Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik VI

Diplomarbeit

Extraktion geometrischer Formprimitive mit Textur- und Reliefkarten aus 3D Punktwolken mit Farbinformation

---

Bastian Oehler

Erstgutachter: Prof. Dr. Sven Behnke
Zweitgutachter: Prof. Dr. Armin B. Cremers

Betreuer: Dipl.-Inform. Jörg Stückler und Dipl.-Inform. Jochen Welle

Erklärung

Hiermit versichere ich, dass ich die hier vorliegende Diplomarbeit selbstständig angefertigt, keine anderen, als die angegebenen Hilfsmittel verwendet und Zitate kenntlich gemacht habe.

Bonn, den

_____

(Bastian Oehler)

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit wird zu Beginn dieses Kapitels durch zwei Anwendungsbeispiele motiviert. Anschließend werden die Ziele und die Herangehensweise erläutert, bevor im nächsten Abschnitt verwandte Arbeiten betrachtet werden. Am Ende des Kapitels steht ein kurzer Überblick über den weiteren Aufbau der Arbeit.

1.1 Motivation

Das Erkennen und Manipulieren von Gegenständen ist ein zentrales Forschungsgebiet in der Robotik. Roboter werden mit verschiedenen Sensoren ausgestattet, die eine Wahrnehmung der Umgebung ermöglichen. Dazu gehören 3D Scanner, wie die Microsoft Kinect, und schwenkbare 2D Laserentfernungsmesser. Die Sensoren liefern als Ausgabe 3D Punktwolken, im Falle der Kinect mit Farbinformationen. In den Daten muss dann die Objekterkennung gelöst werden. Das in dieser Arbeit entwickelte Verfahren kann dazu verwendet werden, die Suche der Objekte zu vereinfachen. Aus der 3D Punktwolke werden zylindrische und planare Segmente extrahiert und als alternative Darstellung der Szene genutzt. Die planare Segmentierung, die im Zuge dieser Arbeit entwickelt wurde, wurde bereits auf der International Conference on Intelligent Robotics and Applications 2011 als Veröffentlichung [10] akzeptiert.

Bei der Beschleunigung der Objekterkennung könnte das in dieser Arbeit entwickelte Verfahren verwendet werden, um die Sensorendaten in Form einer Punktwolke durch Formprimitive angenähert darzustellen. Ein einfaches Objekt könnte als texturiertes und mit Reliefkarten versehenes Formprimitiv repräsentiert werden. Um das Objekt in der segmentierten Szene zu finden, könnte ein Primitiv von gleichen Abmessungen

gesucht werden. Zusätzlich können die 2D Texturinformation der Formprimitive miteinander verglichen werden um Mehrdeutigkeiten aufzulösen. Dabei können gängige 2D Bildverarbeitungsverfahren angewendet werden.

Einen weiteren Anwendungsfall für die Segmentierung gibt es in der Teleoperation von teilautonomen Robotern. Hier kommt es vor allem darauf an, dem Bediener des Roboters zeitnah eine möglichst exakte Visualisierung der Roboterumgebung zu präsentieren. Um dem Bediener eine größtmögliche Übersicht zu gewährleisten, ist es üblich, die Szene mit einem 3D Scanner abzutasten. Da eine 3D Punktwolke für den ungeübten Betrachter allerdings schwierig zu interpretieren ist, bietet sich eine Repräsentation durch texturierte Formprimitive zur übersichtlicheren Darstellung an. Zusätzlich ist die Datenübertragung bei der Teleoperation nicht immer mit großer Bandbreite möglich. Daher ist eine Komprimierung der Eingabepunktwolke von Nutzen. Das in dieser Arbeit vorgeschlagene Verfahren kann dazu genutzt werden, die Punkte als Kombination von Formprimitiven darzustellen, um die Datenmenge zu reduzieren. Vorliegende Farbinformationen können als Textur gespeichert und bei Bedarf mit reduziertem Farbraum oder in geminderter Auflösung übertragen werden. In Reliefkarten gespeicherte Abweichungen von den Formprimitiven können nach der Übertragung der Daten genutzt werden, um die ursprüngliche Punktwolke genauer zu approximieren.

1.2 Zielsetzung und Herangehensweise

Das Ziel der Arbeit ist die Entwicklung eines Verfahrens zur effizenten Segmentierung einer unsortierten Punktwolke und dessen Evaluierung. Als Eingabe erhält der Algorithmus eine 3D Punktwolke. Zusätzlich zur Positionsinformation ist es möglich, dass Intensitäten oder Farbwerte vorliegen werden. Hierbei werden Remissionswerte, die viele Laserscanner zurückliefern, als Intensitäten behandelt. Als Ausgabe erzeugt der Algorithmus eine Menge von parametrisierten Formprimitiven, welche die Eingabe-Punktwolke annähern. Punkte der Eingabemenge, die nicht zu Formprimitiven zusammengefasst werden können, werden als “unsegmentiert” markiert. Die Formprimitive der erzeugten Ausgabe werden mit einer Belegtheitskarte versehen, so dass die Primitive dadurch auf durch 3D Punkte belegte Teile des Primitivs reduziert werden können. Außerdem werden zur Verfügung stehende Farb- oder Intensitätsinformationen als Textur angezeigt. Abweichungen von den parametrisierten Modellen werden in Form von Reliefkarten gespeichert und dargestellt.

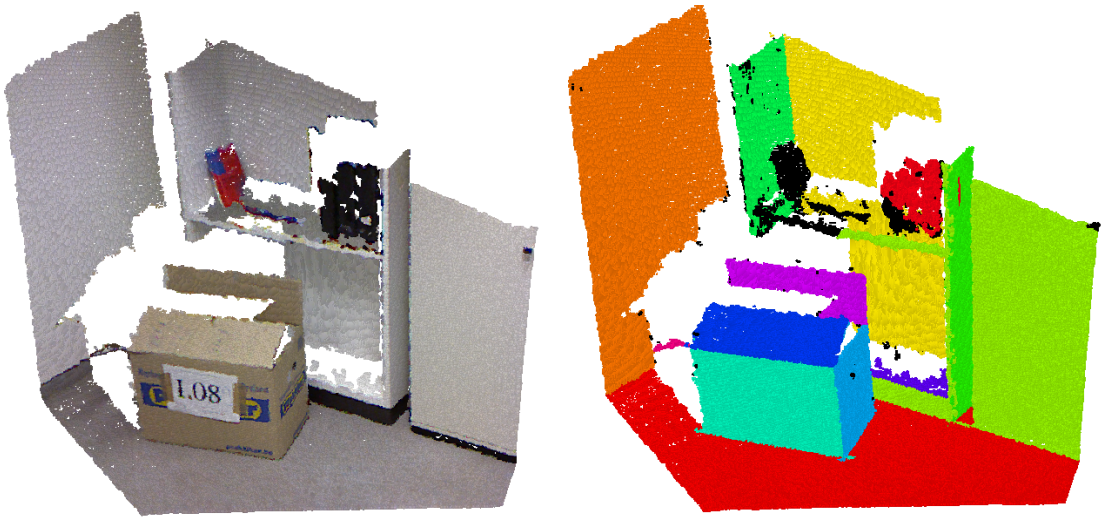


Abbildung 1.1: Beispielszene links und ihre planare Segmentierung auf der rechten Seite. Die Farbe eines Bildpunktes kodiert die Zugehörigkeit zu einem Segment, schwarze Bildpunkte sind keinem Segment zugeordnet.

Um diese Ausgaben zu erzeugen, werden zuerst unterschiedliche Auflösungsstufen der Punktmenge generiert. Ein Punkt auf einer gröberen Auflösung wird Oberflächenelement, kurz Surfel genannt, da er mehrere Oberflächenpunkte zusammenfasst. Im Anschluss werden für diese Surfel Normalen berechnet, bevor die Auflösungsstufen von grob nach fein durchlaufen werden. Dabei werden alle Surfel einer Auflösung mithilfe der Hough-Transformation vorsegmentiert, das heißt, dass alle Surfel dazu genutzt werden, Modellhypothesen zu generieren und die entsprechenden Parameter in einem Histogramm einzutragen. Die Suche des maximal bewerteten Parametersatzes im Histogramm führt zur am besten bewerteten Modellhypothese. Die daran beteiligten Surfel werden im nächsten Schritt auf die Oberfläche des geschätzten Modells projiziert und anhand eines regelmäßigen Gitters in Zusammenhangskomponenten aufgeteilt. Die Einzelpunkte der Punktmenge, die zu einem Surfel einer Zusammenhangskomponente beigetragen haben, werden anschließend dazu genutzt, mithilfe von RANSAC die Parameter des zuvor gefundenen Modells und die tatsächlich beteiligten Punkte genauer zu bestimmen. Auf den folgenden Auflösungsstufen werden die Surfel zuerst den bereits gefundenen Segmenten zugeordnet, bevor die Segmentierung der nicht zugeordneten Surfel mit der Hough-Transformation erneut beginnt. Auf der höchsten Auflösungsstufe verfeinert eine Nachbearbeitung die Punktzuordnung zu den Segmenten.

Zur Beurteilung des Verfahrens, werden die Ergebnisse der Segmentierung mit dem populären SegComp Datensatz [6] verglichen, der aus Tiefenbildern und händisch erzeugter Segmentierung besteht. Des Weiteren wurden im Laufe der Arbeit ausgewählte Kinect Tiefenbilder händisch segmentiert und zur Evaluierung genutzt. Zusätzlich wird die Echtzeitfähigkeit des Verfahrens anhand einiger Testdaten überprüft, die mit einer Kinect und einem schwenkbaren Laser erzeugt werden.

Abbildung 1.1 zeigt eine mit der Microsoft Kinect erzeugte colorierte Punktwolke und eine Beispielsegmentierung durch das hier entwickelte Verfahren. Schwarz eingefärbte Punkte wurden in keinem Segment zugewiesen, bunte Punkte zeigen durch ihre Einfärbung die Zugehörigkeit zu einem planaren Segment an.

1.3 Beitrag der Arbeit

Der Hauptbeitrag der Arbeit besteht in der geschickten Kombination zweier bekannter und populärer Verfahren zur Segmentierung:

- Kombination von RANSAC und Hough-Transformation in einem Multiresolutionsverfahren
- Verbesserung der Hough-Transformation für Zylinder
- Approximation von Unsicherheiten im Parameterraum der Hough-Transformation
- Reliefkarten und Texturen zur Repräsentation der ursprünglichen Daten

1.4 Verwandte Arbeiten

In den folgenden Unterabschnitten werden einige verwandte Arbeiten betrachtet, die zum Teil als Grundlage für den in dieser Arbeit entwickelten Algorithmus dienen. Zuerst wird ein Segmentierungsverfahren auf RANSAC-Basis beschrieben, danach wird auf Arbeiten zur Hough-Transformation im dreidimensionalen Raum eingegangen. Im Anschluss an diese zwei Verfahren, die auf Punktwolken arbeiten, steht noch ein kurzer Überblick über Segmentierungsverfahren, die auf Tiefenbildern arbeiten.

1.4.1 Segmentierung mit RANSAC

Schnabel et al. haben in ihren Arbeiten [13] [14] ein Verfahren zur Erkennung von Ebenen, Zylindern, Kugeln, Tori und Kegelstümpfen in Punktwolken vorgestellt, das mithilfe des RANSAC-Verfahrens funktioniert. Weiterhin wurde bei der RANSAC-typischen zufälligen Auswahl der Beobachtungen eine Einschränkung hinzugefügt, die ausnutzt, dass Punkte, die in der Nähe voneinander liegen, wahrscheinlicher zu einem Formprimitiv gehören, als wenn sie sich weit entfernt befinden. Die Anzahl der Beobachtungen für die Modellgenerierung wurde in diesem Verfahren absichtlich höher als minimal gewählt, damit die Generierung überbestimmt ist und damit statistisch stabiler wird. Nachdem ein Modell mit unterstützenden Punkten festgelegt wurde, ermittelt eine Zusammenhangsanalyse unterschiedliche Komponenten des Primitivmodells und trennt nicht zusammenhängende Teile voneinander. Die verschiedenen Primitive werden gegeneinander gewichtet, indem die Gesamtzahl der unterstützenden Punkte miteinander verglichen wird.

Gotardo et al. [4] segmentieren das gegebene Tiefenbild mithilfe von Kanteninformationen vor. Für diese Segmente werden anschließend mit einer modifizierten RANSAC-Variante Ebenenparameter geschätzt.

1.4.2 Segmentierung mit Hough-Transformation

In den Arbeiten von Rabbani et al. [11] [20] [19] werden Hough-Transformations Verfahren zur Erkennung von Zylindern, Ebenen und Kugeln aus 3D Punktwolken vorgeschlagen. Alle drei Verfahren arbeiten dabei in mehreren Stufen, um die Problematik der hohen Dimension des Parameterraums zu umgehen. Das heißt, dass nicht nur ein Histogramm verwendet wird, um alle Modellparameter gleichzeitig zu bestimmen. Statt dessen werden mehrere Histogramme gebildet, um einzelne Parameter getrennt voneinander zu bestimmen.

Aufgrund der nötigen Auflösung, welche die Histogramme jeder Hough-Transformationsstufe haben müssten, um eine genaue Schätzung der Parameter zu erhalten, ist dieses Verfahren allein nicht effizient oder, bei geringer Auflösung, recht ungenau. Daher wird im hier präsentierten Algorithmus eine Abwandlungen des von Rabbani entwickelten Verfahrens für Ebenen benutzt. Relativ grob aufgelöste Histogramme werden verwendet, um eine Vorsegmentierung vorzunehmen und die Ausreißerquoten für einen anschließenden RANSAC-Schritt niedrig zu halten. Das

von Rabbani [11] vorgeschlagene Verfahren zur Detektion von Zylindern wird in dieser Arbeit zum einen um eine Möglichkeit zur Approximation von Unsicherheiten im Parameterraum erweitert. Zum anderen werden lokale Punktepaare statt einzelnen Punkten benutzt, um daraus einzelne Parameterhypothesen zu generieren, statt ganze Kurven in ein Histogramm eintragen zu müssen.

