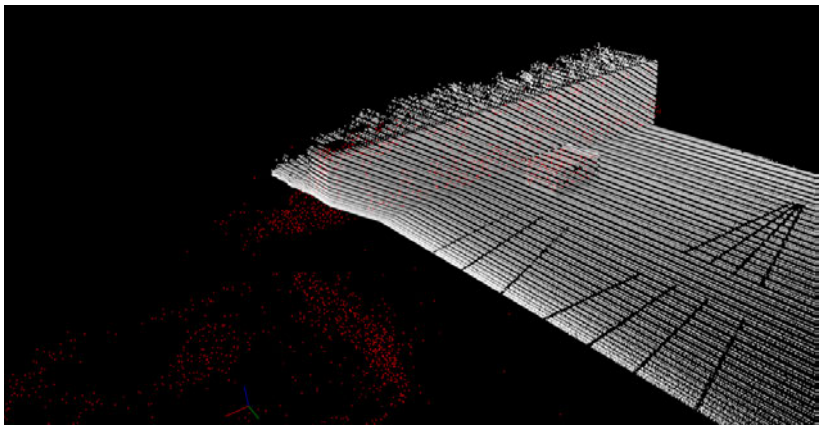


Abbildung 4.11.: Überlagerte LiDAR Punktewolke mit Photogrammterie Punktewolke. Photogrammetrie in Rot, LiDAR in Weiß. Szene Nr. 1 in Tabelle A.2 (vgl. Anhang A).

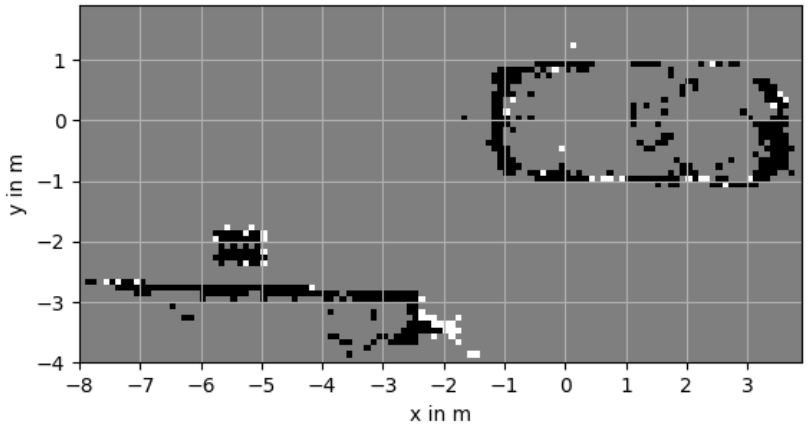
im Folgenden erläutert. Anhand dieser Szene lassen sich die Genauigkeitsunterschiede gut ermitteln, da einfache geometrische Formen in der Szene enthalten sind und diese mit vielen Punkten rekonstruiert wurden.

Im Allgemeinen unterscheiden sich die LiDAR Karte und die Photogrammetrie Karte in zwei Punkten. Das Referenzfahrzeug ist in der LiDAR Szene nicht mit abgebildet, da der 2D LiDAR Sensor am Fahrzeug angebracht ist. Die Referenz zu dem Fahrzeug ist jedoch durch das Koordinatensystem gegeben. Des Weiteren ist die Auflösung der LiDAR Karte um ein Vielfaches höher als die Photogrammetrie Auflösung, diese

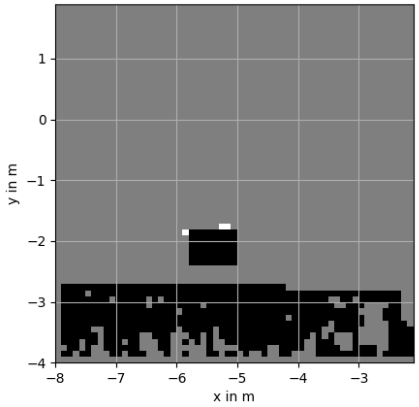
Unterschiede sind deutlich an dem Detailgrad in der Karte sichtbar.

