HAW HAMBURG

Bachelorarbeit

Tim Krogmann

Objekterkennung in 3D-Lidardaten

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Tim Krogmann

Objekterkennung in 3D-Lidardaten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Tim Tiedemann
Zweitgutachter: Prof. Dr. Andreas Meisel

Eingereicht am: 19.01.2021

Tim Krogmann

Thema der Arbeit

Objekterkennung in 3D-Lidardaten

Stichworte

Punktwolken, Vorverarbeitung, Segmentierung, Merkmale, Klassifizierung

Kurzzusammenfassung

In dieser Arbeit wird mit einem 3D-LiDAR Sensor in einem Raum Entfernungsdaten aufgenommen. Es wird gezeigt, welche Verarbeitungsschritte notwendig sind, um in den lückenhaften Daten Objekte wiederzuerkennen und welche Hindernisse in jedem Schritt überwältigt werden müssen. Es werden erforderliche Segmentationsverfahren gezeigt, welche die Daten in kleinere Gruppen von Punkten einteilen und wertlose Punkte entfernen. Weiterhin werden Merkmalsvektoren aus den Daten extrahiert, die Informationen über die Oberfläche der zu erkennenden Objekte codieren und damit eine kompakte Beschreibung der Objekte in den Daten sind. Die Merkmalsvektoren werden dann von einer Support-Vector-Machine klassifiziert und es wird untersucht, welche Merkmalsvektoren sich besser und welche schlechter eignen.

Tim Krogmann

Title of Thesis

Object recognition in 3D-lidar data

Keywords

Pointclouds, Preprocessing, Segmentation, Features, Object recognition, Classification

Abstract

In this bachelor thesis a 3D-LiDAR sensor is used to record range data in a room. It is shown which preprocessing steps are necessary to recognize objects in the occluded data and which obstacles have to be overcome in each step. Required segmentation procedures

are shown, which divide the data into smaller groups of points and remove uninteresting points from the cloud. In addition, feature vectors are extracted from the data, which encode information about the surface of the objects and are therefore a compact description of the objects in the data. The feature vectors are then classified by a support vector machine and it is examined which feature vectors are recognized better and which are recognized worse.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Ein neumodischer 3D-Laserscanner erzeugt präzise 3D-Karten, die in Form von 3D-Karten die Umwelt darstellen. Das macht sie nicht nur für das Gebiet des autonomen Fahren interessant, sondern auch für die Robotik, Landwirtschaft und viele weitere Gebiete. Durch die zusätzlichen Tiefeninformationen ist der Laserscanner, anders als herkömmliche 2D-Kameras nicht abhängig von guten Lichtverhältnissen. Die LiDAR Technologie und die Sensoren, die diese Technologie verwenden, machen zwar rasante Fortschritte, jedoch bleibt das Erkennen von Objekten in LiDAR Daten ohne weitere Farbinformationen eine Herausforderung und wird aufgrund von Störsignalen, Rauschen und Lücken in den Daten zu einer komplexen Aufgabe. Dazu kommt die hohe benötigte Rechenleistung durch die hohe Datenmenge.

1.2 Zielsetzung

In dieser Arbeit wird ein Velodyne 3D VLP-16 LiDAR in Betrieb genommen und damit in einem Büroraum Entfernungsdaten von Gegenständen erfasst. Das Ziel der Arbeit ist, die zu den Stuhl-Clustern gehörigen Punkten aus den rauschenden und lückenhaften Entfernungsdaten zu extrahieren und diese anhand von berechneten Merkmalen richtig zu klassifizieren. Dafür werden bekannte Verfahren angewendet, die sich bei dem Umgang mit 3D-Daten bewährt haben. Es werden vor-verarbeitende Schritte sowie Vor- und Nachteile dieser gezeigt.
Anders als bei [10], die mit nahezu perfekt Daten gearbeitet haben, enthalten die Daten nur die "Linien", die mit einer vollständigen Umdrehung des Sensors erzeugt wurden. Das ist vergleichbar Daten mit einem fest montierten LiDAR-Sensor. Es entstehen viele

Lücken sowie große Änderungen in der Dichte innerhalb der Daten und erschwert die Extraktion und Bewertung von Informationen. Mit Segmentierungs-Verfahren werden die Daten zu Clustern segmentiert und die Ergebnisse bewertet.

Für die erstellten Cluster werden Merkmale berechnet, welche als Feature-Deskriptoren beschreiben kompakt unterschiedliche Eigenschaften des jeweiligen Cluster beschreiben.

Für die Klassifizierung der Deskriptoren der Objekte wird eine Support-Vector-Machine trainiert, welche dann jedes Feature-Histogramm klassifiziert, indem sie jedem Cluster eine Klasse zuordnet. Die SVM wird mit verschiedenen Parametern auf ihre Genauigkeit analysiert und schließlich wird evaluiert, welche Features für die Klassizifierung besser geeignet und welche schlechter geeignet sind.

1.3 Struktur

In dem 2. Kapitel werden zunächst Grundlagen dargestellt. Des weiteren wird der verwendete LiDAR-Sensor vorgestellt und es zeigt wissenschaftliche Arbeiten mit einer ähnlichen Vorgehensweise. Das 3. Kapitel gibt einen kurzen Einblick in die Daten und den Datenfluss im System und es werden die benutzten Bibliotheken aufgelistet. Im 4. Kapitel werden die verwendete Filter und Algorithmen gezeigt, deren Funktionsweise und Zweck erklärt und dargestellt. Es wird die Segmentierung der Daten gezeigt und vorhandene Probleme diskutiert. Danach werden die verwendeten Feature-Deskriptoren für die Klassifizierung vorgestellt. Es folgt die Trainings- und Testphase und es wird das Klassifizierungsverfahren Support-Vector-Machine vorgestellt. Das 5. Kapitel veranschaulicht die Ergebnisse und es wird gezeigt, welche Feature-Deskriptoren die höchste Genauigkeit erzielten konnten. Kapitel 6 beinhaltet die Zusammenfassung und ein Ausblick für weitere Arbeiten.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für die Arbeit zusammengefasst und die verwendete Hardware dargestellt.

2.1 LiDAR

LiDAR (**li**ght **d**etection **a**nd **r**anging) ist ein Verfahren zur Bestimmung der Entfernung von Objekten [18]. Der LiDAR-Sensor misst die Zeit und Intensität von reflektierten Laserstrahlen, um so die Entfernung zu den Objekten zu berechnen und damit die Umgebung durch 3D-Punkte abbildet[17] und so 3D-Karten von der Umwelt erzeugt.

2.2 Verwendete Hardware

Für diese Arbeit wurde ein Velodyne VLP-16 Puck verwendet, der mit 16 Infrarotlasern und 16 Infrarot Transmittern ausgestattet ist. Der Sensor hat ein horizontales Sichtfeld von 360° und kann vertikal einen Winkel von 30° abdecken. Alle 0.1s erzeugt der Sensor etwa 30000 Punkte mit der Time-Of-Flight Methode. Dabei misst der Sensor nicht nur den Zeitpunkt, womit die $X-, Y-$, und Z- Werte der Punkte berechnet werden, sondern auch die Intensität des reflektierten Signals[17]. Die Intensitätswerte können Informationen über das Material des reflektierten Objektes enthalten und für sind für eine erfolgreiche Klassifizierung geeignet[7], haben aber hier keine Verwendung gefunden. Die Verarbeitung und Berechnungen erfolgten auf einem 2.4 GHz Dual-Core Prozessor mit 8 GB Arbeitsspeicher auf Ubuntu 18.04.

In Abbildung 2.1 ist der verwendete Sensor und seine Umgebung zu sehen.

Abbildung 2.1: Umgebung und Position des VLP-16 Puck

2.3 Verwandte Arbeiten

In verschiedenen Arbeiten wurde gezeigt, dass die Klassifizierung von Punktmengen eine hohe Genauigkeit erzielen kann. In [7] werden aus der Luft aufgenommene LiDAR Daten in verschiedene Klassen (Gebäude, Straße, Baum) eingeteilt. Anhand von geometrischer Eigenschaften ihrer Oberfläche und den gemessenen Intensitätswerten werden die Daten mittels einer zuvor trainierten Support-Vector-Machine klassifiziert und dabei Genauigkeiten von über 95% erreicht.