1.4.3 Segmentierung von Tiefenbildern

Die Segmentierungsverfahren, die auf Tiefenbildern arbeiten, nutzen größtenteils Regionenwachstums-, Regionenzerteilungsverfahren oder eine Mischung aus beidem. Jiang et al. [9] segmentieren Tiefenbilder zuerst zeilenweise, um dann die Liniensegmente mithilfe von Heuristiken miteinander zu verbinden. Ein Nachbearbeitungsverfahren verbessert iterativ die Kanten der gefundenen Segmente. Taylor et al. [18] haben einen Algorithmus entwickelt, der zuerst kleine 3×3 Fenster auf Planarität prüft, die im Laufe des Verfahrens immer weiter verschmolzen werden. Sie erzielen Geschwindigkeitszuwachs durch geschickte Repräsentation der Ebenenparameter. Harati et al. [5] vergleichen zwei Segmentierungsverfahren auf Bearing-Angle (BA) Bildern. Ein Verfahren arbeitet ähnlich wie [9] durch Segmentierung der Bildzeilen und anschließendes Zusammenfügen der Zeilensegmente zu Regionen. Das andere Verfahren bildet zweidimensionale Segmente im Kantenbild, die getrennt werden, durch aus den BA Bildern extrahierte Kanteninformationen.

Durch die gegebene Pixelnachbarschaft, können Verfahren auf Tiefenbildern Informationen über den Zusammenhang von einzelnen Bildpunkten erschließen und ausnutzen. Bei unsortierten 3D Punktwolken sind solche Informationen nicht enthalten und können daher nicht genutzt werden. Verfahren, die auf unsortierten Punktwolken arbeiten, sind damit allgemeiner, da auch Informationen verarbeitet werden können, die nicht zu einem Tiefenbild zusammengefügt werden können. Ein Beispiel hierfür ist eine Sequenz von Tiefenbildern, die zu einer Vollansicht eines Objektes zusammengefügt wurden, oder eine Punktwolke, die aus der Fusion mehrerer Sensoren entstanden ist.

1.5 Aufbau der Arbeit

In Kapitel 2 werden die Grundlagen vorgestellt, wie zum Beispiel der Oktalbaum als Datenstruktur für Punktmengendaten. Danach wird auf die Hauptkomponen-

tenanalyse eingegangen. Weiterhin werden hier die Hough-Transformation und das RANSAC-Verfahren erklärt. Den Abschluss des Kapitels bilden einige Erläuterungen zu Zusammenhangskomponenten.

In Kapitel 3 wird ein Verfahren zur Detektion von Formprimitiven entwickelt, das die in den Grundlagen beschriebenen Verfahren erweitert und kombiniert. Zusätzlich dazu wird der Einsatz einer „grob zu fein“-Strategie erläutert bevor auf mögliche Schritte zur Nachbearbeitung eingegangen wird. Der letzte Abschnitt des Kapitels befasst sich mit der Repräsentation und Visualisierung der extrahierten Formprimitive.

Im Anschluss daran wird in Kapitel 4 anhand von Experimenten evaluiert, wie gut das vorgeschlagene Verfahren arbeitet. Hierzu wird unter anderem das populäre SegComp-Toolkit [6] zum Vergleich von Segmentierungsverfahren verwendet. Es werden ebenfalls Daten einer Kinect und eines 3D Laserscanners zur Bewertung benutzt.

In Kapitel 5 werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf mögliche zukünftige Arbeiten sowie noch ausstehende Aufgaben in Bezug auf Segmentierung von unsortierten 3D Punktwolken geboten.

2 Grundlagen

In diesem Kapitel werden die Grundlagen der Arbeit erörtert. Zuerst wird die genutzte Hardware beschrieben, dann wird der Oktalbaum erklärt. In den darauf folgenden Abschnitten wird zuerst die Hauptkomponentenanalyse und im Anschluss daran die Hough-Transformation und das RANSAC-Verfahren erläutert. Im Anschluss werden Zusammenhangskomponenten in regelmäßigen Gittern und danach Orientierungshistogramme beschrieben.

2.1 Kinect und Laser-Entfernungsmesser

In diesem Abschnitt wird auf die zur Verfügung stehende Hardware eingegangen. In den Robotern der Arbeitsgruppe Autonome Intelligente Systeme sind verschiedene Sensoren im Einsatz. Zum einen benutzt das NimbRo@Home-Team der Universität Bonn mehrere Microsoft Kinects, zum anderen werden auch schwenkbare 2D Laserscanner benutzt um damit die Umgebung eines Roboter aufzunehmen.

Die Kinect der Firma Microsoft ist eine Verbindung aus Infrarotprojektor, Infrarotkamera und Farbkamera. Außerdem beinhaltet sie noch einige Motoren zum Schwenken und Mikrofone zur Aufnahme von Geräuschen, die in dieser Arbeit allerdings nicht weiter von Bedeutung sind. Mithilfe des Musters, das der Projektor erzeugt, können die gemessenen Intensitätswerte der Infrarotkamera in Tiefenwerte umgerechnet werden. Durch die bereits in der Fabrik bestimmte Kalibrierung kann eine Registrierung des Farbbildes und des Tiefenbildes von der Kinect vorgenommen werden, so dass als Ausgabe eine colorierte 3D Punktwolke mit einer Auflösung von 640×480 Punkten erzeugt wird.

2D Laserscanner wie Modell UTM-30LX der Firma Hokuyo können, wie bei den NimbRo@Home-Robotern bespielsweise verbaut, in einer schwenkbaren Einheit bewegt werden und so zum Generieren von 3D Punktwolken genutzt werden. Generell können mit 2D Lasern Oberflächen durch Abschwenken gemessen werden. Die gemessenen Distanzwerte des Lasers müssen dann mit der Position und Orientierung des Lasers zur Zeit der Aufnahme verrechnet werden, um eine korrekte Rekonstruktion zu erhalten.

Je nach Modell und Hersteller ist es möglich, auf Intensitätswerte der Laserentfernungsmessung zurückzugreifen, welche die Remission der abgetasteten Oberfläche widerspiegelt.

3D Laserscanner, wie zum Beispiel die Modelle HDL-32E und HDL-64E der Firma Velodyne, liefern direkt eine, je nach Öffnungswinkel, Winkelauflösung und Umdrehungszahl, unterschiedlich dichte 3D Punktwolke.

Für die Beurteilung der Ergebnisse wurden zum Teil auch Daten von einem Kamerasystem der Firma ABW genutzt. Ähnlich wie bei der Kinect werden, mithilfe von strukturiertem Licht aus einem Projektor, Disparitäten zwischen Kamera und Projektor gemessen und daraus Tiefeninformationen berechnet. Die ABW Kamera hat eine Auflösung von 512×512 Bildpunkten. Der 30 Tiefenbilder umfassende Datensatz, der zur Auswertung genutzt wurde, wurde in der Arbeitsgruppe für Computer Vision der Universität Bern aufgenommen.

Des weiteren wurde ein zusätzlicher Datensatz zur Auswertung genutzt, der mit einer Laser-Radar-Kamera der Firma Perceptron im CESAR lab des Oak Ridge National Laboratory (Tennessee) generiert wurde. Hierbei entstanden ebenfalls 30 Tiefenbilder mit einer Auflösung von 512×512 Bildpunkten.

2.2 Oktalbaum

Ein Oktalbaum ist eine dreidimensionale Erweiterung eines Binärbaumes beziehungsweise Quartalbaumes im zweidimensionalen (vergleiche Samet et al. [12]). Ein Knoten eines Oktalbaumes hat entsprechend bis zu acht Kindknoten, wobei ein Knoten ein Volumen im dreidimensionalen Raum repräsentiert. Ein einzelner Knoten umfasst dabei einen würfelförmigen Bereich. Die Kindknoten eines Knotens werden konstruiert, indem das Volumen des Knotens in acht gleichgroße, disjunkte Oktanten geteilt wird (vgl. Abb. 2.1).

Die Konstruktion eines Oktalbaumes beginnt mit der Wurzel, also einem einzigen Knoten, der das gesamte, zuvor festgelegte Volumen beinhaltet. Beim Eintragen einzelner Punkte in den Oktalbaum, werden Knoten genau dann in ihre Kinder aufgeteilt, wenn mehrere Punkte in einen Knoten fallen. Um eine beliebig feine Unterteilung zu vermeiden, wird eine Schranke für die kleinstmögliche Auflösung gesetzt, so dass ein Knoten nicht weiter aufgeteilt wird, obwohl mehrere Punkte in das kleinste repräsentierte Volumen fallen. Abbildung 2.2 zeigt ein Beispiel für einen Oktalbaum.

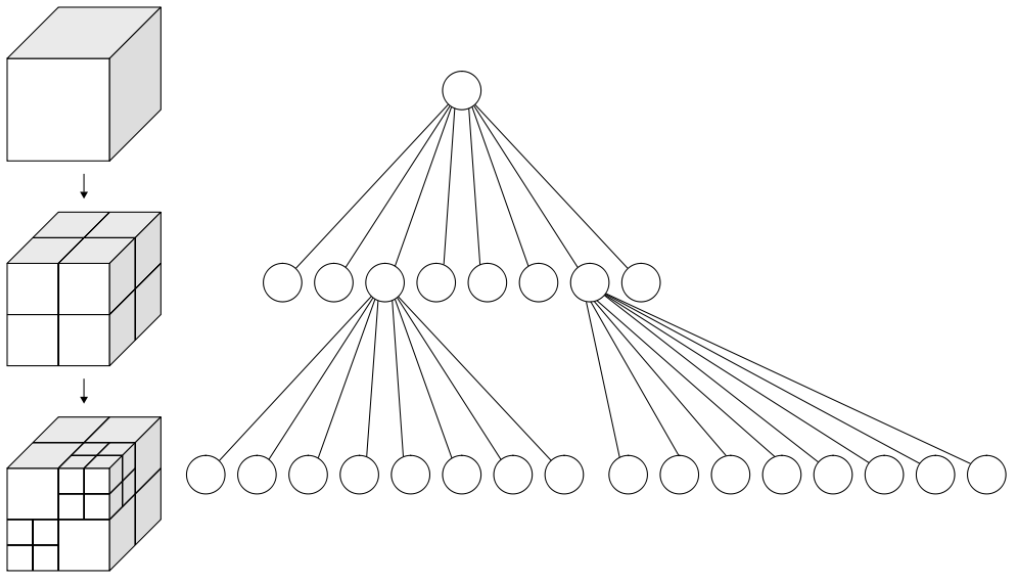


Abbildung 2.1: Visualisierung eines Oktalbaumes in dreidimensionaler Struktur links und als Graph auf der rechten Seite.

Die feste Struktur des Oktalbaumes teilt den verfügbaren Raum in Teilbereiche, die auf jeder Baumtiefe paarweise disjunkt sind, also untereinander eine leere Schnittmenge bilden. Die Tiefe eines Knotens im Baum ist durch die Anzahl der Kanten definiert, die der Knoten von der Wurzel des Baumes entfernt ist.

Aufgrund der Konstruktion des Baumes gibt es weniger Blattknoten b im Oktalbaum als Punkte in der Punktmenge p:

$$b \leq p. \tag{2.1}$$

Ein vollständiger Baum ist ein Baum, dessen innere Knoten die maximale Anzahl Kinder besitzen. Alle Knoten, die keine Blätter sind, werden als innere Knoten bezeichnet. Die Anzahl aller Knoten m eines vollständigen Oktalbaumes mit Baumtiefe

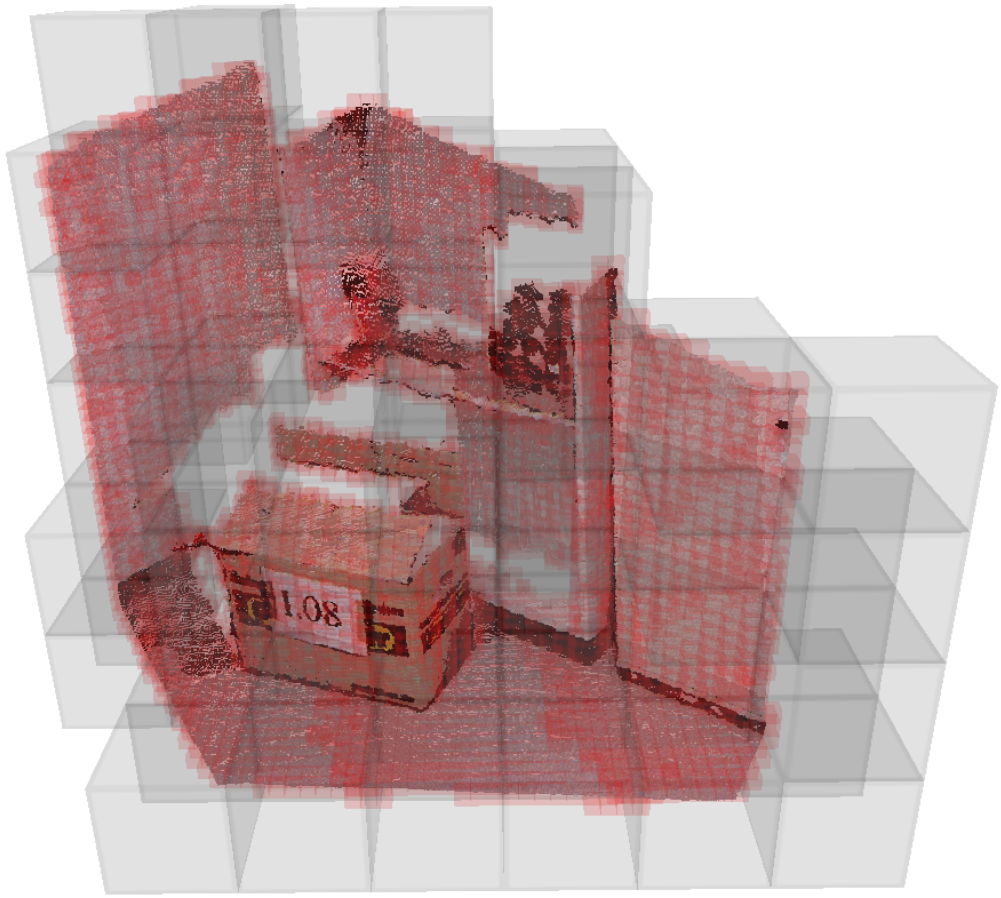


Abbildung 2.2: Knoten eines Oktalbaumes repräsentiert durch ihr Volumen. Es wurden zwei Auflösungsstufen dargestellt. Die grauen Würfel repräsentieren Knoten der Tiefe 8 des Oktalbaumes und haben eine Kantenlänge von $32cm$. Die Knoten der Tiefe 11 wurden durch rote Würfel mit einer Kantenlänge von $4cm$ abgebildet.