Die in Abbildung 4.12 abgebildeten Karten wurden mit einer Zellenauflösung von 10 cm berechnet. Damit sollte die in der Szene platzierte Box (BxT 60 cm x 80 cm) mit 6 x 8 Zellen abgebildet werden.

Die Geometriedarstellung der Photogrammetrie Szene stimmt mit der LiDAR Szene qualitativ überein. Die Orientierung der Karte ist richtig im VHM Koordinatensystem abgebildet. Die relevanten Objekte, wie die Box und die Wand sind in der Karte abgebildet. Bei genauerer Betrachtung der Zellstati ist zu sehen, dass die Box nicht durchgehen als Hindernis angezeigt wird. Diese Eigenschaft konnte auf die inhomogene Rekonstruktion zurückgeführt werden, da die Zelle nur am Rand durch Punkte belegt wurde. Dazu wurde die Schwelle für hohe Punkte in der Zelle herabgesetzt, sodass nur noch ein hoher Punkt für eine hohe Zelle benötigt wird.



(a)



(b)

Abbildung 4.12.: Vergleich LiDAR Szene zu Photogrammetrie Szene. Auflösung der Zellen 10cm, Kartendarstellung ist mit einer Schwelle von zwei Punkten pro Zelle für hohe Punkte gefiltert. Szene Nr. 1 in Tabelle A.2 (vgl. Anhang A)(a) Photogrammetrie Szene mit einer rekonstruierten Box und Wand im unteren linken Bereich der Szene. (b) LiDAR Szene des Referenzfahrzeugs mit einer rekonstruierten Box und Wand.

5. Schlussbetrachtung

In dieser Arbeit wurde ein Verfahren zur Erstellung einer Referenzkarte für die Validierung von Park- und Manövrierassistenzsystemen entwickelt. Die Grundlage dieser Referenzkarte sind Punktewolken, die mithilfe der Photogrammetrie erzeugt wurden. Das Verfahren wurde in einem protoypischen Ansatz in der Sprache Python umgesetzt. Die Implementierung wurde in Abschnitt 4.3 anhand von ausgewählter Referenzszenen mit einem LiDAR-Referenzfahrzeug validiert.

Das Verfahren wurde dafür in die vier separaten Bestandteile Datenaufbereitung, Bodenschätzung, Höhensegmentierung und Kartierung unterteilt. In Kapitel 3 wurden für diese Bestandteile verschiedene Methoden zur Umsetzung recherchiert und untersucht.

Da die Punktewolken eine sehr schwankende lokale Qualität haben, wurden in Abschnitt 3.1 Methoden zur Filterung untersucht. Diese zeigten, dass die Filterung anhand eines *Voxel Grid Filters* am effektivsten ist, um die Dichte zu homogenisieren. Es wurde außerdem ein Bodenmodell basierend auf einer Ebenen-Schätzung entwickelt (vgl. Abschnitt 3.2), dass es ermöglicht das Koordinatensystem der Punktewolke am Boden auszurichten. Die Ausrichtung kann damit ohne Informationen zur Kameraposition erfolgen. Dies war erforderlich, da die Punktewolken keine definierte Lage in ihrem Koordinatensystem haben. Die Höhensegmentierung wurde auf Basis des ausgerichteten Koordinatensystems erreicht. Die Punktewolke konnte so in einen kollisionsrelevanten und nicht relevanten Bereich segmentiert werden, da die Bodenschätzung sich als sehr robust erwies und das Koordinatensystem mit der z-Achse senkrecht auf dem Boden ausgerichtet werden konnte. Die Darstellung der Referenzkarte erfolgte anschließend in einer Höhenkarte (vgl. Abschnitt 3.4), die dem *Occupancy Grid Mapping* nachempfunden ist. Hier zeigte sich, dass eine weitere Filterung der Karte nötig ist, da die Karte noch vereinzelte falsch hohe oder niedrige Zellen beinhaltet. Es wurden eine Punkteschwelle für die Zellen implementiert. Diese Schwelle unterdrückt ein Status-Update der Zelle bis zu einer definierten Punkteanzahl.