In [3] wird eine trainierte SVM benutzt um verschiedene urbane Objekte (z.B. Bus, Ampel) zu erkennen. Die LiDAR Daten wurden in einem städtischen Umfeld aufgenommen und die Objekte liegen in Form von Clustern vor. Für jedes Cluster werden verschiedene globale Deskriptoren berechnet, welche das ganze Objekt beschreiben, und dann von klassifiziert werden. Dabei werden verschiedene Featurevektoren benutzt Genauigkeiten von über 70% erzielt. Hier hat die SVM keine guten Ergebnisse erzielt und es ein weiterer Klassifikator hinzugefügt.

Rusu [14] hat mit einem globalen Deskriptor, dem Viewpoint-Feature-Histogramm, segmentierte Objekte erkannt und klassifiziert. Bei den Objekten handelte es sich um Kü-

chenware und Geschirrteilen, welche nur geringe Unterschiede aufweisten und trotzdem fast immer richtig erkannt wurden.

In [10] wurde gezeigt, dass es selbst mit einem schwenkbaren(für die z-Achse) 2D-LiDAR auf einem Roboter montiert, möglich ist, sehr genaue Karten zu erstellen und darin Cluster zu erkennen.

3 Systemdesign

3.1 Verwendete Bibliotheken

-Robot Operating System (ROS)
ROS ist ein Framework, das Dienste wie z.B. Hardwareabstraktion oder Gerätetreiber für Roboter zu Verfügung stellt. Es wird genutzt um den Sensor in Betrieb zu nehmen und die Daten aufzunehmen. [11]

-Point Cloud Library
Die Point Cloud Library (PCL) enthält viele Filter und Algorithmen für die Verarbeitung von 3D-Punktwolken. Diese Library wird genutzt für die Verarbeitung der Daten. [**?**]

-Visualization Toolkit
Das Visualization Toolkit ist in PCL integriert und wird zur Visualisierung der Ergebnisse genutzt. [16]

-LIBSVM
Die Library LIBSVM wird für die Support-Vector-Machine benutzt. [2]

3.2 Daten

Mit jedem Scan erzeugt der Sensor etwa 28900 Datenpunkte. Die Daten liegen in Form von unstrukturierten Punktwolken vor und enthalten die Punkte, die von dem Sensor gemessen wurden. In Abbildung 3.1 ist ein Exemplar von den zu erkennenden Objekten in 2D(a) und 3D(b) zu sehen.

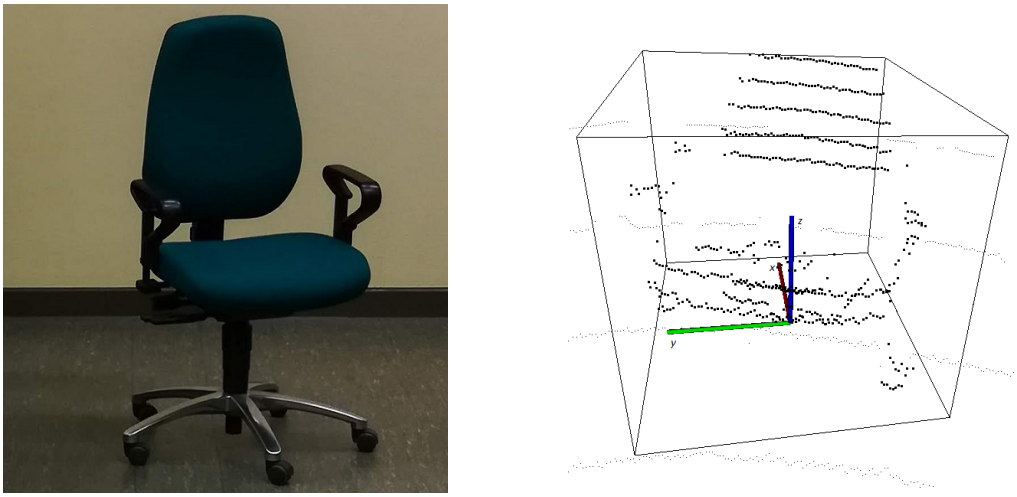

Abbildung 3.1: Repräsentation eines Stuhl-Objektes in 2D(links) und 3D(rechts)

3.3 Systemarchitektur

Die Verarbeitungs-Pipeline beschreibt die Systemarchitektur und den Datenfluss. Jeder Scan, also alle erzeugten Punkte der Punktwolke nach einer vollständigen Rotation des Sensors, durchläuft alle Schritte und liegt nach den Segmentierungs-Schritten in Form von Clustern vor. Für jedes Cluster werden Merkmalsvektoren extrahiert, welche dann durch einen Klassifikator mit einem Label versehen werden, der die jeweilige Klasse des Clusters identifiziert.

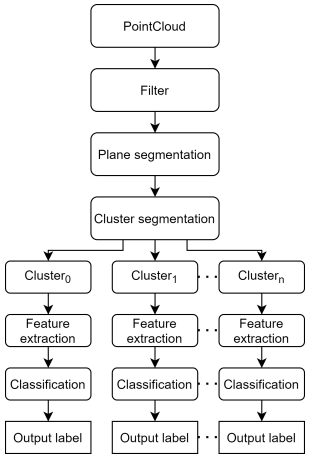

Abbildung 3.2: Verarbeitungspipeline

4 Experimente und Ergebnisse

Dieses Kapitel stellt die verwendeten Filter und Verfahren dar und zeigt Experimente und Ergebnisse mit den aufgenommenen Daten.

4.1 Vorverarbeitung und Filterung

Das Ziel der Vorverarbeitung ist uninteressante Punkte herauszufiltern ohne dabei Informationen über die Objekte zu verlieren. Es sind vor-verarbeitende Schritte (preprocessing) notwendig, die u.a. das Rauschen in den Daten verringern.

4.1.1 Statistischer Filter

Zunächst werden statistische Ausreißer entfernt. Ausreißer sind die Punkte, die weit vom nächst näheren Punkt entfernt liegen. Sie können mithilfe der durchschnittlichen Entfernung und dessen Standardabweichung σ aller Punkte zu ihrem jeweiligen k-ten Nachbar entfernt werden. Weicht ein Punkt mehr als α * σ davon ab, wird er als Ausreißer bezeichnet und wird von der Punktwolke entfernt. Mit dem Faktor α kann die gewünschte Dichte eingeschränkt werden [12] aber auch bei kleinem α die Gesamtzahl der Punkte und damit die Anzahl der Informationen reduzieren. Um so wenig Punkte wie möglich zu entfernen und alle wichtigen Informationen zu erhalten, wurde ein hohes α=2.6 bei k=10 Nachbarn.

4.1.2 Glättungsfilter

Mit dem von Alexa et. al [1] beschriebenen Verfahren kann das Rauschen innerhalb hoch aufgelösten 3D-Daten verringern. Das Verfahren verbessert die Position der Punkte, indem es die Position jedes Punktes an seine lokale Nachbarschaft anpasst. Dafür wird

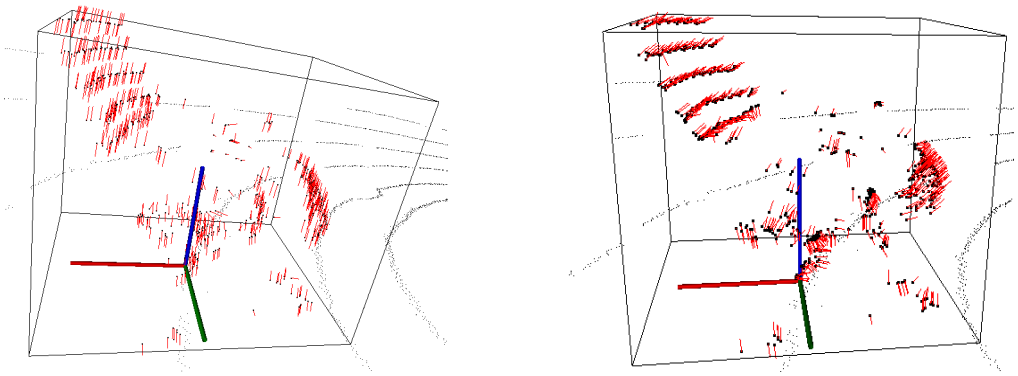

Abbildung 4.1: Vor dem Glättungsfilter (links) und nach dem Glättungsfilter (rechts) und die Normalvektoren in rot