t kann durch die Anzahl der Blattknoten b wie folgt ausgedrückt werden:

$$m = \left(\frac{1}{8}\right)^0 b + \left(\frac{1}{8}\right)^1 b + \ldots = b \sum_{i=1}^{t} \left(\frac{1}{8}\right)^i. \tag{2.2}$$

Mithilfe der geometrischen Reihe ergibt sich eine obere Schranke für die Gesamtzahl der Knoten m:

$$m \leq b \sum_{i=1}^{t} \left(\frac{1}{8}\right)^i = b \lim_{t\to\infty} \sum_{i=1}^{t} \left(\frac{1}{8}\right)^i = \frac{b}{1-\frac{1}{8}} = \frac{8}{7}\, b. \tag{2.3}$$

Damit gilt für einen nicht vollständigen Oktalbaum, der höchstens so viele Blätter hat, wie ein vollständiger Oktalbaum:

$$m \leq \frac{8}{7}\, p. \tag{2.4}$$

2.3 Hauptkomponentenanalyse

Nach Hoppe et al. [7] können mithilfe einer Hauptkomponentenanalyse (engl.: Principal Component Analysis, kurz PCA) Punktnormalen in einer unsortierte Punktwolke bestimmt werden, indem nächste Nachbarn betrachtet werden. Eine Annahme des Verfahrens ist allerdings die annähernde Planarität dieser nächsten Nachbarpunkte. Diese Annahme ist in den meisten Fällen zwar erfüllt, aber kann bei 3D Punktwolken auch verletzt sein. Ein Eckpunkt eines Würfels ist ein Gegenbeispiel zu obiger Annahme. Die nächsten Nachbarn zur Ecke liegen auf drei unterschiedlichen Flächen und bilden damit keine annähernd planare Fläche.

Um eine Hauptkomponentenanalyse an einem Punkt durchzuführen, bestimmt man zuerst den Mittelwert der Punktnachbarschaft. Im Falle einer 3D Punktwolke, wie sie hier betrachtet wird, ist der Mittelwert $\boldsymbol{\mu}_P$ auch gleichzeitig der Schwerpunkt $\boldsymbol{cog}_P$ der Punktnachbarschaft P mit k Punkten $\boldsymbol{p}_i$,

$$\boldsymbol{cog}_P = \boldsymbol{\mu}_P = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{p}_i. \tag{2.5}$$

Im Anschluss daran wird die Kovarianzmatrix $\boldsymbol{\Sigma}_P$ der Punktnachbarschaft P berechnet:

$$\boldsymbol{\Sigma}_P = \frac{1}{k} \sum_{i=1}^{k} (\boldsymbol{p}_i - \boldsymbol{\mu}_P)(\boldsymbol{p}_i - \boldsymbol{\mu}_P)^T. \tag{2.6}$$

Eine Eigenwertanalyse der Kovarianzmatrix ergibt Eigenwerte und Eigenvektoren. Der Eigenvektor zum kleinsten Eigenwert entspricht der Normalen der Ausgleichsebene durch die Punktmenge, die durch die beiden größeren Eigenvektoren aufgespannt wird. Die Eigenwerte sind proportional zur quadratischen Summe der Abweichungen vom Mittelwert entlang des entsprechenden Eigenvektors.

2.4 Schnelle Normalenberechnung auf mehreren Skalen

Um aus einem Oktalbaum alle Oberflächenelemente einer Auflösungsstufe zu erhalten, werden alle Knoten einer festgelegten Baumtiefe ausgewählt.

Oktalbaumknoten einer Tiefe repräsentieren ein gleich großes Volumen. Damit auf jeder Auflösungsstufe alle Punkte durch Oktalbaumknoten repräsentiert werden, zählen zu einer Auflösungsstufe auch immer alle Blattknoten aus geringeren Tiefen. Eine Oktalbaumzelle repräsentiert eine Untermenge der Punktwolke und wird im weiteren Text als Surfel bezeichnet, was eine englische Abkürzung für Oberflächenelement (engl.: Surface element) ist.

Um die mittlere Position eines Surfels zu erhalten, wird beim Einfügen der Punkte in den Oktalbaum, während des Durchlaufens eines Punktes von der Wurzel bis zu einem Blattknoten, sowohl ein Zähler für die Anzahl k der repräsentierten Punkte je Knoten, als auch eine Summe der Positionen all dieser Punkte $\boldsymbol{p}_i$ gepflegt. Die Menge der Punkte dieses Knotens wird mit P bezeichnet,

$$P = \{\boldsymbol{p}_1, .., \boldsymbol{p}_k\}. \tag{2.7}$$

Somit kann auf einfache Weise der Mittelwert $\boldsymbol{\mu}_P$ berechnet werden,

$$\boldsymbol{\mu}_P = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{p}_i. \tag{2.8}$$

Die Kovarianzmatrix $\boldsymbol{\Sigma}$ der Punktpositionen P in einem Knoten wird wie folgt bestimmt:

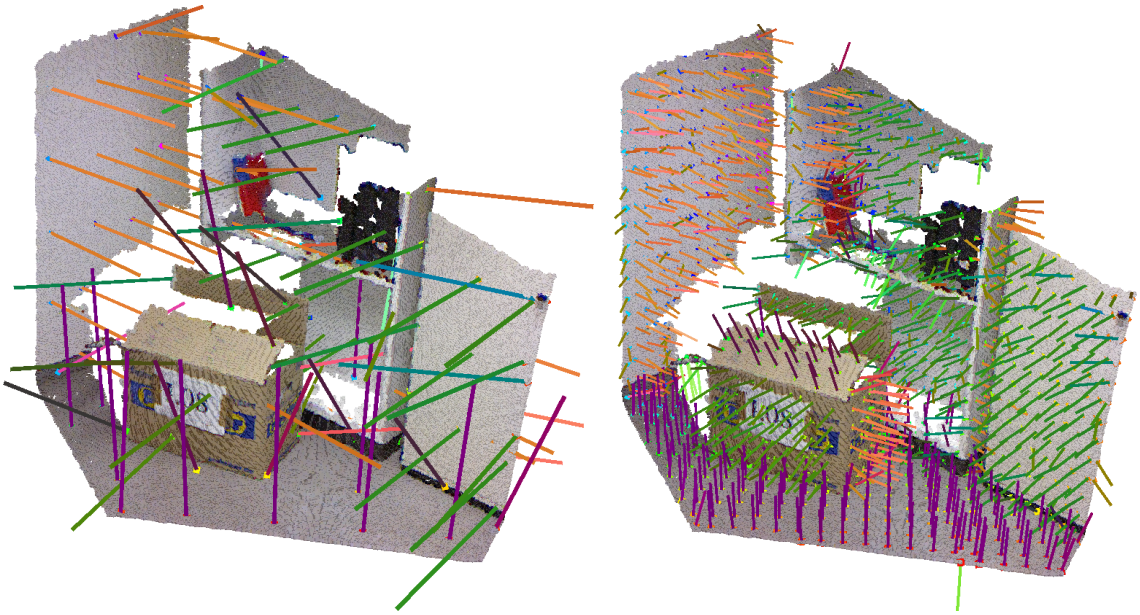


Abbildung 2.3: Visualisierung der Oberflächennormalen auf einer groben (links) und einer feineren Auflösungsstufe (rechts). Die Farbe der Normalen kodiert ihre Richtung. Die Komponenten (x, y, z) der Normalen wurden auf die Farben rot, grün und blau des RGB-Farbraumes abgebildet.

$$\boldsymbol{\Sigma}_P = \frac{1}{k}\sum_{i=1}^{k}\left(\boldsymbol{p}_i - \boldsymbol{\mu}_P\right)\left(\boldsymbol{p}_i - \boldsymbol{\mu}_P\right)^T \tag{2.9}$$

$$= \frac{1}{k}\sum_{i=1}^{k}\left(\boldsymbol{p}_i\boldsymbol{p}_i - \boldsymbol{\mu}_P\boldsymbol{\mu}_P^T\right) \tag{2.10}$$

$$= \left(\frac{1}{k}\sum_{i=1}^{k}\boldsymbol{p}_i\boldsymbol{p}_i\right) - \boldsymbol{\mu}_P\boldsymbol{\mu}_P^T. \tag{2.11}$$

Hierbei wurde zur Umformung der Verschiebungssatz verwendet. Die letzte Formulierung ist nun so gewählt, dass die Summe in jedem Knoten, in dem ein Punkt liegt, während des Aufbaus des Oktalbaumes schnell aktualisiert werden kann.

Diese Kovarianzmatrix wird zur Berechnung der Normalen mithilfe der Hauptkomponentenanalyse benötigt.

Abbildung 2.3 zeigt von diesem Verfahren berechnete Normalen. Die Länge der Normalen entspricht der Abmessung des zur Normalenbestimmung verwendeten Volumens, die Farben kodieren die Normalenrichtung.

2.5 Hough-Transformation

Die Hough-Transformation ist ein Verfahren zur Detektion eines Modells M anhand von Beobachtungen B. Hierzu werden die Beobachtungen in den diskretisierten Parameterraum des Modells transformiert und anschließend wird dort nach einem Parametersatz gesucht, der von den meisten Beobachtungen gestützt wird. Das Verfahren geht zurück auf die Arbeit von Hough [8]. Algorithmus 2.1 zeigt schematisch das Vorgehen des Verfahrens.

Algorithmus 2.1 : Segmentierung mit Hough-Transformation

Data : Beobachtungen $B = \boldsymbol{b}_1, .., \boldsymbol{b}_g$
Result : Modelle M
for $i = 1 .. g$ **do**
    generiere parametrisiertes Modell $m(\boldsymbol{b}_i)$;
    trage $m(\boldsymbol{b}_i)$ in Parameterraum $\mathcal{P}$ ein;
end
while $\exists \mathfrak{p} \in \mathcal{P}$ *mit Anzahl der beitragenden Beobachtungen* $\#(beobachtungen(\mathfrak{p})) > schwellwert$ **do**
    extrahiere Modell $m'(\mathfrak{p})$ mit Parametern $\mathfrak{p}$;
    setze $M := M \cup \{m'(\mathfrak{p})\}$;
    entferne aus $\mathcal{P}$ alle Beiträge von Beobachtungen $\boldsymbol{b}$, die zu Modell $m(\mathfrak{p})$ beigetragen haben;
end

Zuerst ist also eine diskretisierte Darstellung des Parameterraumes des Modells nötig. Hierzu wird für jeden Parameter p_i eine Menge aller sinnvoll möglichen Werte $\mathcal{P}_i$ generiert. Die Darstellung des Parameterraumes ergibt sich aus dem kartesischen Produkt der einzelnen Parameterdarstellungen $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 \times \ldots \times \mathcal{P}_d$, bei d Parametern. Der Parameterraum $\mathcal{P}$ hat damit Dimension d. Zur Repräsentation des diskretisierten Parameterraumes wird ein Histogramm verwendet. Ein Histogramm ist eine Darstellungsform einer diskreten Häufigkeitsverteilung.

Genügt eine Einzelbeobachtung, um das Modell zu bestimmen, trägt man an der einen entsprechenden Stelle des Parameterraumes ein, dass ein Modell mit diesen Parametern durch eine Beobachtung gestützt wird. Für den Fall, dass das Modell aus einer Einzelbeobachtung nicht vollständig bestimmt werden kann, muss eine höherdimensionale Mannigfaltigkeit, z.B. eine Kurve, statt eines einzelnen Punktes in den Parameterraum eingetragen werden. Dazu wird für alle nicht bestimmten Parameter jede mögliche diskrete Belegung angenommen und zur Berechnung des Modells genutzt.

Nachdem alle Beobachtungen in den Parameterraum eingetragen wurden, werden alle Parametersätze ausgewählt, die von mindestens einer bestimmten Anzahl an Beobachtungen gestützt werden. Diese Parametersätze beschreiben Vorkommen des Modells in den Beobachtungen.

Der Vorteil der Hough-Transformation besteht in der Unabhängigkeit von großen Ausreißerzahlen. Leider kann die Hough-Transformation bei komplexen Modellen sowohl speicherineffizient als auch rechenzeitaufwändig sein. Die hohe Speicherkomplexität ergibt sich daraus, dass der hochdimensionerte Parameterraum diskret abgebildet wird. Die hohe Rechenzeit wird beim Eintragen hochdimensionaler Mannigfaltigkeiten pro Beobachtung und beim der Maximumssuche im hochdimensionalen Parameterraum benötigt.