Folgend werden die Ergebnisse kritisch bewertet und ein Ausblick für Erweiterungen und Anwendungsfälle gezeigt.

5.1. Bewertung

Da eine Skalierung der Punktewolke zu Beginn des Verfahrens nötig ist, um die Punktewolken Filter anwenden zu können, wurde hier ein einfacher Ansatz gewählt. Die Skalierung anhand von zwei gewählten Punkten, ist jedoch nicht genau und erfordert einen manuellen Schritt. Dieser Schritt ist zeitlich nicht kritisch. Jedoch kann nicht immer sichergestellt werden, dass die gewünschten Punkte in der Punktewolke rekonstruiert wurden. Wenn Marker für die Skalierung eingesetzt werden würden, könnte sichergestellt werden, dass diese erkannt werden.

Das in Abschnitt 3.2 entwickelte Bodenmodell hat sich als sehr robust herausgestellt, da es auch kleinere Bodenpartien segmentiert und korrekt ausrichtet.

Da das Fahrzeug in der Szene nicht durch Marker als absolute Referenz verwendet werden konnte, wurde die Referenz zu dem Fahrzeug ebenfalls manuell durch gewählte Punkte eingebracht. Dies diente hauptsächlich der Validierung des Verfahrens und wurde daher nicht oft verwendet. Dies hatte jedoch den Nachteil, dass nur bestimmte Punktewolken ausgerichtet und validiert werden konnten, da nicht alle Szenen ausreichend gut rekonstruiert werden konnten.

Die Validierung (vgl. Abschnitt 4.3) zeigte, dass die Referenzkarte, die Fahrzeugumgebung qualitativ gut abbildet und die Geometrie der Objekte identifizierbar ist. Dies konnte durch verschiedene Referenzszenen (vgl. Abschnitt A.2 Referenz Szenen) gezeigt werden.

5.2. Ausblick

Das entwickelte und implementierte Verfahren ermöglicht es, die Umgebung eines Testfahrzeugs zu visualisieren und anschließend die aufgenommenen Messdaten zu plausibilisieren. Dies würde den direkten Vergleich der Ultraschall-Echos mit den durch die Photogrammetrie rekonstruierten Daten ermöglichen. Das Anzeigen der Messdaten direkt in der entwickelten Karte ist jedoch noch nicht möglich. Hierzu ist noch die Entwicklung eines Messobjekt-Parsers nötig. Dieser kann dann die aufgenommenen Messobjekte mithilfe der Bibliothek *matplotlib* in der Karte anzeigen.

Die Ultraschall Daten des Park- und Manövrierassistenzsystems werden aktuell von Hand gelabelt. Das heißt, die Echos werden mit semantischen Informationen zu ihrem Ursprung gekennzeichnet. Dieses Labeln ist zeitaufwendig und nicht immer konsistent.

Die Referenzkarte kann durch die Videoaufnahmen der Photogrammetrie, simultan zur Erstellung, mit semantischen Informationen angereichert werden. Hierzu ist es nötig die Objektinformationen aus den Videoaufnahmen in den 3D-Bereich zu projizieren. Die Informationen können dann den einzelnen Zellen geordnet werden.

A. Verwendete Szenen

In diesem Kapitel ist ein Überblick über alle in der Arbeit verwendeten Szenen gezeigt. Dieser beinhaltet Farbbilder der Szene und die rekonstruierte Punktewolke. In Abschnitt A.1 sind alle Szenen aufgeführt, die mithilfe der Photogrammetrie Pipeline rekonstruiert worden sind. Im Abschnitt A.2 werden Szenen mit einer LiDAR Referenzmessung aufgeführt.

A.1. Photogrammetrie Szenen

Tabelle A.1.: Rekonstruierte Szenen.