für jeden Punkt ein Polynom n-ten Grades an die nächsten Nachbarn im Radius r angenähert, die Position des Punktes wird in Richtung des Polynoms transformiert und damit der Abstand des aktuellen Punktes zum Polynom reduziert. Dadurch werden die die Punkte hinsichtlich ihrer Nachbarn geglättet und damit das Rauschen reduziert. In Abbildung 4.1 ist ein Stuhl-Cluster abgebildet sowie sein zugehöriger Hüllkörper (axis-aligned bounding box) und die Normalenvektoren, welche aus den nächsten 5 Nachbarn berechnet wurden. Es ist zu erkennen, dass die Normalenvektoren vor Anwendung des Filters (links) fast alle in dieselbe (falsche) Richtung zeigen und nach Anwendung des Filters (rechts) bis auf wenige Ausnahmen in die gewünschte Richtung. Die falschen Normalenvektoren entstehen dadurch, dass durch das Rauschen drei oder mehr Punkte eine unerwünschte Ebene bilden und die Normalenberechnung damit beeinflussen und verfälschen. Da sich aus den Normalenvektoren Rückschlüsse auf die Krümmung der Oberfläche ziehen lassen und für die Merkmalsextraktion benötigt werden, haben sie direkten Einfluss auf die Klassifizierung.
Das Problem an diesem Filter entsteht dadurch, dass bei dichteren Stellen ein kleineren Radius zu verwenden ist, damit weniger Nachbarn für das Polynom zu berücksichtigen, damit keine Konturen verloren gehen und bei weniger dichten Stellen einen größeren Radius, damit korrekt gemessene Punkte nicht an den vom Rauschen beeinflussten Punkt angepasst werden. Ohne einen dichte abhängigen Parameter kann dieser Filter hier nicht verwendet werden.

4.2 Segmentierung

Beim Segmentieren geht es darum, die große Menge an Punkten in kleinere Mengen zu teilen. Hierbei wird ein Teil der Punkte, die nicht zu dem Objekt gehören, herausgefiltert. Das reduziert die Komplexität und den Rechenaufwand. In diesem Abschnitt wird die Vorgehensweise beim Segmentieren der Punktwolke gezeigt.

4.2.1 Filterung der Bodenebene

Da ein großer Teil der Punkte zum Boden des Raumes gehört und somit keine Informationen über die Objekte in dem Raum enthält, werden die zur Bodenebene gehörigen Punkte herausgefiltert. Dafür wurde RANSAC (*Ra*ndom *S*ample *C*onsensus, ein iterativer Algorithmus verwendet, welcher eine Schätzung der Ebenenparameter durchführt [6]. Es wird schrittweise versucht, eine Ebene in der Punktwolke auszurichten, sodass möglichst viele Punkte von dieser Ebene eingeschlossen werden. In jedem Schritt werden die Ebenenparameter neu berechnet und alle Punkte zu die dieser Ebene gezählt, dessen Distanz d zu dieser Ebene kleiner als ein Schwellwert e ist. Die Punkte, die nach der letzten Iteration zu der Ebene gehören, werden herausgefiltert. Hier wurden die Parameter e=0.01cm und die Anzahl der Iterationsschritte *500* gewählt. Dieser Filter kann nur angewendet werden, wenn die Bodenebene nicht gekrümmt ist und dann keine Ebene mehr darstellt. In Abbildung 4.2 sind die zur Bodenebene gehörigen Punkte markiert.

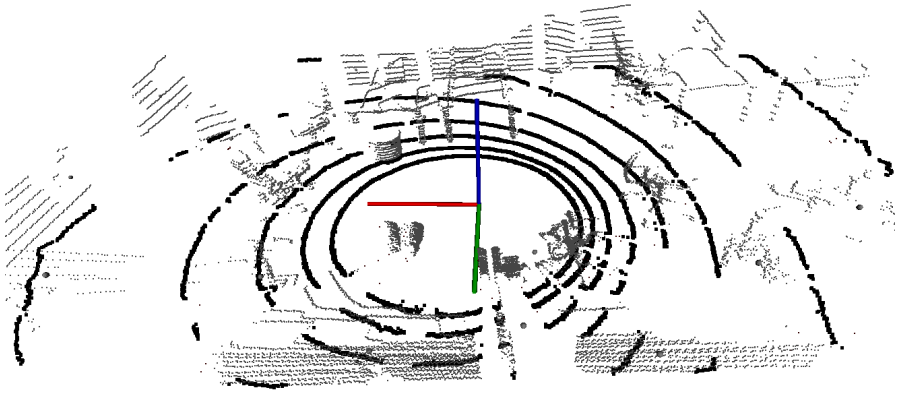

Abbildung 4.2: Mittels RANSAC ermittelte Bodenebene

4.2.2 Clustering

Beim Clustering geht es darum, Punkte mit ähnlichen Eigenschaften zu gruppieren und zu Untermengen von Punkten zusammenzufassen, die dann zusammen ein Cluster bilden. Dabei soll ein Cluster alle Punkte enthalten, die zu einem Objekt gehören und keine Punkte von anderen Objekten. Ähnliche Eigenschaften sind z.B. die Distanz zwischen zwei Punkten oder die Unterschiede in der Krümmung, gegeben durch die Winkel der Normalvektoren der beiden Punkte.

Dieser Verarbeitungsschritt ist von besonderer Bedeutung, da die Cluster als ganzes erkannt werden sollen und nicht zugehörige Punkte falsche Informationen liefern würden[12]. Das euklidische Clustering Verfahren bewertet die Ähnlichkeit zweier Punkte nur anhand des Abstandes zwischen ihnen. Ist der Abstand kleiner als ein Schwellwert e, werden beide Punkte zu einem Cluster hinzugefügt. Es werden solange ähnliche Punkte zu dem Cluster hinzugefügt, bis es keinen Punkt mehr gibt, der weniger als ein Abstand e zu allen Punkten in diesem Cluster entfernt ist. Genauso wird für die verbleibenden Punkte vorgegangen. Mit dem Parameter e=0.2cm kann dieser Algorithmus die Daten grob segmentieren.

Dieser Ansatz führte zur Über- und Untersegmentierung der Objekte, wie in Abbildung 4.4 für eine Übersegmentierung und in Abbildung 4.5 exemplarisch dargestellt ist.

Es wurde das in [4] beschriebene DBSCAN-Verfahren (*D*ensity-*B*ased *S*patial *C*lustering of *A*pplications with *N*oise) implementiert. Beim DBSCAN-Verfahren wird die Anzahl der Nachbarschaftspunkte innerhalb eines Radius r als Bewertungskriterium für die Ähnlichkeit genommen. Die *Eps*-Nachbarschaft eines Punktes beinhaltet alle Punkte innerhalb eines Radius r von diesem Punkt, und gibt damit die Dichte eines Punktes an. Ist die Anzahl der Punkte in *Eps* eines Punktes größer als eine Zahl MinPts, wird dieser Punkt als Kernpunkt eines Clusters gewertet. Alle direkten Nachbarn von dem Punkt und alle dichte-erreichbaren Punkte werden dem Cluster hinzugefügt. Zwei punkte sind dichte-erreichbar, wenn alle Punkte zwischen den beiden Punkten eine *Eps*-Nachbarschaft der Größe MinPts haben (also Kernpunkte sind) und alle Punkte zwischen den beiden Punkten auf einem Weg verbunden sind, bei dem der Abstand nie größer ist als r. Die Punkte mit einer *Eps*-Nachbarschaft kleiner als MinPts sind dann entweder Randpunkte eines Clusters oder Ausreißer. Der Parameter MinPts beschränkt die Cluster auf die Punkte, die mindestens MinPts in ihrer *Eps*-Nachbarschaft haben und der Parameter r gibt den Radius für die *Eps*-Nachbarschaft an.