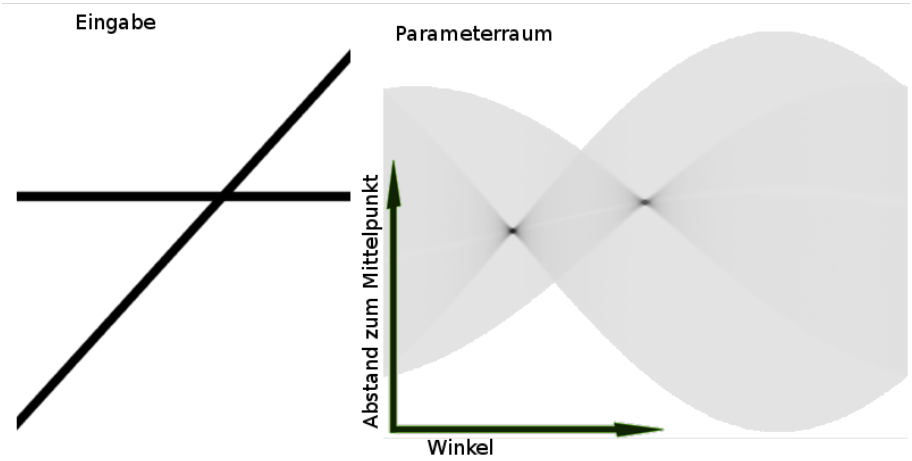


Abbildung 2.4: Abbildung zweier Geraden (links) und der daraus resultierenden Einträge im Histogramm des Parameterraumes der Hough-Transformation (rechts). Je dunkler der Farbwert an einer Stelle des Histogrammes, desto mehr Punkte tragen zu dieser Parameterkonfiguration bei.

Ein einfaches Beispiel für die Hough-Transformation ist die Suche von Geraden in einer Menge von 2D Punkten (siehe Abbildung 2.4 für eine Illustration des Beispiels). Das Modell für eine Gerade wird wie folgt formuliert:

$$M(\alpha, a) = \{(x, y) | y = \cos(\alpha)x + a \text{ mit } x, y \in \mathbb{R})\}. \tag{2.12}$$

Die 2D Punkte werden als Beobachtungen $B \subset \mathbb{R}^2$ bezeichnet. Der Parameterraum $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 = \mathbb{R}^2$ ist gegeben durch die beiden Parameter $\mathfrak{p}_1 = \alpha$, $\mathfrak{p}_2 = a$ und hat damit die Dimension $d = 2$. Ein Punkt beziehungsweise eine Beobachtung $B_i = (x, y)$ wird in eine Kurve

$$a = y - \cos(\alpha)x \tag{2.13}$$

transformiert, die in den Parameterraum eingetragen wird. Das heißt, dass zu jedem Punkt für jeden gültigen Wert von α ein Wert für a berechnet wird. Ein gültiges Wertepaar (α, a), das Gleichung 2.13 erfüllt, wird dann im Histogramm an der den Parametern entsprechenden Stelle eingetragen. Eine Auswahl der lokalen Maxima über einem bestimmten Schwellwert liefert die Parameter α und a sowie jeweils die unterstützenden Punkte für Geraden im zweidimensionalen Raum.

Zusätzliche Informationen, wie zum Beispiel Punktnormalen oder lokale Linensegmente können diese Verfahren beschleunigen [1]. Ein Punkt mit einer Punktnormale, die mit Hilfe eines Gradientenfilters auf einem Bild ermittelt werden kann, bestimmt die Parameter für eine Gerade im zweidimensionalen Raum voll. Auf gleiche Weise läßt sich die Information durch Generierung von Punktepaaren in unmittelbarer Nachbarschaft gewinnen. Daraus ergibt sich, dass der Winkel α aus der Normalen $\boldsymbol{n} = (n_x, n_y)^T$ oder dem Punktepaar $(\boldsymbol{p}, \boldsymbol{q})$ berechnet werden kann. Die Normale ergibt sich aus dem Punktpaar über das Steigungsdreick und die Normierung des resultierenden Vektors:

$$\boldsymbol{n} = \frac{1}{|(p_x - q_x, p_y - q_y)^T|} \cdot (p_x - q_x, p_y - q_y)^T. \tag{2.14}$$

Der Winkel α berechnet sich wie folgt:

$$\alpha = \arctan\left(\frac{n_y}{n_x}\right). \tag{2.15}$$

Damit ergibt sich ein Modell, das von keinem Parameter abhängig ist. In den Parameterraum und das entsprechende Histogramm muss demnach nur ein Punkt eingetragen werden.

2.5.1 Fehlerbehandlung bei der Hough-Transformation

Sowohl Ballard [1] als auch Shapiro [15] [17] [16] haben sich mit Messungenauigkeiten und deren Auswirkung auf die Hough-Transformation beschäftigt. Shapiro hat sich dabei insbesondere mit der statistischen Fehlerfortpflanzung im Falle der Geraden- und Kurvensuche im zweidimensionalen Raum befasst. Ballard schreibt in Bezug auf ein dreidimensionales Histogramm für Kreise im zweidimensionalen Raum, dass man Messungenauigkeiten auch durch regelmäßige Ungenauigkeit im Parameterraum approximieren kann.

2.6 RANSAC

RANSAC nach Fischler et. al. [3] ist ein iteratives Verfahren zur Detektion und Parameterschätzung eines Modells M aus einer Menge von Beobachtungen B. Die Abkürzung RANSAC steht hierbei für die englischen Begriffe RANdom SAmple Consensus, was soviel bedeutet wie "Konsens durch Wahl zufälliger Stichproben". Wie der Name schon sagt, wird eine minimale Anzahl an Stichproben zufällig ausgewählt und im Kontext der übrigen Beobachtungen bewertet, siehe Algorithmus 2.2.

Sobald genügend Iterationen gemacht wurden, so dass bei gegebener Ausreißerquote die Wahrscheinlichkeit zum Auffinden des Modells groß genug ist, bricht das Verfahren ab. Der Erwartungswert für die Anzahl der Iterationen E[k] errechnet sich laut Fischler et al. [3] aus der Wahrscheinlichkeit w eines Punktes, in der Toleranzschwelle eines Modells zu liegen und der minimalen Anzahl b an benötigten Beobachtungen zur Generierung des Modells:

$$\mathrm{E}[k] = w^{-b}. \tag{2.16}$$

Die Wahrscheinlichkeit w berechnet sich aus der Anzahl der Modellpunkte des größten Vorkommens eines Modells N_M in der Punktwolke und der Gesamtzahl der Punkte der Punktwolke N:

$$w = \frac{N_M}{N}. \tag{2.17}$$

Das Problem stellt die unbekannte Wahrscheinlichkeit w dar, die abhänging von der jeweiligen Szene ist und damit ohne Vorwissen nicht bestimmt werden kann. Die Variable b ist abhänging vom gesuchten Modell und damit für jede Szene, aber nicht jedes Formprimitiv gleich.

Algorithmus 2.2 : Segmentierung mit RANSAC

Data :
Beobachtungen $B = \boldsymbol{b}_1, .., \boldsymbol{b}_g$;
Anzahl der Iterationen it;
minimale Anzahl an Beobachtungen für gesuchtes Modell x;
minimale Anzahl an nötigen Beobachtungen für ein Modell min_b;
maximaler Fehler für ein Modell max_f;
Result : Modelle M

```
for |B| > min_b do
    fehler := ∞ ;
    for i = 0 .. it do
        Wähle zufällig x Beobachtungen B' aus B;
        generiere Modell m(B');
        if fehler > fehler(m(B')) then
            m̂ := m(B');
            fehler := fehler(m(B'));
        end
    end
    B = B\B';
    M := M ∪ {m̂};
end
```

w	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6
0,9	1,1	1,2	1,4	1,5	1,7	1,9
0,8	1,3	1,6	2,0	2,4	3,0	3,8
0,7	1,4	2,0	2,9	4,2	5,9	8,5
0,6	1,7	2,8	4,6	7,7	13	21
0,5	2,0	4,0	8,0	16	32	64
0,4	2,5	6,3	16	39	98	244
0,3	3,3	11	37	123	412	1372
0,2	5,0	25	125	625	3125	15625
0,1	10	100	1000	10000	100000	100000

Tabelle 2.1: Erwartungswert für die Anzahl der Iterationen E[k], abhänging von der Anzahl der benötigten Beobachtungen n und vom Verhältnis zwischen Modellpunkten und Gesamtpunktzahl w

Die Standardabweichung für die Anzahl der Iterationen berechnet sich wie folgt:

$$\sigma(k) = \sqrt{\mathrm{E}[k^2] - \mathrm{E}[k]^2}. \tag{2.18}$$

Nach Umformungen ergibt sich

$$\sigma(k) = w^{-n} \sqrt{1 - w^n} \leq w^{-n}. \tag{2.19}$$

Wenn ein Modell mit 99.7% Wahrscheinlichkeit gefunden werden soll, muss zum Erwartungswert $\mathrm{E}(k)$ dreimal die Standardabweichung hizuaddiert werden. Die Anzahl der Iterationen k' beträgt dann

$$k' = \mathrm{E}(k) + 3\sigma(k) \leq 4w^{-n}. \tag{2.20}$$

Im ersten Schritt wird zufällig eine minimale Anzahl an Beobachtungen gewählt, die im nächsten Schritt zur Generierung des parametrisierten Modells benötigt werden. Anschließend wird eine Untermenge der Beobachtungen erstellt, deren Einzelbeobachtungen das Modell unter Berücksichtigung einer maximalen Abweichung ϵ_{ransac} unterstützen. Nachdem ein bestes Modell gefunden ist, kann abschließend mittels Optimierung, beispielsweise durch Minimierung des quadratischen Fehlers, ein genaueres Modell für die unterstützenden Beobachtungen erhalten werden.

Einer der Vorteile von RANSAC ist die große Robustheit gegenüber Ausreißern. Selbst bei einer hohen Ausreißerrate von über 50% ist eine Detektion wahrscheinlich, wenn genügend Iterationen durchlaufen, also genügend Modelle generiert und getestet werden.

Bei hoher Ausreißerrate sind allerdings sehr viele Iterationen notwendig. Da bei der Bewertung des Modells jede Beobachtung erneut gegen das Modell getestet wird, müssen sehr viele dieser Tests gemacht werden. Durch die zufällige Stichprobenwahl kann es bei hohen Ausreißerquoten und zu wenigen Iterationen vorkommen, dass bei gleicher Eingabemenge und gleicher Rechenzeit unterschiedliche Ergebnisse erzielt werden. Wird die Ausreißerquote nicht vor der Suche des Modells bestimmt, kann passieren, dass die Anzahl der Iterationen zu niedrig gewählt wird. Durch Verletzung der Gleichung 2.16 bzw. Gleichung 2.20 könnte keine zuverlässige Detektion erfolgen. Ohne eine vorhergehende Detektion kann die Ausreißerquote nicht zuverlässig bestimmt werden. Man müsste eine Heuristik anwenden und hätte je nach Heuristik und Beobachtungen Fehler in der Bestimmung der Anzahl der nötigen Iterationen. Daher wurde in dieser Arbeit eine grob gerasterte Hough-Transformation als Vorverarbeitungsschritt zur Verringerung der Ausreißerquoten und der Senkung der benötigten Iterationen eingefügt.

2.7 Orientierungshistogramm

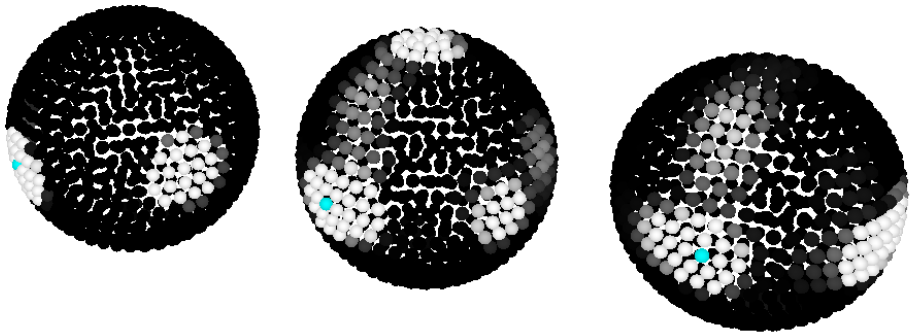


Abbildung 2.5: Orientierungshistogramme für unterschiedliche Auflösungsstufen. Schwarz gefärbte Punkte auf der Kugeloberfläche stehen für nicht oder kaum belegte Histogrammfelder. Je heller die Farbe, desto höher ist die Summe der Gewichte der beteiligten Orientierungen. Das Maximum der Gewichtssumme wurde mit türkiser Farbe gekennzeichnet.

Das Orientierungshistogramm, das dazu verwendet wird, die Orientierung von Ebenen beziehungsweise die Achse von Zylindern zu ermitteln, ist ein zweidimensionales Histogramm, das kein regelmäßiges Gitter bildet. Die zwei Dimensionen des Histogramms ergeben sich aus den beiden Winkeln, welche die Orientierung bestimmen. Werden beide Winkel regelmäßig parametrisiert, so häufen sich die repräsentierten Orientierungen an den Polkappen.

Daher wurde, wie bei Rabbani [11], eine Approximation einer regelmäßigen Verteilung auf einer Kugeloberfläche gewählt. Hierzu wird der erste Winkel regelmäßig, der andere Winkel abhängig vom ersten Winkel parametrisiert.