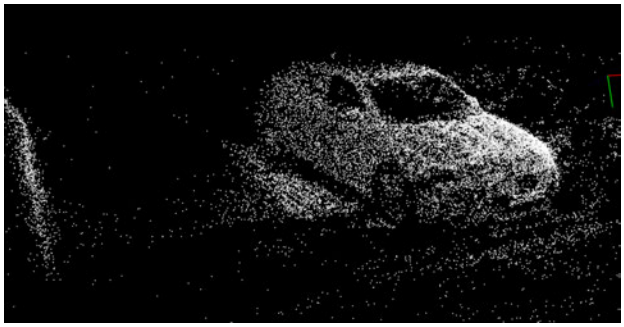
(1) Straßenszene mit flachem Bordstein

47 Aufnahmen

Photogrammetrie Szene
55153 Punkte

(2) Entwicklungsfahrzeug mit Tarnfolie

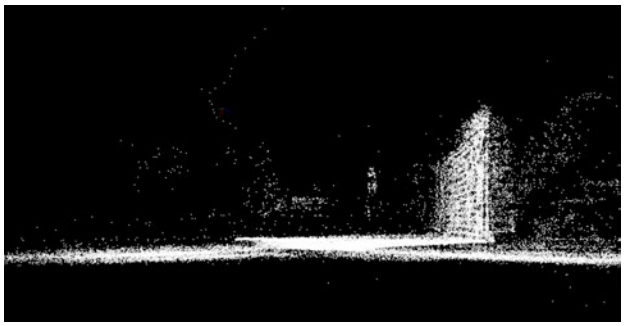
34 Aufnahmen



Photogrammetrie Szene
17697 Punkte

(3) Parkplatz Szene mit Pollern

52 Aufnahmen



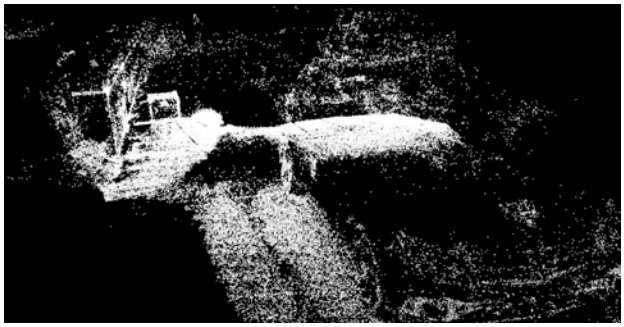
Photogrammetrie Szene
73818 Punkte

A.2. Referenz Szenen

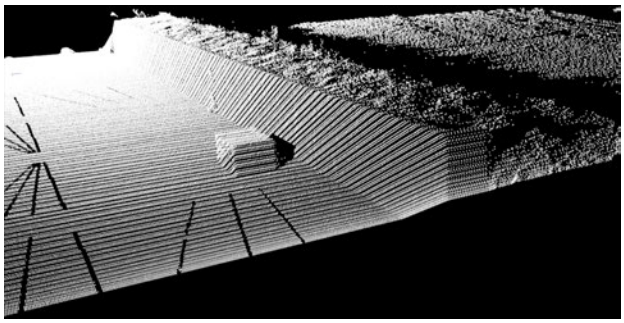
Tabelle A.2.: Szenen der Referenzmessung.