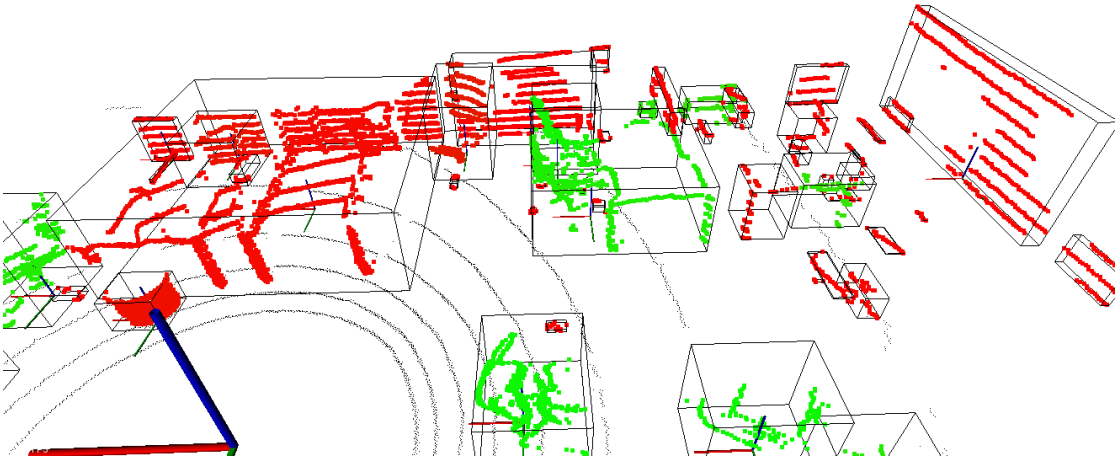

Abbildung 4.3: Segmentierungsergebnisse mit dem DBSCAN Verfahren

Dieses Verfahren hat mit den Parametern *MinPts*=10 und r=0.15cm zwar bessere Clustering Ergebnisse erzielt als das euklidische und konnte an manchen Bereichen eine Untersegmentierung verhindern, jedoch führt die Wahl eines konstanten r nicht zu den erwünschten Ergebnissen. Die Ergebnisse sind in Abbildung 4.3 abgebildet. Da die Dichte (Anzahl der Punkte in einem Radius r) innerhalb der Daten mit größerer Entfernung zum Sensor abnimmt und damit der Abstand zwischen zwei Punkten zunimmt, entstehen häufig in der Nähe des Sensors wenige, sehr große Cluster und weiter entfernt entstehen viele kleine Cluster. Es muss ein Verfahren gefunden werden, das die Punkte abhängig von ihrer Dichte segmentiert.
Der GDBSCAN (Generalized-DBSCAN) [15] adressiert dieses Problem und erweitert den DBSCAN Algorithmus, indem die Anzahl der Punkte in der *Eps*-Nachbarschaft keinen Schwellwert mehr darstellen, sondern durch Gewichtung Einfluss auf die Größe der *Eps*-Nachbarschaft eines Punktes nehmen und so Cluster mit einem großen durchschnittlichen Abstand von Punkten sowie Cluster mit einem kleinen durchschnittlichen Abstand erstellt werden.

Fan et. al(2018)[5] haben einen selbst-adaptiven Algorithmus für eine Cluster Segmentierung von hoch aufgelöste Punktwolken vorgeschlagen, der den Radius r für die Berechnung der *Eps*-Nachbarschaft für jeden Punkt neu berechnet. Dafür bewerten sie die Ähnlichkeit von Punkten anhand des Abstandes des aktuellen Punktes zur Mitte sowie anhand des Unterschiedes in der Krümmung zu seinen Nachbarn. Die Mitte ist dabei

der Punkt mit dem niedrigsten Abstand zu allen anderen Punkten. Durch die richtige Gewichtung der beiden Größen können sehr kleine Änderungen in der Krümmung der Oberfläche einen großen Abstand zum nächsten näheren Punkt ausgleichen und somit alle z.B. alle übersegmentierten nebeneinander liegenden Wandsegmente zu einem Cluster hinzugefügt werden, auch wenn sie entfernter voneinander positioniert sind. Genauso könnte an den untersegmentierten, sehr dichten Stellen nahe des Sensors große Veränderungen in der Krümmung den sehr geringen Abstand ausgleichen und damit z.B. Objekte einem Tisch separiert werden.

Um fortzufahren wird auf eine individuelle, nicht allgemeingültige Lösung zurückgegriffen, um mit der Segmentierung bessere Ergebnisse zu erzielen. Die Lösung muss aber zukünftig durch Implementierung eines Clustering Verfahrens wie z.B. das GDBSCAN Verfahren ersetzt werden.

Dafür wird zunächst wird das euklidische Clustering Verfahren mit einem konstanten e verwendet, um bewusst Untersegmentierungen zu erzeugen, welche dann durch das DBSCAN Verfahren mit einem von der Clusterdichte abhängiges *Eps* aufgelöst werden sollen. Für jedes erstellte Cluster werden die Distanzen der Punkte zu ihrem 4. Nachbar berechnet und durch Sortieren der Werte zeigen sich die am dichtesten und die am wenigsten dichten Punkte in diesem Cluster, gemessen an dem Abstand zu ihrem 4. Nachbar, wie in [4] beschrieben. In Abbildung 4.6 ist diese Dichtefunktion für vier Cluster unterschiedlicher Größe dargestellt. Es ist zu erkennen, dass die meisten Punkte innerhalb eines Clusters eine vergleichsweise kleine Distanz zum 4. Nachbar aufweisen und damit dichte-erreichbar sind, und einige wenige Punkte eine vergleichsweise sehr große Distanz. Die Punkte mit der größten Distanz sind meistens entweder Randpunkte oder ungewollte, nicht zugehörige Punkte, da sie stark von dem Durchschnitt der Distanzen abweichen und häufig zwei einzelne Cluster verbinden und damit für eine Untersegmentierung verantwortlich sind. Für jedes durch das euklidische Clustering Verfahren erstellte Cluster wird ein neues e, abhängig von der Dichte zugewiesen und dem DBSCAN-Verfahren übergeben. Damit konnten bei den meisten Clustern nahe des Sensors eine Untersegmentierung durch kleineres e vermieden werden und weiter entfernte Cluster durch durch größeres e trotz ihrer geringen Dichte segmentiert werden. Auch die Klassifizierung konnte verbessert werden, das aber auf Kosten eines hohen Aufwands durch zweifaches Clustern der Daten und soll zukünftig durch das vorgeschlagene GDBSCAN [15] Verfahren ersetzt werden.