Entspricht nun n_ϕ der Anzahl von möglichen Werten für ϕ im Histogramm, dann berechnet sich die Anzahl der möglichen Werte $n_{\theta i}$ für θ aus dem festen Wert ϕ_i durch die Formel:

$$n_{\theta i} = 2n_\phi \sin \phi_i + 1. \tag{2.21}$$

Damit lassen sich bei bekannten Indices ϕ_{index} und θ_{index} sehr leicht die entsprechenden Werte der Winkel ϕ_o und θ_o berechnen. Aufgrund der regelmäßigen Parametrisierung ist der Winkel zwischen zwei Histogrammzellen ϕ_{step} bekannt,

$$\phi_o = \phi_{index}\, \phi_{step}. \tag{2.22}$$

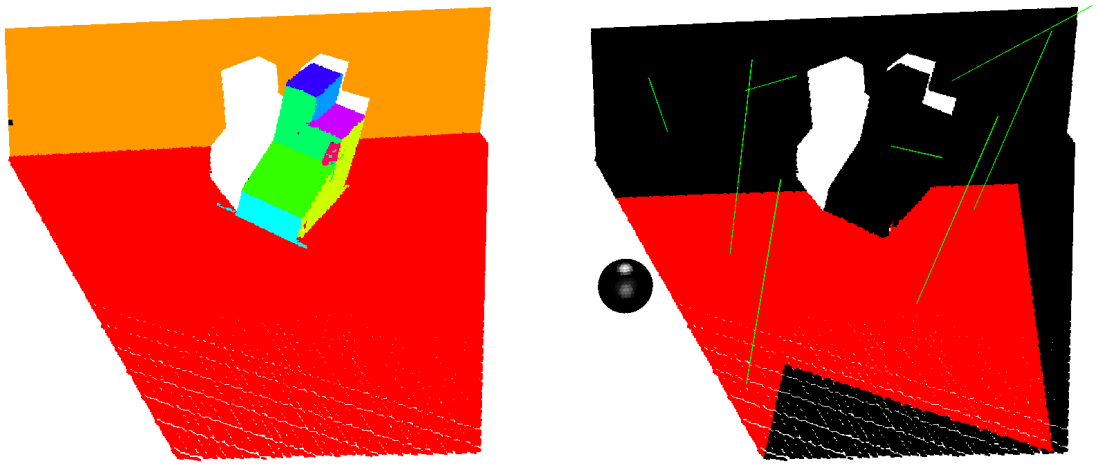


Abbildung 2.6: Punktwolke nach Segmenten eingefärbt (links) und Orientierungshistogramm für eine Auflösungsstufe von 1,28m (rechts). Das linke Bild dient nur der Veranschaulichung der Szene. Auf dem rechten Bild ist an der linken Seite des Bildes eine Kugel abgebildet, die das Orientierungshistogramm darstellet. Schwarz gefärbte Punkte auf der Kugeloberfläche stehen für nicht oder kaum belegte Histogrammfelder. Je heller die Farbe, desto höher ist die Summe der Gewichte der beteiligten Orientierungen. Im oberen Bereich der Kugel ist eine Häufung zu sehen, die durch die vier Normalen dort entsteht, die sich im rot markierten Teil der Punktwolke befinden. Der leicht grau schimmernde Bereich der Kugel im vorderen Teil entsteht durch Eintragen der anderen vier Normalen und bildet keinen guten Häufungspunkt.

Der Winkel θ_{step} zwischen zwei möglichen Werten in der Parametrisierung von θ berechnet sich in Abhängigkeit von ϕ_o:

$$\theta_{step} = \frac{\pi}{2 \sin \phi_o}. \tag{2.23}$$

Damit lässt sich der Wert θ_o des Winkels θ abhänging vom Index θ_{index} bestimmen:

$$\theta_o = \theta_{index}\, \theta_{step} = \theta_{index}\, \frac{\pi}{2 \sin \phi_o}. \tag{2.24}$$

Abbildung 2.6 zeigt ein Orientierungshistogramm in Form von kugelförmig angeordneten Punkten. Je ein Punkt auf der Kugel repräsentiert dabei ein Histogrammfeld. Die Farbe des Punktes kodiert dabei die Summe der Gewichte dieses Feldes.

2.8 Zusammenhangskomponenten

Definitionen von Zusammenhangskomponenten beziehen sich in der Literatur üblicherweise auf Graphen, wie auch die folgende Definition. Sei V eine Menge von Knoten und sei $E \subseteq V \times V$ eine Menge von Kanten. Zusammen definieren diese beiden Mengen einen ungerichteten Graphen $G = (V, E)$. Zusammenhangskomponenten in einem ungerichteten Graphen sind eine Menge von zusammenhängenden Knoten. Zwei Knoten sind genau dann zusammenhängend, wenn sie durch eine Kante miteinander verbunden sind:

$$zusammenhaengend(v_i, v_j) \quad \Leftrightarrow \quad \exists e = (v_i, v_j) \in E. \tag{2.25}$$

Für das hier vorgestellte Verfahren werden Zusammenhangskomponenten in einem regelmäßigen Gitter benötigt. Hierzu werden bei einem Gitter der Größe $I \times J \subseteq \mathbb{N}^2$ für jede Zelle (i, j) in diesem Gitter eine Belegtheitsinformation $b_{i,j}$ benötigt:

$$b_{i,j} \in \{0, 1\} \quad \forall (i, j) \quad \text{mit} \quad i \in I \quad \text{und} \quad j \in J. \tag{2.26}$$

Die Symbole 0 und 1 stehen für "unbelegt" beziehungsweise "belegt". Diese Belegtheitsinformation repräsentiert die Knotenmenge in einem entsprechenden Graphen.

Die Kantenmenge eines Graphen bestimmt maßgeblich die Zusammenhangskomponenten. Auf regelmäßigen Gittern wird zu diesem Zweck eine Nachbarschaft definiert.

Zwei Beispiele für Nachbarschaften sind die Von-Neumann-Nachbarschaft, in der alle Zellen als benachbart gelten, die eine Gitterkante gemeinsam haben. In der Moore-Nachbarschaft werden zusätzlich noch Zellen als benachbart angesehen, wenn sie eine gemeinsame Gitterecke besitzen.

Eine formelle Definition der Von-Neumann-Nachbarschaft:

$$Nachbar_N(i, j) = \{(i + 1, j), (i - 1, j), (i, j + 1), (i, j - 1)\}. \tag{2.27}$$

Die Moore-Nachbarschaft wird entsprechend definiert als:

$$Nachbar_M(i, j) = Nachbar_N(i, j) \cup \{(i+1, j+1), (i+1, j-1), (i-1, j+1), (i-1, j-1)\}.$$

Bei Nachbarschaften auf begrenzten Gittern muss der Rand des Gitters besonders beachtet werden. Eine Möglichkeit zur Randbehandlung besteht in der Annahme,

dass aufgrund des Randes fehlende Nachbarn wie nicht belegte Zellen behandelt werden.

Eine Zusammenhangskomponente ist nun eine Menge C, von Zellen beziehungsweise ihrer Repräsentation durch Koordinatenpaare, die zueinander benachbart sind und die alle als "belegt" gelten,

$$(i,j) \in C \Leftrightarrow (b_{i,j} = 1 \wedge Nachbar(i,j) \cap C \neq \emptyset)\,, \tag{2.28}$$

wobei $Nachbar(i,j)$ eine beliebige Nachbarschaft der Zelle (i,j) bezeichnet.

Um die Zusammenhangskomponenten eines regelmäßigen Gitters zu bestimmen, müssen alle Zellen einmal besucht und eine Tiefen- oder Breitensuche in jeder Zelle gestartet werden, die als belegt markiert ist, aber noch nicht besucht wurde. Hierzu ist eine vorhergehende Markierung aller Zellen als "nicht besucht" vorauszusetzen. Jede nicht besuchte, aber belegte Zelle, die in der Nachbarschaft einer besuchten, belegten Zelle liegt, wird in die Zusammenhangskomponente dieser Zelle eingefügt. Sobald eine Tiefen-/Breitensuche beendet ist, ist auch die entsprechende Zusammenhangskomponente vollständig gefunden. Die nächste Suche beginnt dann in einer noch nicht besuchten, aber belegten Gitterzelle, falls es eine solche noch gibt.

Die Laufzeit der Suche nach Zusammenhangskomponenten ergibt sich aus der Anzahl der belegten Zellen $\#(belegt)$, die besucht werden müssen und der Anzahl der Tests $\#(tests)$, welche benachbarten Zellen noch besucht werden müssen und welche nicht. Zum einen wird dieser Test für jede Gitterzelle einmal aufgerufen, um zu prüfen, ob in dieser Zelle eine Zusammenhangskomponente gestartet wird. Daher besteht der erste Summand aus der Anzahl der Gitterzellen $m = |I| \cdot |J|$. Zum anderen wird für jede belegte Zellen genau einmal jeder Nachbar getestet. Danach wird die Zelle, deren Nachbarn alle getestet wurden markiert, damit kein erneuter Besuch stattfindet. Es ergibt sich der zweite Summand aus dem Produkt der Anzahl der belegten Zellen $\#(belegt)$ und der Anzahl der Nachbarn in der gewählten Nachbarschaft $\#(Nachbarn)$:

$$\#(tests) = m + \#(belegt) \cdot \#(Nachbarn). \tag{2.29}$$

Die Anzahl der belegten Zellen $\#(belegt)$ ist maximal so groß, wie die Anzahl der Zellen im Gitter m,

$$m >= \#(belegt). \tag{2.30}$$

Es ergibt sich für die Gesamtzahl der Tests:

$$\#(tests) \leq m + m \cdot \#(Nachbarn) \in O(m). \tag{2.31}$$

Die Gesamtzahl der Besuche $\#(visits)$ ist gleich der Anzahl der belegten Zellen $\#(belegt)$, da jede belegte Zelle nur genau einmal besucht wird,

$$\#(visits) = \#(belegt) < m \in O(m). \tag{2.32}$$

3 Segmentierungsverfahren

In diesem Kapitel wird das im Rahmen der Diplomarbeit entwickelte Verfahren zur Detektion von Formprimitiven in einer unsortierten 3D-Punktwolke beschrieben. Hierzu ist zuerst die Berechnung von Surfels auf unterschiedlichen Auflösungsstufen und die Schätzung der Oberflächennormalen nötig. Im Anschluss wird das Verfahren auf einer Surfelmenge einer Auflösungsstufe beschrieben, bevor der Übergang zwischen den Skalen erklärt wird. Danach wird die Visualisierung der Abweichungen der Punkte von den Formprimitiven in Form von Reliefkarten erläutert. Zusätzlich werden vorhandene Farbwerte der Punktwolke in Texturen dargestellt.

3.1 Normalenfilterung

Da die Bestimmung der Normalen aus fast beliebig kleinen Ausschnitten der Punktmenge erfolgt, die beliebig in diesem Ausschnitt verteilt liegen kann, müssen an dieser Stelle noch einige Beschränkungen für gültige Normalen eingeführt werden. Zum einen werden mindestens drei Punkte benötigt, um eine Normale zu berechnen. Je mehr Punkte zur Berechnung beitragen, desto (statistisch) stabiler wird die Normale, vorausgesetzt dass die Punkte ein planares Segment beschreiben. Daher wird ein Schwellwert ν_{normal} genutzt, der eine Mindestanzahl an beteiligten Punkten pro Normale angibt. Zum anderen macht es nur Sinn Normalen für die folgenden Berechnungen zu berücksichtigen, die auch aus einer fast planaren Punktmenge berechnet wurden. Hierzu wurde die Krümmung γ_k an dieser Normalen $\boldsymbol{n}_k$ aus den

Eigenwerten λ_i bestimmt:

$$\gamma_k = \frac{\lambda_0}{\lambda_2} \tag{3.1}$$

Das Verhältnis zwischen dem kleinsten λ_0 und dem größten Eigenwert λ_2 gibt an, wie die Ausdehnungen eines Surfels in der größten und der kleinsten Ausdehnungsrichtung zueinander stehen. Für γ_k gilt dabei $0 < \gamma_k <= 1$, wobei ein Wert nahe 0 auf eine stabile Normale eines nahezu planaren Surfels hindeutet und ein Wert von 1 ein Hinweis auf eine im Volumen gleichverteilte Punktmenge ist. Ein zweites Kriterium für die Güte einer Normalen ist das Verhältnis γ_{k2} der beiden zweitgrößten Eigenwerte:

$$\gamma_{k_2} = \frac{\lambda_1}{\lambda_2}. \tag{3.2}$$

Dieses Verhältnis gibt die zweidimensionale Ausdehnung eines planaren Surfels an. Wenn der größere Eigenwert λ_1 sehr viel größer als der zweitgrößte Eigenwert λ_2 ist, so deutet das auf eine eher längliche Ausdehnung des Surfels hin. Sind die zwei größten Eigenwerte ähnlich groß, so ist die Ausdehnung des Surfels entlang der Ausgleichsebene in Richtung beider Eigenvektoren etwa gleichmäßig.

Der Schwellwert, den die Krümmung γ_k einer Normalen maximal erreichen darf, damit die Normale nicht verworfen wird, wird mit γ_{max} bezeichnet. Der Schwellwert γ_{max2} bestimmt das maximale Verhältnis zwischen den größeren Eigenwerten γ_{k2}.