(1) Box an einer Wand platziert

74 Aufnahmen



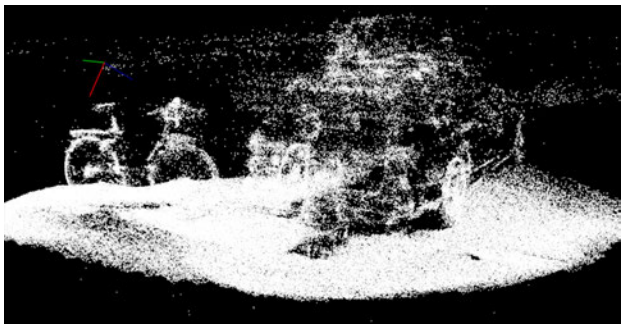
Photogrammetrie Szene
100429 Punkte



LiDAR Szene
308949 Punkte

(2) Fahrzeug rückwärts gegen Bordstein mit Objekten

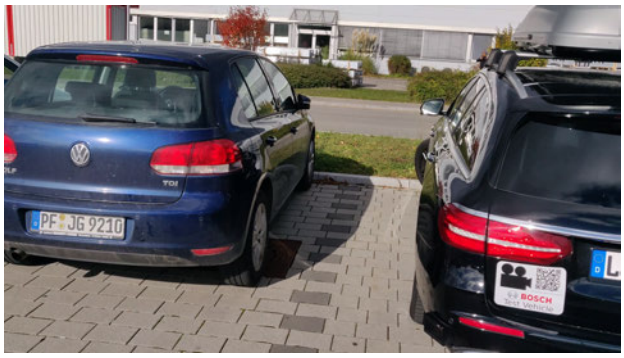
121 Aufnahmen



Photogrammetrie Szene
233826 Punkte

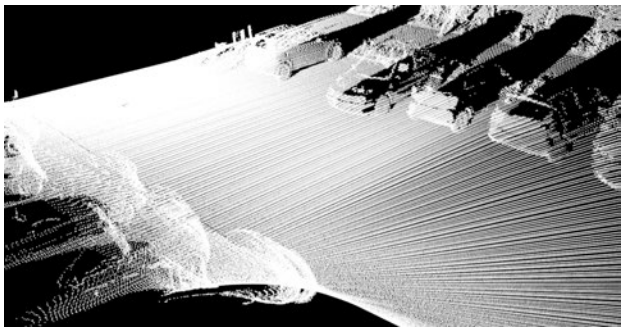
LiDAR Szene
336520 Punkte

(3) Fahrzeug in einer Parklücke



83 Aufnahmen

Photogrammetrie Szene
93466 Punkte



LiDAR Szene
451699 Punkte

Literatur

[1] Peter (ICCT Europe) Mock. *European Vehicle*. Techn. Ber. Berlin, Germany: The International Council On Clean Transportation, 2019, S. 105. URL: https://tinyurl.com/y3t7kmta.

[2] Ig. *Neuzulassungsstatistik - Faherassistenzsysteme immer beliebter*. 2019. URL: https://industrie.de/technik/assistenzsysteme-immer-beliebter/ (besucht am 31. 10. 2020).

[3] Euro NCAP. "Euro NCAP 2025 Roadmap: In pursuit of Vision Zero". In: *Euro NCAP Report* (2017), S. 1 19. URL: https://cdn.euroncap.com/media/30700/euroncap-roadmap-2025-v4.pdf.

[4] Robert Bisch GmbH. [Online; accessed November 4, 2020]. 2019.

[5] Wolfgang Förstner und Bernhard P Wrobel. *Photogrammetric Computer Vision Statistics, Geometry, Orientation and Reconstruction*. Bd. 1. 6330 Cham, Switzerland: Springer Nature, 2016, S. 8 46. ISBN: 9783319115498. DOI: 10.1016/S0076-5392(09)60371-4.

[6] OpenCV. [Online; accessed August 23, 2020]. 2019.

[7] David G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* (2004). URL: https://www.cs.ubc.ca/{~}lowe/papers/ijcv04.pdf.

[8] OpenCV. [Online; accessed August 23, 2020]. 2019.

[9] OpenCV. [Online; accessed November 28, 2020]. 2019.

[10] David Nistér und Henrik Stewénius. "Scalable recognition with a vocabulary tree". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Bd. 2. 2006. DOI: 10.1109/CVPR.2006.264.

[11] Marius Muja und David G. Lowe. "Fast approximate nearest neighbors with automatic algorithm configuration". In: *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications*. Bd. 1. 2009. DOI: 10.5220/0001787803310340.

[12] Pierre Moulon, Pascal Monasse und Renaud Marlet. “Adaptive Structure from Motion with a Contrario Model Estimation”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Bd. 7727 LNCS. PART 4. 2013, S. 257 270. DOI: 10.1007/978-3-642-37447-0_20. URL: http://link.springer.com/10.1007/978-3-642-37447-0{_}20.