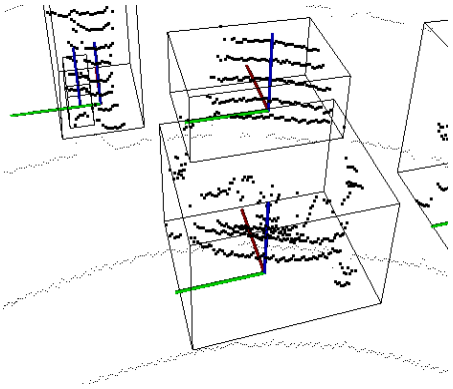

Abbildung 4.4: Durch Lücken verursachte Übersegmentierung

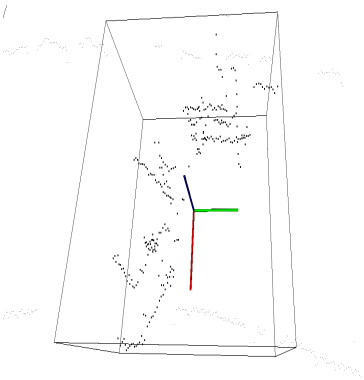

Abbildung 4.5: Untersegmentierung von zwei Clustern

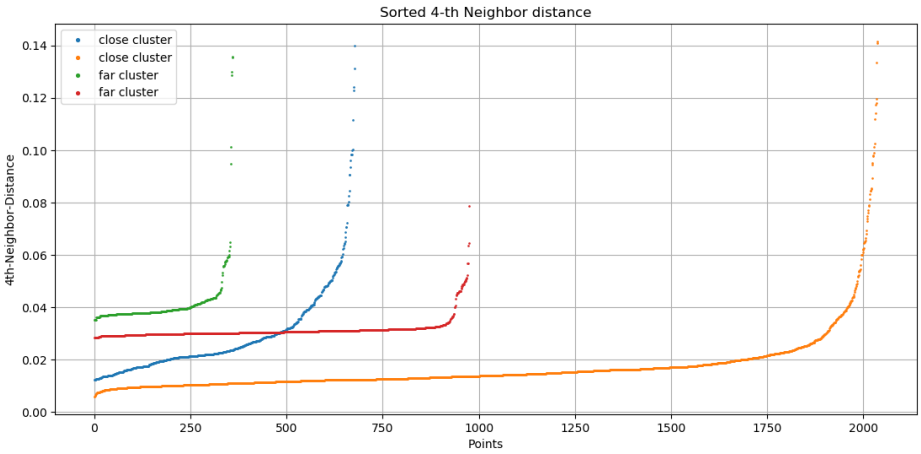

Abbildung 4.6: Sortierte Distanz zum 4. Nachbar aller Punkte eines Clusters

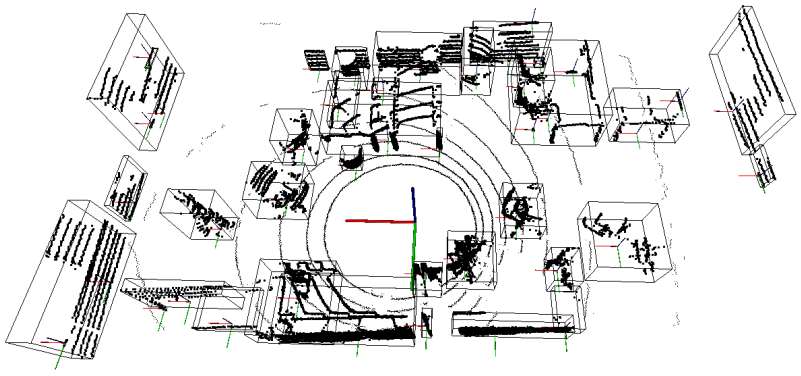

Abbildung 4.7: Segmentierte Punktwolke

In Abbildung 4.7 sind die Clustering Ergebnisse dargestellt. Es ist zu erkennen, dass die Cluster in der Nähe des Sensors zumeist richtig segmentiert wurden, selbst wenn nur wenige Zentimeter voneinander entfernt waren. Außerdem wurden größere, zusammenhängende Wandsegmente durch das größere e richtig als ganzes Cluster segmentiert. Des weiteren sind einige untersegmentierte Cluster zu sehen, welche hauptsächlich durch Lücken in den Daten entstanden sind. Die Lücken in den Daten sind Freiräume, die entstehen, wenn Teile des Objekts durch andere Objekte (oder durch das Objekt selbst) überdeckt sind und so nicht wahrgenommen werden konnten. Häufig wurden die Lehne des Stuhls vom Cluster segmentiert, weil sie durch die Lücken einen zu großen Abstand aufwiesen.

4.3 Features

Von jedem erstellten Cluster werden Merkmale (Features) extrahiert, welche möglichst kompakt aber präzise das Objekt repräsentieren sollen. Die Features sollten bestmöglich rotations- und dichte- und skaleninvariant sein, um weiter entfernte oder rotierte Cluster genauso richtig erkannt werden, wie Cluster nahe des Sensors. Außerdem sollten sie schnell berechnet werden können und gleichzeitig so viele Informationen enthalten wie möglich. Es werden drei verschiedene Feature-Deskriptoren vorgestellt und kurz erläutert, welche Informationen sie enthalten.

In Abbildung 4.8 ist ein Stuhl-Cluster sowie alle drei berechneten Feature-Histogramme dargestellt.

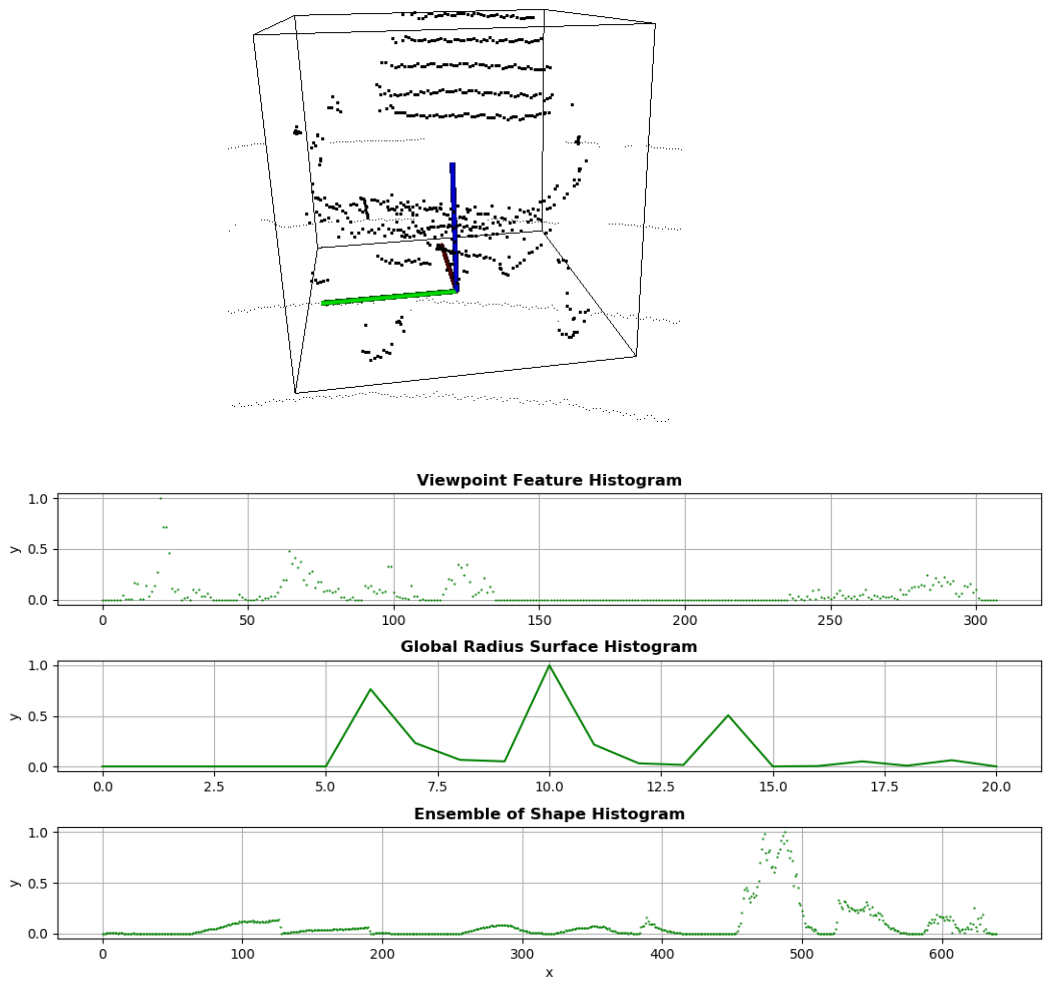

Abbildung 4.8: Stuhl-Cluster und dazugehörige Feature-Vektoren

4.3.1 Viewpoint Feature Histogram

Das Viewpoint Feature Histogram (VFH) [14] baut auf dem lokalen Fast Point Feature Histogram (FPFH)[13] auf, welches eine Erweiterung des Point Feature Histogrammes (PFH)[12] ist. Das Point Feature Histogramm fasst Veränderungen in der lokalen Umgebung eines Punktes in einem Histogramm zusammen, indem die Unterschiede der Normalenvektoren von dem Punkt zu denen seiner Nachbarn berechnet werden. Dazu wird für einen Punkt mit jedem Punkt in seiner lokalen Nachbarschaft und zwischen diesen ein Vektor gebildet und drei Winkel berechnet: Der erste Winkel befindet sich zwischen dem Normalenvektor des ersten Puntes und dem aufgespannten Vektor, der zweite Winkel zwischen dem Normalenvektor des zweiten Punktes und dem Normalenvektor des aufgespannten Vektors und der dritte lässt sich mit den Normalenvektoren der beiden Punkte berechnen. Zusätzlich wird der Abstand der beiden Punkte berechnet und dadurch ergeben sich vier Werte für jedes Punktpaar in der Nachbarschaft eines Punktes. Da bei k Nachbarn diese vier Werte $\frac{k-1}{2}$ mal berechnet werden, ist es bei größeren Punktwolken sehr zeitaufwendig und häufig ein Engpass in Echtzeit Systemen. Das FPFH optimiert das PFH indem es nur die direkten Verbindungen zwischen dem aktuellen Punkt und seinen Nachbarn berücksichtigt, wodurch zusätzliche Verbindungen zwischen Nachbarn entfernt werden. Dies reduziert die Komplexität von O(n*k^2)auf O(n*k) für eine Punktwolke mit n Punkten. Die berechneten Werte werden anhand ihrer Häufigkeit in einem Histogramm zusammengefasst.
Das VFH erweitert das FPFH um eine Blickrichtungskomponente. Dafür wird bei der Berechnung einen Blickrichtungsvektor hinzugefügt, der vom Sensor in Richtung des Mittelpunktes des Clusters zeigt. Die Winkel werden zwischen den Normalen der Punkte und dem Blickrichtungsvektor gebildet und in einem Histogramm zusammengefasst. Dadurch wird zusätzlich zu den Veränderungen in der in Oberfläche die Ausrichtung, also die Lagewinkel der Cluster in dem Histogramm codiert und kann bei erfolgreicher Klassifizierung für eine Posenschätzung verwendet werden, da die Berechnung des FPFH unabhängig von der Rotation und Größe des Clusters ist[14]. Das VFH hat schon in [3] und [14] gezeigt, dass es in dichten Punktwolken Objekte anhand ihrer Eigenschaften in der Oberfläche unterscheidbar macht. Hier wird das VFH mit den den Normalvektoren berechnet aus den jeweil 15 nächsten Nachbarspunkten berechnet.