3.2 Detektion von Formprimitiven

Zuerst werden in diesem Abschnitt die Detektion von Ebenen und die dabei vollzogenen Teilschritte beschrieben. Dazu gehören Hough-Transformation als Vorsegmentierung und Zerlegung in Zusammenhangskomponenten. Der letzte Teilschritt ist RANSAC, das zur Verfeinerung der Primitivparameter sowie zur genauen Bestimmung der beteiligten Punkte verwendet wird. Anschließend werden Hough-Transformation, Zerlegung in Zusammenhangskomponenten und RANSAC für Zylinder beschrieben. Abbildung 3.1 zeigt ein Ablaufdiagramm

3.2.1 Ebenen

Die Detektion von Ebenensegmenten unterteilt sich in drei Schritte. Zuerst wird die Hough-Transformation angewendet, um eine erste Näherung für Ebenenpara-

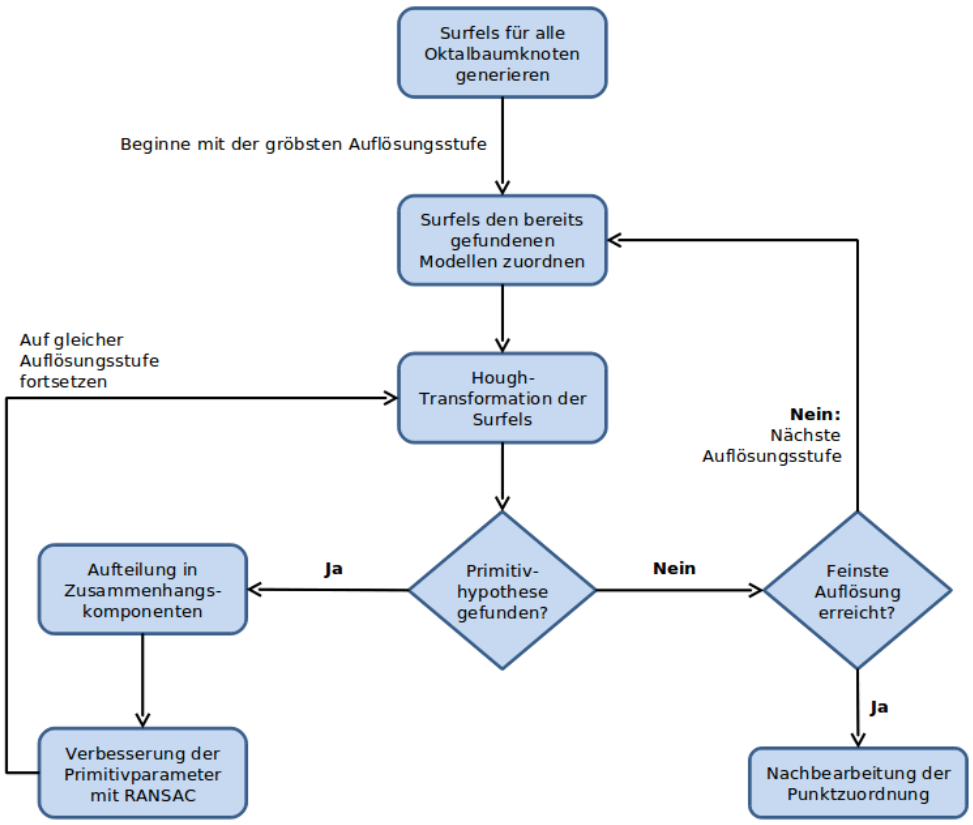


Abbildung 3.1: Dieses Ablaufdiagramm zeigt schematisch die einzelnen Schritte des Verfahrens. Zuerst werden alle Surfels und ihre Normalen für alle Auflösungsstufen des Oktalbaumes generiert. Die Auflösungsstufen werden von grob nach fein einzeln betrachtet. Beginnend mit der ersten Auflösungsstufe werden alle Surfels der Stufe zu bereits gefundenen Modellen zugeordnet. Die nicht zugeordneten Surfels werden mithilfe der Hough-Transformation vorsegmentiert. Ergibt die Hough-Transformation eine Primitivhypothese, so werden alle beteiligten Surfels anhand der Oberfläche des Primitivs in Zusammenhangskomponenten aufgeteilt. Mittels RANSAC werden die Primitivparameter verbessert. Die Surfels, aus denen erfolgreich Formprimitive detektiert wurden, werden aus der Menge der nicht segmentierten Surfels entfernt, bevor die Hough-Transformation erneut angewendet wird. Ergibt der Hough-Transformations Schritt keine Primitivhypothese, so setzt der Algorithmus die Detektion von Formprimitiven auf der nächstfeineren Auflösungsstufe fort. Gibt es keine nächstfeinere Auflösungsstufe, so wird die Detektion von Formprimitiven mit einem Nachverarbeitungsschritt abgeschlossen.

meter zu erhalten und weit entfernte Ausreißer zu identifizieren. Um bei der Detektion nicht nur Ebenen, sondern begrenzte, zusammenhängende Ebenensegmente zu erhalten, wird im nächsten Schritt eine Suche nach Zusamenhangskomponenten durchgeführt. Für jede dieser Zusammenhangskomponenten wird im Anschluss daran RANSAC ausgeführt um weitere Ausreißer zu detektieren und eine bessere Näherung der Ebenenparameter zu erhalten.

Vorsegmentierung im Hough-Raum

In diesem Unterabschnitt wird zuerst die Hough-Transformation für Ebenen beschrieben und danach das in der Arbeit verwendete Verfahren entwickelt.

Hough-Transformation für Ebenen Eine Ebene wird repräsentiert durch ihre Normale $\boldsymbol{n}$ und dem als positiv definierten Abstand zum Ursprung d. Damit gilt für jeden Punkt $\boldsymbol{p}$ der Ebene die Hesse-Normal-Form:

$$\langle \boldsymbol{n}, \boldsymbol{p} \rangle - d = 0, \tag{3.3}$$

wobei mit $\langle \cdot, \cdot \rangle$ das Skalarprodukt bezeichnet wird.

Um die Ebenennormale möglichst effizient zu kodieren, wurde bei Rabbani [11] und auch in dieser Arbeit eine Repräsentation über ein Winkelpaar (θ, ϕ) gewählt, so dass für die Normale beziehungsweise für die Winkel θ und ϕ folgende Gleichungen gelten:

$$\boldsymbol{n}_{\theta,\phi} = (n_1, n_2, n_3)^T = (cos(\theta)sin(\phi), sin(\theta)sin(\phi), cos(\phi))^T, \tag{3.4}$$

$$\theta_n = \arccos(n_3), \tag{3.5}$$

$$\phi_n = \arccos\left(\frac{n_1}{\sin(\arccos(n_3))}\right). \tag{3.6}$$

Um eine Ebene eindeutig zu bestimmen, sind also drei Parameter nötig. Dabei legen die beiden Winkel θ und ϕ die Orientierung und der Abstand d die Entfernung vom Ursprung fest. Es ergibt sich das Modell M mit:

$$M(\theta, \phi, d) = \left\{(x, y, z) | \left\langle \boldsymbol{n}_{\theta,\phi}, (x, y, z)^t \right\rangle - d = 0 \text{ mit } x, y, z \in \mathbb{R})\right\}. \tag{3.7}$$

Eine Einzelbeobachtung $\boldsymbol{p} = (x, y, z)^t \in B \subset \mathbb{R}^3$, also ein 3D Punkt, führt für eine Orientierung (θ, ϕ) zu einem Abstand d zum Ursprung des Koordinatensystems:

$$d(\theta, \phi, \boldsymbol{p}) = \langle \boldsymbol{n}_{\theta,\phi}, \boldsymbol{p} \rangle = x \cdot cos(\theta)sin(\phi) + y \cdot sin(\theta)sin(\phi) + z \cdot cos(\phi). \tag{3.8}$$

Zu jedem Punkt müssen alle Orientierungen betrachtet und die berechnete Parameterkonfiguration in den Parameterraum eingetragen werden. Hierfür wird die Parametrisierung der Orientierung gemäß Kapitel 2 Abschnitt 2.7 durchlaufen und für jede Orientierung wird die Distanz einer Ebene, die durch den Punkt $\boldsymbol{p}$ geht, mithilfe von Gleichung 3.8 bestimmt. An den entsprechenden Stellen im Histogramm wird dann der Punkt $\boldsymbol{p}$ als beitragende Beobachtung eingetragen:

$$\forall \theta, \phi : his(\theta, \phi, d(\theta, \phi)) \leftarrow his(\theta, \phi, d(\theta, \phi)) \cup \{\boldsymbol{p}\}. \tag{3.9}$$

Das Histogramm $his(\theta, \phi, d)$ beinhaltet also alle Beobachtungen und die daraus resultierenden Möglichkeiten für die drei Parameter des Modells. Das Maximum dieses Histogramms und damit das beste Modell wird bestimmt, indem das Histogrammfeld ermittelt wird, das die meisten Punkte beinhaltet.

Falls Punktnormalen zur Verfügung stehen, kann statt dessen das Ebenenmodell aus der Einzelbeobachtung mit Gleichung 3.8 berechnet werden und nur ein einzelner Parametersatz in das Histogramm $his(\theta, \phi, d)$ eingetragen werden. Die Winkel θ und ϕ berechnen sich aus der Normalen $\boldsymbol{n_p}$ des Punktes $\boldsymbol{p}$, wie in Formel 3.5 angegeben:

$$his_n(\theta_{\boldsymbol{n_p}}, \phi_{\boldsymbol{n_p}}, d(\theta, \phi)) \leftarrow his_n(\theta_{\boldsymbol{n_p}}, \phi_{\boldsymbol{n_p}}, d(\theta, \phi)) \cup \{\boldsymbol{p}\}. \tag{3.10}$$

Um die Rechenzeit möglichst gering zu halten und auch den Speicherbedarf zu minimieren, wird ein zweistufiges Verfahren angewendet. Die Normale und die Distanz der Ebene zum Ursprung werden darin nacheinander bestimmt. Dieses Verfahren ist in der Arbeit von Vosselman et al. [20] und der Doktorarbeit von Rabbani [11] zu finden. Dabei wird im Gegensatz zum einstufigen Verfahren kein dreidimensionaler Parameterraum sondern statt dessen ein zweidimensionaler und ein eindimensionaler Parameterraum benötigt.

Beim ersten Schritt wird die Normaleninformationen der Punkte genutzt, um die Normalenrichtung zu finden, die von den meisten Punktnormalen unterstützt wird. Hierzu werden alle Punktnormalen in der Winkeldarstellung in ein zweidimensionales Histogramm eingetragen,

$$his(\theta_{\boldsymbol{n_p}}, \phi_{\boldsymbol{n_p}}) \leftarrow his(\theta_{\boldsymbol{n_p}}, \phi_{\boldsymbol{n_p}}) \cup \{\boldsymbol{p}\}. \tag{3.11}$$

und im Anschluss das Histogrammfeld mit den meisten Treffern gesucht.

In der zweiten Stufe der Hough-Transformation für Ebenen werden nur noch die Punkte benutzt, die zum Maximum der Normalenorientierungen beigetragen haben. Es wird aus den Punkten und der gegebene Normalenrichtung der entsprechnde Abstand der resultierenden Ebene ermittelt und in ein eindimensionales Histogramm eingetragen. Der Abstand berechnet sich wie zuvor mit Formel 3.8. Das Histogrammfeld mit den meisten Einträgen bestimmt sowohl die Modellpunkte als auch die Belegung des Distanz-Parameters:

$$his(d) \leftarrow his(d) \cup \{\boldsymbol{p}\}. \tag{3.12}$$

Nachteile des zweistufigen Verfahrens ergeben sich daraus, dass die Informationen der beiden Stufen nicht miteinander zusammenhängen. Einige Häufungspunkte in der Orientierung führen nicht zu Häufungspunkten im Distanzhistogramm, denn gleich orientierte Punkte können auf ganz unterschiedlichen Ebenen liegen. Daher extrahiert das hier beschriebene Verfahren zusätzliche Häufungspunkte und verwirft diejenigen, die in der zweiten Stufe kein gutes Maximum bilden.

Modellierung von Unsicherheiten der Ebenenparameter Um mithilfe der Hough-Transformation Ebenen zu finden, werden in dieser Arbeit Surfel mit Normalen verwendet, so dass beim Eintragen im Parameterraum nur einzelne Parameterwerte statt ganze Kurven eingetragen werden müssen. Des Weiteren wurde der zweistufige Ansatz gewählt, um Effizienz zu erzielen.

Zuerst werden alle Surfelnormalen in ein Orientierungshistogramm eingefügt, wie es schon in Abschnitt 2.5 beschrieben wird. Um die Ungenauigkeit in der Bestimmung der Normalen zu berücksichtigen, wird die Normale mit Unsicherheit eingetragen. Hierzu wird um die berechnete Normale ein Kreis an Nachbarn betrachtet und das Surfel wird gewichtet eingefügt. Das Gewicht w_j einer Normalen $\boldsymbol{n}_j$ im Orientierungshistogramm richtet sich dabei nach dem Winkel zur berechneten Normalen $\boldsymbol{n}_k$ und der maximalen Winkelabweichung α_{max}. Die Krümmung γ_k wird ebenfalls zur Abschwächung des Gewichts genutzt. Wie bereits beschrieben, steht ein kleiner Wert für eine stabile Normale und ein großer Wert, Nahe des Schwellwertes γ_{max} für eine weniger stabile Normale. Als zusätzlicher Faktor wurde die Anzahl N_k der zugrundeliegenden Punkte aus der ursprünglichen Punktmenge des Surfels benutzt. Dieser Faktor bestimmt, wie die einzelnen Surfelnormalen untereinander gewertet

werden. Eine Normale, die durch eine dichte Punktmenge mit vielen Punkten gestützt wird, ist damit schwerer gewichtet, als eine Normale, die nur aus wenigen Punkten entstand. Das Gewicht ergibt sich aus der folgenden Formel:

$$w_j = N_k \cdot \left(1 - \frac{\gamma_k}{\gamma_{\max}}\right) \cdot \frac{(|\langle \boldsymbol{n}_k, \boldsymbol{n}_j \rangle| - \cos(\alpha_{max}))}{\cos(\alpha_{max})}, \tag{3.13}$$

$$\forall \boldsymbol{n}_j : his(\theta_{\boldsymbol{n}_j}, \phi_{\boldsymbol{n}_j}) \leftarrow his(\theta_{\boldsymbol{n}_j}, \phi_{\boldsymbol{n}_j}) \cup \{(w_j, \boldsymbol{p})\}. \tag{3.14}$$

Das Maximum des Orientierungshistogramms ergibt sich dann aus der Orientierung mit der maximalen Summe aller Gewichte für diese Orientierung.

Nachdem die Orientierung für eine Ebene bestimmt wurde, wird, wie in Abschnitt 2.5, ein Histogramm gebildet, um die Distanz der Ebene zum Ursprung zu bestimmen, die von den meisten Surfels gestützt wird.