[13] Michal Jancosek und Tomas Pajdla. “Multi-view reconstruction preserving weakly-supported surfaces”. In: *CVPR 2011*. IEEE, 2011, S. 3121 3128. ISBN: 978-1-4577-0394-2. DOI: 10.1109/CVPR.2011.5995693. URL: http://ieeexplore.ieee.org/document/5995693/.

[14] Martin A. Fischler und Robert C. Bolles. “Random sample consensus”. In: *Communications of the ACM* 24.6 (1981), S. 381 395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://dl.acm.org/doi/10.1145/358669.358692.

[15] Wikimedia. [Online; accessed November 28, 2020]. 2007.

[16] Zhihan Lu, Paul Guerrero, Niloy J. Mitra u. a. “Open3D”. In: 2016. DOI: 10.1145/2945292.2945302.

[17] Wikimedia. [Online; accessed November 28, 2020]. 2019.

[18] Steven W. Zucker. “Region growing: Childhood and adolescence”. In: *Computer Graphics and Image Processing* 5.3 (1976), S. 382 399. ISSN: 0146664X. DOI: 10.1016/S0146-664X(76)80014-7. URL: https://linkinghub.elsevier.com/retrieve/pii/S0146664X76800147.

[19] T. Rabbani, F. van den Wildenberg und G. Vosselman. “Segmentation of point clouds using smoothness constraint”. In: *International archives of photogrammetry, remote sensing and spatial information sciences* 36.5 (2006), S. 248 253. ISSN: 16821750. URL: https://www.isprs.org/proceedings/XXXVI/part5/paper/RABB{_}639.pdf.

[20] W. Kabsch. “A solution for the best rotation to relate two sets of vectors”. In: *Acta Crystallographica Section A* 32.5 (1976), S. 922 923. ISSN: 0567-7394. DOI: 10.1107/S0567739476001873. URL: http://scripts.iucr.org/cgi-bin/paper?S0567739476001873.

[21] Jugesh Sundram, Duong Van Nguyen, Gim Song Soh u. a. “Development of a Miniature Robot for Multi-robot Occupancy Grid Mapping”. In: *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, 2018, S. 414 419. ISBN: 978-1-5386-7066-8. DOI: 10.1109/ICARM.2018.8610745. URL: https://ieeexplore.ieee.org/document/8610745/.

[22] *CloudCompare*. 2020. URL: http://www.cloudcompare.org/.

[23] Radu B. Rusu. *Removing outliers using a StatisticalOutlierRemoval filter*. 2020. URL: https://pointclouds.org/documentation/tutorials/statistical{_}outlier.html{\#}statistical-outlier-removal (besucht am 13. 11. 2020).

[24] Radu B. Rusu. *Downsampling a PointCloud using a VoxelGrid filter*. 2020. URL: https://pointclouds.org/documentation/tutorials/voxel{_}grid.html (besucht am 13. 11. 2020).

[25] W Gellert, S Gottwald, M Hellmich u. a. *The VNR Concise Encyclopedia of Mathematics*. Hrsg. von W. Gellert, S. Gottwald, M. Hellwich u. a. 2. Aufl. Dordrecht: Springer Netherlands, 1990, S. 768. ISBN: 978-94-011-6984-4. DOI: 10.1007/978-94-011-6982-0. URL: http://link.springer.com/10.1007/978-94-011-6982-0.

[26] H. Moravec und Alberto Elfes. "High resolution maps from wide angle sonar". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Bd. 2. Institute of Electrical und Electronics Engineers, 1985, S. 116 121. DOI: 10.1109/ROBOT.1985.1087316. URL: http://ieeexplore.ieee.org/document/1087316/.

[27] John D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science & Engineering* 9.3 (2007), S. 90 95. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.55. URL: http://ieeexplore.ieee.org/document/4160265/.

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Gehring

Vorname: Philipp

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Verfahren zur Erstellung von Referenzkarten aus kamerabasierten Punktewolken für die Validierung von Parkassistenzsystemen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Stuttgart (Ort) 1.12.2020 (Datum) Unterschrift im Original