4.3.2 Ensemble of Shape Functions Histogramm

Das Ensemble of Shape Functions Histogam(ESFH) fasst charakteristische Eigenschaften des Clusters in einem Histogramm zusammen[19]. Die Eigenschaften beinhalten Distanz-, Winkel- und Flächenverteilungen innerhalb des Clusters und werden in einem Histogramm der Länge 640 gespeichert. Das Histogramm setzt sich aus zehn kleineren Histogrammen mit jeweils 64 Werten zusammen. Die Werte werden berechnet indem zufällige Punkte des Clusters ausgewählt werden und Vektoren zwischen jeweils zwei dieser Punkte berechnet werden. Die Winkel zwischen den Vektoren werden berechnet und abhängig davon, ob die Vektoren auf der Oberfläche des Clustes liegen, außerhalb oder beides, werden die Winkel entsprechend in das erste, zweite oder dritte Histogramm gespeichert. Die ersten drei Histogramme enthalten die Verteilung der Winkel, die nächsten drei die Verteilung der Flächen und weitere drei die Verteilung der Distanzen in dem Cluster. Das letzte Histogramm enthält das Verhältnis der Distanzen.

4.3.3 GRSD

Der global radius-based surface describer(GRSD)[9] ist ein globales Histogramm der Länge 21, das sich aus dem lokalen radius-based surface describer (RSD)[8] zusammensetzt. Der RSD beschreibt, welche geometrische Formen(z.B. Ebene, Kugel, Zylinder, Rand) in dem radialen Umkreis eines Punktes gefunden wurde. Jeder Punkt wird durch eine Form klassifiziert, indem der minimale und maximale Radius für das anhand der Nachbarschaft angenäherte Polynom berechnet wird. Es werden jetzt geometrische Formen an das Polynom gepasst und bewertet. Der GRSD summiert die Werte für die gefundenen Formen für alle Punkte in dem Cluster. Als Parameter wird der Radius r = 0.25cm verwendet.

4.4 Klassifizierung und Training

4.4.1 Training

Um die SVM zu trainieren, werden die Daten in Trainings- und Testscans aufgeteilt. Die ersten zehn Scan sind die Trainingsscans und die restlichen Scans sind Testscans. Aus den Trainingsscans werden die Stuhl-Cluster, die richtig segmentiert wurden, zusammen mit einem Label, das die Stuhl-Klasse identifiziert, und die zugehörigen Feature-Histogramme

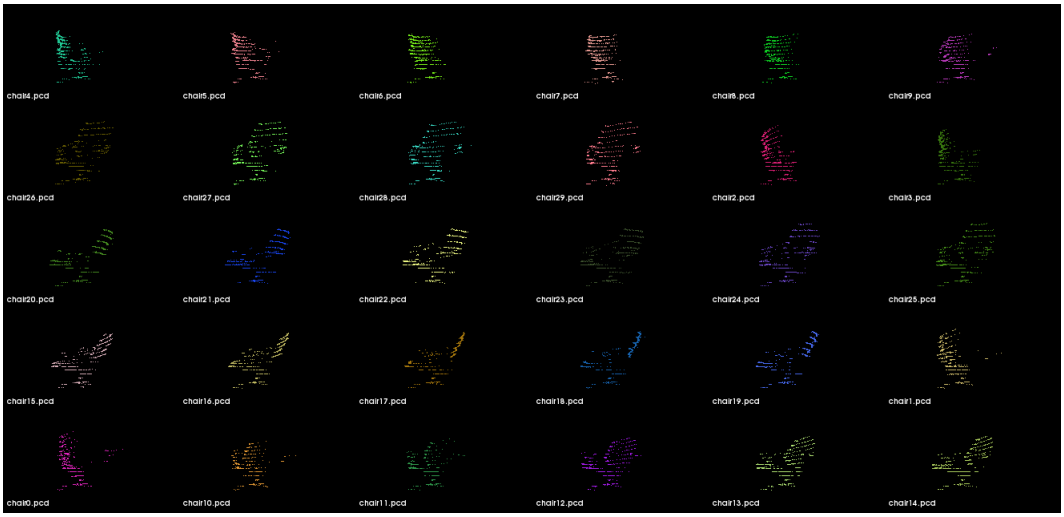

Abbildung 4.9: 3D-Modelle von Trainingsdaten

als Trainingsdaten in einer Datenbank gespeichert. Genauso wird für die Cluster vorgegangen, die keine Stuhl-Punkte enthalten. Um möglichst viele verschiedene Perspektiven auf die zu erkennenden Objekte zu erhalten, wurde bei der Datenaufnahme ein Stuhl um seine z-Achse rotiert und den Trainingsdaten hinzugefügt. Um eine Überanpassung zu vermeiden, enthalten die Trainingsdaten dichte und weniger dichte Cluster von den Stuhl-Objekten, sowie möglichst viele verschiedene Cluster der nicht-Stuhl-Objekte. In Abbildung 4.9 sind die positiven Exemplare der Trainingsdaten abgebildet.

Die Datenbank enthält 60 positive Exemplare und 120 negative.

In Abbildung 4.10 sind für die ersten zehn positiven und negativen Exemplare der Datenbank die drei normierten Feature-Vektoren dargestellt.

4.4.2 Klassifikationsverfahren

Support Vector Machines (SVMs) [2] ist ein überwachtes Verfahren, welches zur Klassifizierung angewendet wird. Durch Trainingsdaten mit bekannter zugehöriger Klasse wird eine Entscheidungsfunktion (decision function) erstellt. Die Entscheidungsfunktion trennt im linearen Fall durch eine Gerade(2D) oder Ebene(3D) die Klassen voneinander, wobei der Abstand der Ebene zu den nächsten Trainingsdaten der Klassen maximiert wird. Beim nicht-linearen Fall, der durch die hochrangigen Feature-Vektoren vorliegt,

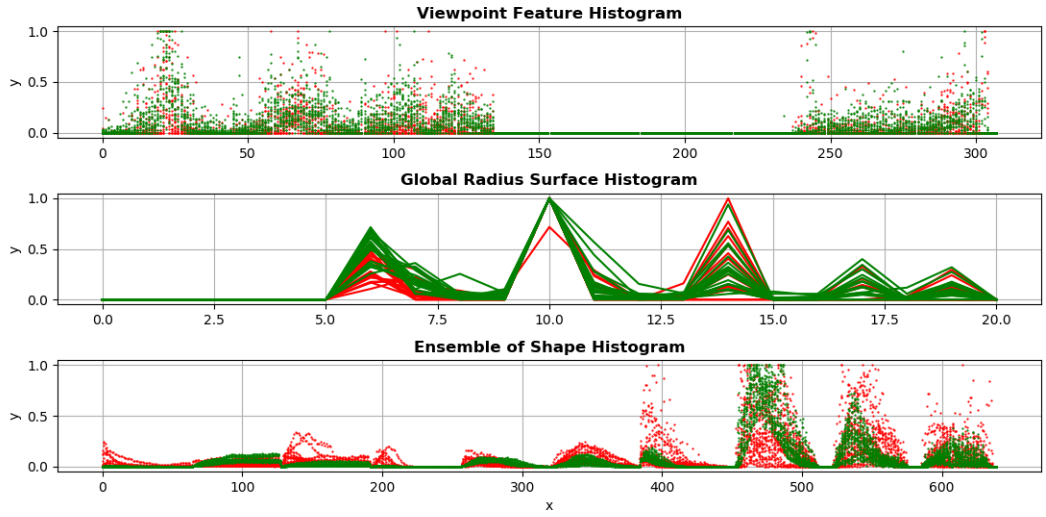

Abbildung 4.10: Feature-Vektoren für 10 positive (grün) und 10 negative (rot) Exemplare

wird die Ebene durch eine Hyperebene ersetzt, die dann die Entscheidungsfunktion darstellt. Durch diese Entscheidungsfunktion werden neue, der SVM unbekannte Datensätze einer Klasse (Stuhl und nicht-Stuhl) zugeordnet. Es kann eine Kern-Funktion (kernel) spezifiziert werden, um die Entscheidungsfunktion und die daraus folgende Klassifizierung so präzise wie möglich zu gestalten. Je nach Anwendung kann die Kern-Funktion linear, polynomial, eine Radiale-Basisfunktion (RBF), eine Sigmoidfunktion oder auch selbst definiert sein.
Wie erwartet stellte sich heraus, dass mit einer RBF als Kern die besten Klassifikationsergebnisse erzielt wurden. Die SVM wird mit den Werten C und γ parametrisiert und berechnet daraus eine Entscheidungsfunktion. C gewichtet dabei die Toleranzgrenze um einer Klasse zugewiesen zu werden und γ beschreibt die Krümmung der Entscheidungsfunktion.