Auch hier ist es sinnvoll, die Ungenauigkeit der Position des Surfels mit einzubeziehen und deswegen das Surfel mit Ungenauigkeit ins gewichtete, eindimensionale Histogramm einzutragen. Die diskrete Parametrisierung ist hierbei abhänging vom Parameter ϵ_{ht}, der die Schrittweite der Parametrisierung angibt. Das heißt, dass zwei benachbarte Parameterwerte im Distanzhistogramm sich in der Distanz um exakt ϵ_{ht} unterscheiden. Das Gewicht der benachbarten Histogrammfelder berechnet sich dabei ähnlich wie bei den Normalen mit linearer Abschwächung. *dev* bezeichnet die Ungenauigkeit der Distanz, d_k ist die berechnete Distanz für einen Punkt k und d_j ist die Distanz eines benachbarten Histogrammfeldes:

$$w_j = \frac{dev - dist(d_k, d_j)}{dev}, \tag{3.15}$$

$$\forall d_j : his(d_j) \leftarrow his(d_j) \cup \{(w_j, \boldsymbol{p})\}. \tag{3.16}$$

Das Finden der maximalen Summe der Gewichte über die Distanzen und der daran beteiligten Surfel bestimmt gleichzeitig eine grob diskretisierte Parametrisierung der Ebenengleichung. Abbildung 3.2 zeigt die Ergebnisse der Hough-Transformation auf unterschiedlichen Oktalbaum Auflösungen.

Das Orientierungshistogramm wird nach Abschluss eines einzelnen Segmentierungsschrittes aktualisiert und die bereits segmentierten Surfel werden mit all ihren Gewichten daraus entfernt, so dass das alte Maximum im Orientierungshistogramm abgeschwächt wird und ein neues Maximum gefunden werden kann, das zu einer anderen Ebenengleichung führt.

Aufgrund dessen, dass auf die ausgewählten Punkte nochmals RANSAC angewendet wird, um weitere Ausreißer zu entfernen, können für die Diskretisierung des Parameterraumes für die Hough-Transformation verhältnismäßig grobe Schritte gewählt werden.

Zerlegung in Zusammenhangskomponenten

Die Zerlegung einer Ebene in ihre Zusammenhangskomponenten wird mit einem regelmäßigen Gitter gelöst. Hierzu müssen die planar angeordneten Kandidatenpunkte auf die Ebene projiziert und anschließend in ein regelmäßiges Gitter eingetragen werden. Die Gittergröße ist dabei relativ zur Dichte der Punkte zu wählen. Gitterzellen, in die mindestens ein Kandidatenpunkt fällt, werden als belegt markiert. Weiterhin enthält jede Gitterzelle nach dem Eintragen aller projizierten Kandidatenpunkte einen Index auf die beteiligten Punkte.

Nachdem die Zusammenhangskomponenten, wie in Abschnitt 2.8 beschrieben, ermittelt wurden, werden alle mit einer Mindestgröße *minsurfels*, gemessen in der Anzahl der beteiligten Surfel, bestimmt. Durch die beteiligten Punkte, die hierzu aus den zusammenhängenden Surfels extrahiert werden müssen, wird dann eine verfeinerte Ebene mittels RANSAC gesucht. Abbildung 3.2 zeigt an einem Beispiel die Zerlegung in Zusammenhangskomponenten.

Ausreißerfilterung mit RANSAC

Die bereits vorgefilterten Surfel werden dann an ein RANSAC-Verfahren übergeben, das in dem durch die Surfelmenge definierten Ausschnitt der Punktwolke die Ebenenparameter verfeinert und die Ausreißerzuordnung auf Punktebene vornimmt. Hierbei werden zuerst alle Punkte bestimmt, die zur zusammenhängenden Surfelmenge gehören, d.h. die in eine Oktalbaumzelle fallen, deren Surfel-Repräsentation in der zusammenhängenden Menge enthalten ist.

Als Menge von Beobachtungen B betrachten wir eine Menge von Punkten im dreidimensionalen Raum $B \subset \mathbb{R}^3$. Wenn RANSAC zur Detektion von Ebenen aus 3D-Punkten benutzt wird, werden drei Punkte oder ein Punkt mit Normale benötigt, um daraus das Modell zu generieren.

Aus drei Punkten $\boldsymbol{A}$, $\boldsymbol{B}$ und $\boldsymbol{C}$ kann eine Ebenengleichung in Hesse-Normal-Form generiert werden:

$$\langle \boldsymbol{n}, \boldsymbol{p} \rangle - d = 0. \tag{3.17}$$

Die Normale $\boldsymbol{n}$ berechnet sich als Kreuzprodukt aus den Verbindungsvektoren der Punkte:

$$\boldsymbol{n} = (\boldsymbol{B} - \boldsymbol{A}) \times (\boldsymbol{C} - \boldsymbol{A}). \tag{3.18}$$

Um die Distanz d der Ebene zum Ursprung zu bestimmen, wird einer der Punkte aus der Ebene in die oben genannte Gleichung 3.17 eingesetzt. Es ergibt sich:

$$d = \langle \boldsymbol{n}, \boldsymbol{A} \rangle . \tag{3.19}$$

Der Fehler des Modells M über die Einzelbeobachtungen B_i wird über den quadratischen Abstand der Punkte zur Ebene berechnet. Der Abstand der Beobachtungen zum Modell $dist(M, B_i)$ wird nur bis zu einem gewissen Grad als Fehler eingerechnet. Der Schwellwert für die maximale Distanz wird mit ϵ_{ransac} bezeichnet:

$$err_M = \sum_{i=1}^{n} (max(dist(M, B_i), \epsilon_{ransac}))^2. \tag{3.20}$$

Ein Punkt, dessen Distanz zur Ebene den Schwellwert überschreitet, wird als Ausreißer betrachtet und zählt damit nicht als Modellpunkt. Die Menge aller Modellpunkte $inlier_M$ ist damit definiert als die Menge aller Einzelbeobachtungen $B_i \in B$ mit einem Abstand $dist(M, B_i)$ zum Modell M der kleiner ist als der Distanzschwellwert ϵ_{ransac}:

$$inlier_M = \{B_i \mid dist(M, B_i) < \epsilon_{ransac}\} \subseteq B. \tag{3.21}$$

Nachdem alle RANSAC Iterationen durchlaufen wurden und das Modell mit dem kleinsten Fehler gefunden wurde, wird als Plausibilitätstest noch ein Abgleich mit den Parametern durchgeführt, die bei der Hough-Transformation das Ergebnis bildeten. Weichen die Parametersätze $(\boldsymbol{n}_{ht}, d_{ht})$ und $(\boldsymbol{n}_{ransac}, d_{ransac})$ für die Modelle zu sehr voneinander ab, dann wird das Ergebnis verworfen, da die Punktmenge sich offensichtlich nicht in ähnliche planare Segmente teilen lässt, wie die im Schritt der Hough-Transformation betrachtete Surfelmenge. Das heißt, dass die Vorsegmentierung der Hough-Transformation vermutlich nicht passend zur Punktmenge war. Auf diese Weise filtert der RANSAC-Schritt die fehlerhaften Segmentierungen, die durch die grobe Vorsegmentierung der Hough-Transformation eventuell gemacht werden. Der Test auf Ähnlichkeit erfolgt mit den Parametern α_{max}, für den Winkel zwischen den Normalen $\arccos \langle \boldsymbol{n}_{ht}, \boldsymbol{n}_{ransac} \rangle < \alpha_{max}$, und ϵ_{ht}, für die Ebenendistanz zum Ursprung $|d_{ht} - d_{ransac}| < \epsilon_{ht}$. Abbildung 3.2 zeigt Ausreißer und zugehörige Punkte anhand einer Beispielszene.

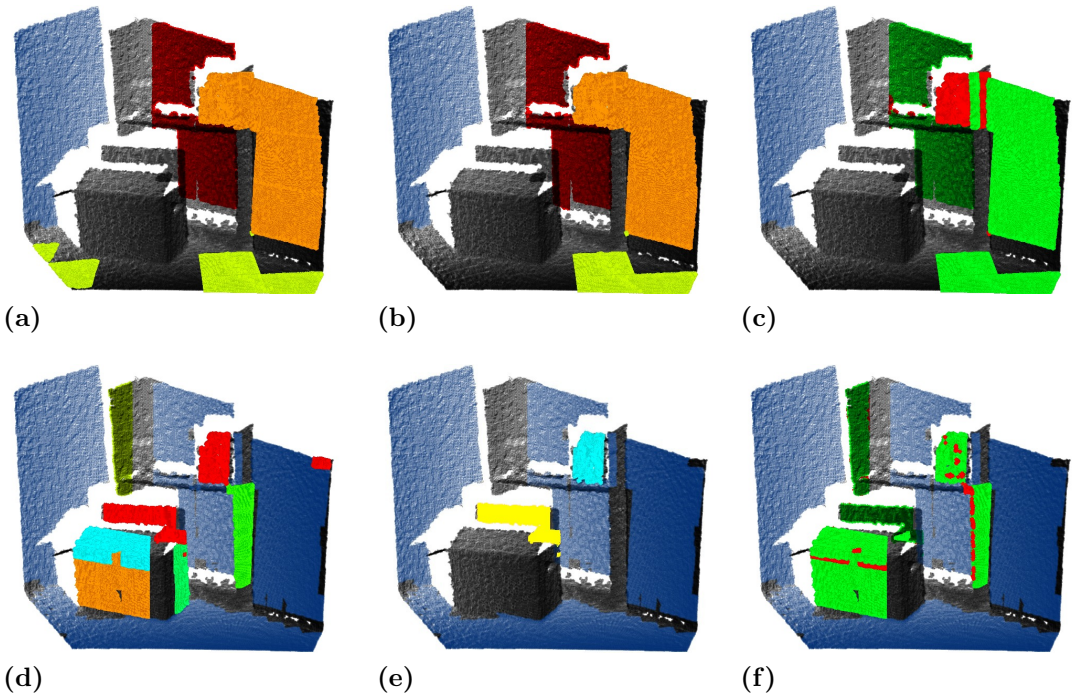


Abbildung 3.2: 3D Punktwolke mit teils vorsegmentierten Bereichen (blaue Punkte), noch unsegmentierten Punkten (schwarz) und aktuellen Teilergebnissen auf verschiedenen Auflösungsstufen (32cm obere Reihe und 16cm untere Reihe). Von links nach rechts werden zuerst die Ergebnisse der Hough-Transformation gezeigt, dann die Zerlegung in Zusammenhangskomponenten und auf den beiden rechten Bildern sind die Ergebnisse des RANSAC-Schrittes illustriert.
Auf Bild 3.2a wurden drei neue Ebenen detektiert (rote, orange und gelbe Markierung). Bild 3.2d zeigt die darauf folgende Auflösungsstufe und fünf mit der Hough-Transformation gefundene Segmente.
Auf Bild 3.2b wurde ausschließlich die gelbe Ebene geändert. Es wurde der Teil links unten im Bild entfernt, da die entsprechenden Oktalbaumknoten keine genügend große Zusammenhangskomponente bildeten. Bild 3.2e zeigt, wie das rot markierte Segment aus Bild 3.2d in zwei Zusammenhangskomponenten (gelb und cyan) aufgeteilt wurde. Die roten Punkte in der oberen rechten Ecke des linken unteren Bildes bilden keine ausreichend große Zusammenhangskomponente und werden verworfen. Die anderen Segmente aus der Hough-Transformation wurden in diesem Bild nicht markiert um die Aufteilung der roten Ebene in die beiden blau beziehungsweise gelb gefärbten Segmente zu verdeutlichen.
Die Punkte auf den beiden rechten Bildern (3.2c und 3.2f), die in einer Ebene mit durch RANSAC verfeinerten Parametern liegen, wurden grün markiert, die Ausreißer sind rot.

3.2.2 Zylinder

Auch bei der Segmentierung der Szene in zylindrische Formprimitive bzw. zylindrische Teilstücke, wird zuerst eine Hough-Transformation zur Vorsegmentierung verwendet. Dazu wurde der Ansatz von Rabbani et al. [11] und [19] erweitert. Anschließend wurden, ähnlich wie bei den Ebenen, Zusammenhangskomponenten bestimmt und die gewonnenen Hypothesen als Eingabe für das RANSAC-Verfahren genutzt.

Hough-Transformation für Zylinder

Die Repräsentation der Zylinderorientierung wird ebenfalls durch zwei Winkel realisiert, wie in Unterabschnitt 3.2.1 Gleichung 3.4. Als zusätzliche Parameter eines Zylinders werden noch Radius und ein Referenzpunkt benötigt. Da der Referenzpunkt in einer 2D Projektion entlang der Zylinderachse bestimmt wird, und seine genaue Position auf der Achse irrelevant ist, sind hierfür nur zwei weitere Parameter zu verwenden. Insgesamt sind also fünf Parameter zur eindeutigen Bestimmung eines Zylinders erforderlich. Da ein fünfdimensionaler Parameterraum ineffizient ist, wie zu Beginn des Abschnittes 2.5 bereits beschrieben, bietet sich ein zweistufiges Verfahren zur Umgehung dieses Problems an.

Rabbani et al. [20] [11] haben ein Verfahren entwickelt, mit dem in 3D Punktwolken mit Normaleninformation mittels zweistufiger Hough-Transformation Zylinder detektiert werden können. Hierzu werden zuerst die Normalen der gegebenen Punkte transformiert, so dass alle für diese Richtung möglichen Zylinderorientierungen in ein Kugelhistogramm eingetragen werden, ähnlich zur Bestimmung der Ebenenorientierung mit Normalen. Der entscheidende Unterschied ist hierbei, dass das Histogramm nicht an der Stelle erhöht wird, die der Normalen entspricht, sondern statt dessen wird das Kugelhistogramm dort erhöht, wo die Orientierung im Histogramm orthogonal zur Punktnormalen ist. Damit entsteht im Kugelhistogramm für jede Normale ein Kreis. Die mehrfachen Schnittpunkte dieser Kreise sind die Kandidaten der Zylinderorientierung für den nächsten Schritt.