Die RBF wird im 5 Kapitel mit verschiedenen Parametern von C und γ getestet und die SVM wird mittels Kreuzvalidierung auf ihre Genauigkeit untersucht.

5 Evaluation

In diesem Kapitel befinden sich die Ergebnisse und die Evaluation dieser.

5.1 Kreuzvalidierung

Die SVM wird durch Kreuzvalidierung der Testdaten auf ihre Präzision untersucht. Dafür teilt die SVM die Trainingsdaten in Trainings- und Testdaten auf, erstellt eine Entscheidungsregel und validiert diese mit den Testdaten. Danach werden die falsch-positiven, falsch-negativen, richtig-positiven und richtig negativen Exemplare gezählt und die Präzision bestimmt. Die SVMs werden mit verschiedenen C und γ getestet und mit den Parametern erstellt, bei der die höchsten Präzisionen erreicht wurden.

C \ γ	10^{-5}	10^{-4}	10^{-3}	10^{-2}
10^{-2}	0.61	0.61	0.61	0.51
10^{-1}	0.61	0.61	0.61	0.51
10^{0}	0.62	0.61	0.64	0.51
10^{1}	0.61	0.61	0.64	0.51
10^{2}	0.61	0.63	0.64	0.51
10^{3}	0.60	0.65	0.64	0.51
10^{4}	0.56	0.59	0.64	0.51

Tabelle 5.1: VFH-SVM Kreuzvalidierung

Die Tabellen 5.1, 5.2 und 5.3 zeigen die Ergebnisse der Kreuzvalidierung der Trainingsdaten mit den jeweiligen Feature-Vektoren. Die Parameter, bei der die höchste Präzision erzielt wurde, werden der Tabelle entnommen und die SVM mit jenen erstellt. Für die Instantiierung der SVM für das VFH wurden die Kern-Parameter C=10^3 und γ=10^{-4}

C \ γ	10^{-5}	10^{-4}	10^{-3}	10^{-2}
10^{-2}	0.62	0.61	0.61	0.51
10^{-1}	0.61	0.61	0.61	0.51
10^{0}	0.63	0.61	0.63	0.51
10^{1}	0.61	0.61	0.63	0.51
10^{2}	0.61	0.63	0.60	0.51
10^{3}	0.60	0.63	0.60	0.51
10^{4}	0.56	0.59	0.60	0.51

Tabelle 5.2: GRSD-SVM Kreuzvalidierung

C \ γ	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}	10^{4}
10^{-2}	0.54	0.53	0.54	0.56	0.61	0.45
10^{-1}	0.47	0.62	0.44	0.53	0.49	0.48
10^{0}	0.41	0.54	0.54	0.48	0.63	0.51
10^{1}	0.60	0.44	0.61	0.62	0.60	0.51
10^{2}	0.52	0.56	0.60	0.60	0.59	0.46
10^{3}	0.48	0.59	0.58	0.57	0.59	0.43
10^{4}	0.58	0.61	0.60	0.57	0.57	0.51

Tabelle 5.3: ESF-SVM Kreuzvalidierung

gewählt, für das GRSD die Parameter C=10^0 und γ=10^3 und für das ESF die Parameter C=10^2 und γ=10^{-4}.

5.2 Klassifikationsergebnisse

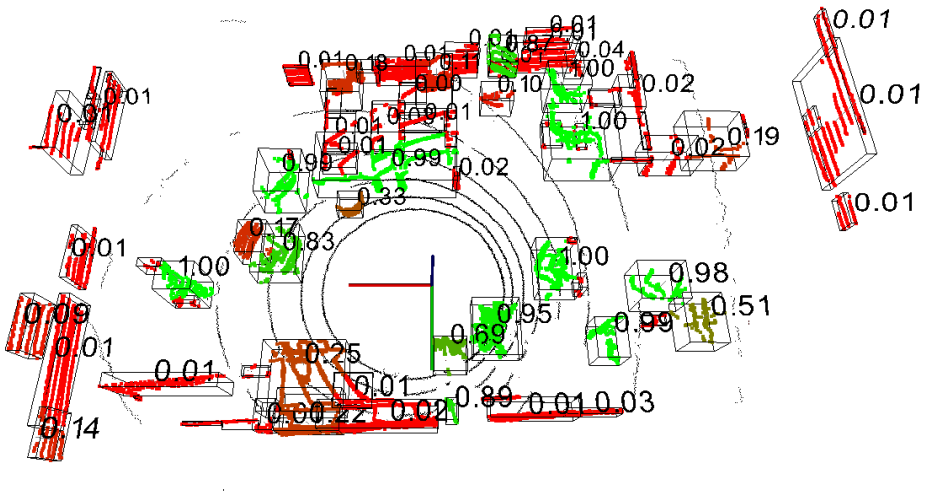

Abbildung 5.1: VFH Klassifikationsergebnisse

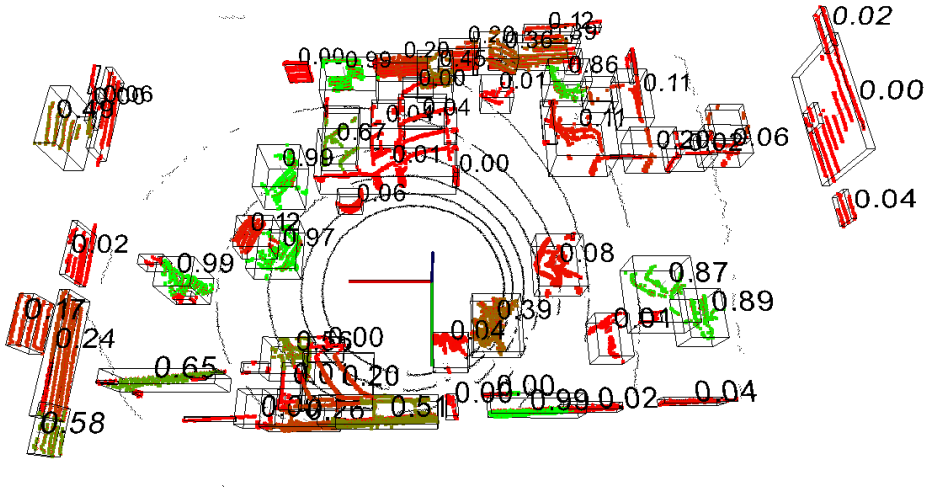

Abbildung 5.2: GRSD Klassifikationsergebnisse

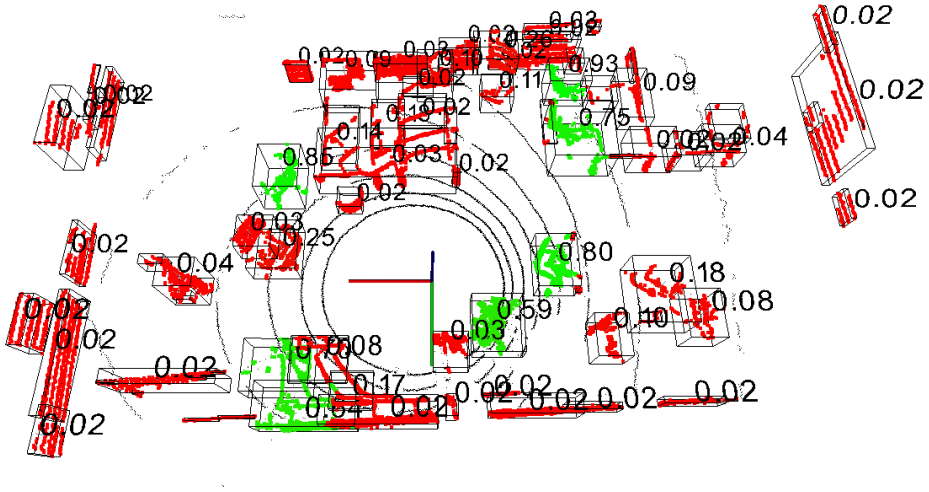

Abbildung 5.3: ESF Klassifikationsergebnisse

5.3 Diskussion

Die höchste Präzision gemessen an der Kreuzvalidierung wurde mit der SVM für das VFH erreicht und genau wie bei der SVM für das ESF ein γ von 10^{-4} gefunden. Bei der SVM für den GRSD wurde ein γ von 10^{3} gefunden. Hier könnte eine Überanpassung der Entscheidungsfunktion an die Trainingsdaten vorliegen, was mit dem Featureberechnung und der Dichte der Cluster zusammenhängen könnte. Da bei der Extraktion des GRSD für den Radius r ein konstanter Wert benutzt wurde und damit nicht dichteunabhängig ist, könnte sich jetzt das Problem der unterschiedlichen Dichten zeigen. Hier müsste genauso wie für die Segmentierung oder den Glättungsfilter ein Wert gefunden werden, der abhängig von der Dichte des Clusters ist, damit dieser Feature-Vektor für eine globale Objekterkennung geeignet ist. In den Abbildungen 5.1, 5.2 und 5.3 sind die Ergebnisse der Segmentierung und der Klassifikation für die Merkmals-Vektoren VFH, GRSD und ESF zu sehen. Zu erkennen sind die Cluster mit ihren Hüllkörpern, sowie die Klasse der Cluster beschrieben durch die Farbe (grün für Stuhl, rot für nicht-Stuhl) und die Wahrscheinlichkeit der errechneten Klasse über den Clustern in schwarz. In den meisten Fällen wurden die nicht-Stuhl Clustern wie z.B. die Wandsegmente als solche erkannt und einige Stuhlcluster richtig klassifiziert. Schwierig wird es für die SVM zwischen Stuhlobjekten und anderen Objekten mit vielen Veränderungen auf der Oberfläche zu unterscheiden

und resultiert in einigen falsch positiven Klassifikationen. In Abbildung 5.1 ist z.B. zu erkennen, dass Teile von einem Tisch als Stuhl klassifiziert wurden. Das lässt sich zum einen auf eine Untersegmentierung beim Clustering zurückführen, da der Tisch nicht als ganzes Objekt segmentiert wurde und damit nur zum Teil die Oberfläche des Tisches beschreibt und zum anderen auf eine nicht perfekte Normalenberechnung aufgrund der unterschiedlichen Dichten in den Daten. Außerdem unterscheidet die SVM nur zwischen den Klassen Stuhl und nicht-Stuhl, was durch das Hinzufügen der Klassen von z.B. Tisch, Wand oder Papierkorb und einer Multi-class SVM erweitert werden kann.

6 Fazit

6.1 Zusammenfassung

In dieser Arbeit wurde gezeigt, wie mit einer guten Vorverarbeitung und Segmentierung Objekte innerhalb eines Scans anhand ihrer charakteristischen Merkmale klassifiziert werden können, trotz einer großen Anzahl an Lücken und der Existenz von starkem Rauschen. Dabei konnte das Viewpoint-Feature-Histogramm mit 65% am besten klassifiziert werden. Es wurde gezeigt, welche Filter abhängig von der Punktdichte sind und warum eine gute Segmentierung der Daten durch die unterschiedlichen Dichten keine leichte Aufgabe darstellt. Cluster mit vielen Punkten und damit vielen enthaltenen Informationen konnten dabei besser erkannt werden als kleine Cluster, mit wenigen Informationen.

6.2 Ausblick

Eine präzise Klassifizierung von segmentierten Objekten in lückenhaften LiDAR Daten ist stark beeinflusst von einer guten Vorverarbeitung und Segmentierung. Das Clustering kann in zukünftigen Arbeiten mithilfe des GDBSCAN Verfahren[15] zu einem dichte unabhängigen Verfahren verbessert. Mit genügend Trainingsdaten und der Wahl von dynamisch berechneten Parametern für Filter- und Clustering Verfahren kann die Klassifizierung verbessert werden. Außerdem kann mithilfe des VFH eine Posenschätzung vorgenommen werden und damit zusätzlich zur Position die Lagewinkel der Cluster berechnet werden.

Literaturverzeichnis

[1] Alexa, Marc ; Behr, Johannes ; Cohen-Or, Daniel ; Fleishman, Shachar ; Levin, David ; Silva, Claudio T.: Computing and Rendering Point Set Surfaces. In: *IEEE Transactions on visualization and computer graphics* 9 (2003), Nr. 1, S. 3–15. – URL 10.1109/TVCG.2003.1175093

[2] Chang, Chih-Chung ; Lin, Chih-Jen: LIBSVM: A library for support vector machines. In: *ACM transactions on intelligent systems and technology (TIST)* 2 (2011), Nr. 3, S. 1–27. – URL https://doi.org/10.1145/1961189.1961199

[3] Chen, Tongtong ; Dai, Bin ; Liu, Daxue ; Song, Jinze: Performance of Global Descriptors for Velodyne-based Urban Object Recognition. (2014). – URL 10.1109/IVS.2014.6856425

[4] Ester, Martin ; Kriegel, Hans-Peter ; Sander, Jiirg ; Xu, Xiaowei: A density-based algorithm for discovering clusters in large spatial databases with noise. 96 (1996), Nr. 34, S. 226–231. – URL https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

[5] Fan, Yuling ; Wang, Meili ; Geng, Nan ; He, Dongjian ; Chang, Jian ; Zhang, Jian: A self-adaptive segmentation method for a point cloud. In: *Visual Computer* 34 (2018), 05, S. 1–15. – URL 10.1007/s00371-017-1405-6

[6] Fischler, Martin A. ; Bolles, Robert C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In: *Commun. ACM* 24 (1981), Nr. 6, S. 381–395. – URL https://doi.org/10.1145/358669.358692. – ISSN 0001-0782

[7] Lodha, Suresh K. ; Kreps, Edward J. ; Helmbold, David P. ; Fitzpatrick, Darren: *Aerial LiDAR Data Classification using Support Vector Machines (SVM)*. URL 10.1109/3DPVT.2006.23, 2006. – 567–574 S

[8] Marton, Z. ; Pangercic, D. ; Blodow, N. ; Kleinehellefort, J. ; Beetz, M.: General 3D modelling of novel objects from a single view. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, URL 10.1109/IROS.2010.5650434, 2010, S. 3700–3705

[9] Marton, Z. ; Pangercic, D. ; Rusu, R. B. ; Holzbach, A. ; Beetz, M.: Hierarchical object geometric categorization and appearance classification for mobile manipulation. In: *2010 10th IEEE-RAS International Conference on Humanoid Robots*, URL 10.1109/ICHR.2010.5686323, 2010, S. 365–370

[10] Neuta, N. L. ; Vivas, S. A. ; Fajardo, N. V. ; Giraldo, O. F. R. ; Cano, V. R.: Low-cost recognition and classification system based on LIDAR sensors. In: *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, URL 10.1109/CCRA.2018.8588139, 2018, S. 1–6

[11] Quigley, Morgan ; Conley, Ken ; Gerkey, Brian ; Faust, Josh ; Foote, Tully ; Leibs, Jeremy ; Wheeler, Rob ; Ng, Andrew Y.: ROS: an open-source Robot Operating System. In: *ICRA workshop on open source software* Kobe, Japan (Veranst.), 2009, S. 5

[12] Rusu, R. B.: Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. (2009). – URL 10.1007/s13218-010-0059-6

[13] Rusu, R. B. ; Blodow, N. ; Beetz, M.: Fast Point Feature Histograms (FPFH) for 3D registration. In: *IEEE International Conference on Robotics and Automation*, URL 10.1109/ROBOT.2009.5152473, 2009, S. 3212–3217

[14] Rusu, R. B. ; Bradski, G. ; Thibaux, R. ; Hsu, J.: *Fast 3D recognition and pose using the Viewpoint Feature Histogram*. 2010. – URL 10.1109/IROS.2010.5651280

[15] Sander, Jörg ; Ester, Martin ; Kriegel, Hans-Peter ; Xu, Xiaowei: Density-based clustering in spatial databases: The algorithm gdbscan and its applications. In: *Data mining and knowledge discovery* 2 (1998), Nr. 2, S. 169–194. – URL https://doi.org/10.1023/A:1009745219419

[16] Schroeder, Will ; Martin, Ken ; Lorensen, Bill: The Visualization Toolkit An Object-Oriented Approach To 3D Graphics Fourth Edition. (2006). – URL https://doi.org/10.1016/j.softx.2015.04.001

[17] Velodyne (Veranst.): *VLP-16 User Manual*. 2018. – URL https://greenvalleyintl.com/wp-content/uploads/2019/02/Velodyne-LiDAR-VLP-16-User-Manual.pdf

[18] Weitkamp, C.: *Lidar*. Springer, 2005. – 3700–3705 S. – URL 10.1007/b106786

[19] Wohlkinger, Walter ; Vincze, Markus: *Ensemble of shape functions for 3D object classification*. URL 10.1109/ROBIO.2011.6181760, 2011. – 2987–2992 S

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

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

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

Objekterkennung in 3D-Lidardaten

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.

_____	_____	_____
Ort	Datum	Unterschrift im Original