Nachdem die Zylinderorientierung bestimmt ist, kann entlang dieser Achse eine zweidimensionale Projektion aller Punktkandidaten und ihrer Normalen vorgenommen werden. Auf diese orientierten 2D Punkte wird ein Verfahren zur Detektion von Kreisen angewendet. Dazu wird abhängig vom Radius des Kreises und der Normalen des

Punktes der mögliche Kreismittelpunkt berechnet und in ein dreidimensionales Histogram eingetragen. Dabei stehen zwei Dimensionen für die 2D Koordinaten des Kreismittelpunktes und eine Dimension für den Zylinderradius.

Mit den Ergebnissen dieses Histogramms und der Zylinderorientierung aus dem Schritt davor ist der Zylinder vollständig bestimmt.

Surfelpaare als Beobachtungen Im Gegensatz zu Rabbanis Ansatz [11] generiert der hier vorgestellte Algorithmus Surfelpaare, aus denen alle nötigen Zylinderparameter geschätzt werden können. Eine ähnliche Idee zur Nutzung von Punktepaaren findet sich auch in Ballards Arbeit [1] zur Verallgemeinerung der Hough-Transformation, um beliebige Formen in 2D Graustufenbildern zu finden.

Um nicht die Paare aus allen Surfeln S zu bilden, wurde eine maximale Distanz $surfeldist_{max}$ in aktueller Surfelgröße zwischen zwei Surfeln festgelegt, aus denen ein Paar $\mathfrak{s}_{(i,j)} \in \mathcal{S}$ generiert wird.

$$\mathcal{S} = \{\mathfrak{s}_{(i,j)} = (s_i, s_j) \in S \times S \mid dist(s_i, s_j) < surfeldist_{max}\} \tag{3.22}$$

Damit kein Paar doppelt auftritt, werden die Surfel markiert, deren Nachbarn schon alle zur Paarbildung genutzt wurden.

Hough-Transformation zur Detektion der Orientierung eines Zylinders Bilden zwei Surfel ein stabiles Paar, so wird eine Orientierungshypothese $\boldsymbol{o}_{(i,j)}$ aus den Normalen $\boldsymbol{n}_i$, $\boldsymbol{n}_j$ der beteiligten Surfel berechnet. Hierzu wird das Kreuzprodukt der beiden Normalen gebildet und normiert:

$$\boldsymbol{o}_{(i,j)} = \frac{\boldsymbol{n}_i \times \boldsymbol{n}_j}{|\boldsymbol{n}_i \times \boldsymbol{n}_j|}. \tag{3.23}$$

Die so pro Surfelpaar $\mathfrak{s}_{(i,j)}$ erstellte Orientierungshypothese wird dann in das Orientierungshistogramm mit entsprechend gewichteten Ungenauigkeiten eingetragen. Das Gewicht berechnet sich so, wie bei der Eintragung der Normalen für die Orientierung der Ebenen. Die Terme für die Krümmung $\gamma_{(i,j)}$ und für die Anzahl der beteiligten Punkte $N_{(i,j)}$ müssen entsprechend angepasst werden. Für eine Orientierung $\boldsymbol{o}_l$ in einer Umgebung, gegeben durch die maximale Winkelabweichung α_{max}, um die Orientierung $\boldsymbol{o}_{(i,j)}$ berechnet sich das Gewicht wie folgt:

$$w_l = N_{(i,j)} \cdot \left(1 - \frac{\gamma_{(i,j)}}{\gamma_{\max}}\right) \cdot \frac{\left(\left|\left\langle \boldsymbol{o}_{(i,j)}, \boldsymbol{o}_l \right\rangle\right| - \cos(\alpha_{max})\right)}{\cos(\alpha_{max})}. \tag{3.24}$$

Die Krümmung $\gamma_{(i,j)}$ des Surfelpaares wird auf das Maximum der Krümmung der beiden Surfel gesetzt,

$$\gamma_{(i,j)} = \max(\gamma_i, \gamma_j). \tag{3.25}$$

Die Anzahl der beteiligten Punkte $N_{(i,j)}$ ist die Summe der Punkte der beiden Surfel, die das Paar bilden,

$$N_{(i,j)} = N_i + N_j. \tag{3.26}$$

Es ergibt sich die Position des Eintrags für die Histogrammfelder aus den Winkeln θ_{o_l} und ϕ_{o_l} für die Orientierung wie in Gleichung 3.5, so dass die Einträge für ein Surfelpaar im Histogramm dann folgende Form haben:

$$\forall o_l : his(\theta_{o_l}, \phi_{o_l}) \leftarrow his(\theta_{o_l}, \phi_{o_l}) \cup \{(w_l, \mathfrak{s}_{(i,j)})\}. \tag{3.27}$$

Das Histogrammfeld mit der höchsten Summe der Gewichte beinhaltet demnach alle Punktepaare $s_{(i,j)}$, die eine ähnliche Zylinderorientierung unterstützen. Aus diesen Punktepaaren werden im Anschluss der Radius und ein Punkt auf der Achse berechnet.

Rotation und Projektion ins 2D Hierzu wird zuerst eine Rotation bestimmt, welche die Zylinderachse parallel zu einer der Koordinatenachsen transformiert, so dass auf einfache Weise eine 2D-Projektion durch Parallelprojektion zur Zylinderachse vorgenommen weden kann. Im Anschluss daran werden die 2D-Normalen und 2D-Positionen der Surfelpaare dazu genutzt, den Radius und anschließend den Mittelpunkt eines Kreises zu bestimmen. Dieser Kreis ist dann eine Projektion des detektierten Zylindermantels.

Die Rotation wird bestimmt, indem der Winkel α_{rot} zwischen der ausgewählten Koordinatenachse und der Hauptachse des Zylinders und eine entsprechende Rotationsachse ermittelt wird. Den Winkel zwischen der Zylinderachse o und einer der Koordinatenachsen e wird durch das Skalarprodukt ermittelt. Da beide Achsen auf die Einheitslänge normiert sind, entfällt der entsprechende Normierungsterm.

$$\alpha_{rot} = \arccos(\langle o, e \rangle) \tag{3.28}$$

Die Rotationsachse $\boldsymbol{a}$ kann mittels Kreuzprodukt berechnet werden.

$$\boldsymbol{a} = (a_x, a_y, a_z)^T = o \times e \tag{3.29}$$

Daraus ergibt sich die Rotation $R(\boldsymbol{a}, \alpha)$ abhängig von der Dreahachse $\boldsymbol{a}$ und dem Drehwinkel α.

Hough-Transformation zur Bestimmung von Radius und Kreismittelpunkt in der zweidimensionalen Projektion Nachdem alle Surfelpaar Schwerpunkte und Normalen rotiert und projiziert wurden, können damit die Mittelpunkte und daraus die Radii bestimmt werden. Aus einem 2D Surfel mit Schwerpunkt $\boldsymbol{A} = (A_x, A_y)^T$ und Normale $\boldsymbol{N}_1 = (N_{1x}, N_{1y})^T$ kann die 2D Gerade bestimmt werden, auf welcher der entsprechende Kreismittelpunkt liegt:

$$\boldsymbol{A} + \lambda_1 \boldsymbol{N}_1 = \boldsymbol{X}. \tag{3.30}$$

Die zweite Gerade duch den Schwerpunkt des zweiten Surfels $\boldsymbol{B} = (B_x, B_y)^T$ mit Normale $\boldsymbol{N}_2 = (N_{2x}, N_{2y})^T$ wird durch die folgende Geradengleichung bestimmt:

$$\boldsymbol{B} + \lambda_2 \boldsymbol{N}_2 = \boldsymbol{Y}. \tag{3.31}$$

Da der Kreismittelpunkt ein gemeinsamer Punkt beider Geraden ist, gilt $\boldsymbol{X} = \boldsymbol{Y}$. Gleichsetzen der beiden obigen Gleichungen ergibt:

$$\boldsymbol{A} - \boldsymbol{B} + \lambda_1 \boldsymbol{N}_1 - \lambda_2 \boldsymbol{N}_2 = 0. \tag{3.32}$$

Aufteilung der Gleichung in ihre Komponenten und Auflösen nach λ_1 ergibt:

$$\lambda_1 = \frac{B_x + \lambda_2 N_{2x} - A_x}{N_{1x}}, \text{und} \tag{3.33}$$

$$\lambda_1 = \frac{B_y + \lambda_2 N_{2y} - A_y}{N_{1y}}. \tag{3.34}$$

Gleichsetzen über λ_1 und Auflösen nach λ_2 führt zu:

$$\lambda_2 = \frac{N_{1x}(B_y - A_y) + N_{1y}(A_x - B_x)}{N_{2x}N_{1y} - N_{2y}N_{1x}}. \tag{3.35}$$

Damit lassen sich die Koordinaten für den Mittelpunkt $\boldsymbol{m} = (m_x, m_y)$ durch Einsetzen von λ_2 in die Geradengleichung 3.31 berechnen:

$$m_x = B_x + \lambda_2 N_{2x}, \tag{3.36}$$

$$m_y = B_y + \lambda_2 N_{2y}. \tag{3.37}$$

Der gemittelte Abstand der Schwerpunkte $\boldsymbol{A}$ und $\boldsymbol{B}$ zum Mittelpunkt $\boldsymbol{m}$ entspricht dann dem geschätzten Radius $r_{(i,j)}$ des Kreises durch die 2D Projektion des Surfelpaares $\mathfrak{s}_{(i,j)}$:

$$r_{(i,j)} = \frac{dist(\boldsymbol{A}, \boldsymbol{m}) + dist(\boldsymbol{B}, \boldsymbol{m})}{2}. \tag{3.38}$$

Unterscheidet sich der Abstand der beiden Schwerpunkte zum Mittelpunkt zu stark, so wird das Paar nicht weiter betrachtet und in kein Histogramm eingetragen.

Alle anderen Surfelpaare $\mathfrak{s}_{(i,j)}$ werden nach dem für sie berechneten Radius $r_{(i,j)}$ in ein Histogramm eingetragen. Das Histogramm hat dabei die gleiche Schrittweite ϵ_{ht}, wie schon das Histogramm für die Distanz einer Ebene zum Ursprung. Auch die Gewichte der benachbarten Histogrammzellen berechnen sich wie beim Distanzhistogramm in Unterabschnitt 3.2.1, wobei *dev* die Ungenauigkeit benennt, mit welcher der geschätzte Radius in das Histogramm eingetragen werden soll. r_l bezeichnet den Radius eines benachbarten Histogrammfeldes und der Unterschied zwischen zwei Radii wird mit $dist(\cdot,\cdot)$ bezeichnet. Ist das Gewicht nicht positiv, so wird ein Surfelpaar nicht in das entsprechende Histogrammfeld eingetragen:

$$w_l = \frac{dev - dist(r_l, r_{(i,j)})}{dev}, \tag{3.39}$$

$$\forall r_l : his(r_l) \leftarrow his(r_l) \cup \{(w_l, \mathfrak{s}_{(i,j)})\}. \tag{3.40}$$

Das Histogrammfeld mit der größten Summe der Gewichte führt erneut zu der Parameterkonfiguration mit dem besten Radius.

Um für alle Surfelpaare $\mathfrak{s}_{(i,j)}$ des maximal bewerteten Radius einen gemeinsamen Mittelpunkt zu finden, wird ein zweidimensionales Histogramm verwendet. Jeder Mittelpunkt $\boldsymbol{m}_{(i,j)}$ eines zweidimensionalen Kreises durch die Schwerpunkte eines Surfelpaares, wird entsprechend gewichtet in das zweidimensionale regelmäßige Histogramm eingetragen:

$$w_{(l,k)} = \frac{dev - dist((m_l, m_k), \boldsymbol{m}_{(i,j)})}{dev}, \tag{3.41}$$

$$\forall (m_l, m_k) : his((m_l, m_k)) \leftarrow his((m_l, m_k)) \cup \{(w_{(l,k)}, \mathfrak{s}_{(i,j)})\}. \tag{3.42}$$

Das Histogrammfeld mit der größten Summe der Gewichte beinhaltet gleichzeitig auch die Surfelpaare mit dem am besten bewerteten Mittelpunkt. Dieser Mittelpunkt wird durch Setzen einer dritten Komponente, je nach paralleler Koordinatenachse, und Anwendung der negativen Rotation $R(\boldsymbol{a}, \alpha)^{-1} = R(\boldsymbol{a}, -\alpha) = R(\boldsymbol{a}, \alpha)^T$ in einen 3D-Punkt transformiert.

Alle Surfelpaare, die im Histogrammfeld mit der größten Summe der Gewichte liegen, werden im Anschluss an die Mittelpunkttransformation aus dem Orientierungshistogramm ausgetragen, damit sie dort bei der nächsten Iteration nicht mehr zu finden sind.

Ausreißerfilterung mit RANSAC

Wenn die gegebenen 3D-Punkte mit Normaleninformationen vorliegen, so reichen zwei Punkte aus, um die fünf Parameter eines Zylinders zu bestimmen (siehe hierzu Unterabschnitt 3.2.2). Seien $\boldsymbol{p}_A$ und $\boldsymbol{p}_B$ zwei Punkte mit den Normalen $\boldsymbol{N}_{\boldsymbol{p}_A}$ und $\boldsymbol{N}_{\boldsymbol{p}_B}$. Dann wird aus diesen beiden Punkten zuerst eine Zylinderorientierung $\boldsymbol{o}$ generiert:

$$\boldsymbol{o}_{A,B} = \frac{\boldsymbol{N}_{\boldsymbol{p}_A} \times \boldsymbol{N}_{\boldsymbol{p}_B}}{\left|\boldsymbol{N}_{\boldsymbol{p}_A} \times \boldsymbol{N}_{\boldsymbol{p}_B}\right|}. \tag{3.43}$$

Eine Projektion der Punkte und Normalen entlang dieser Achse auf eine Ebene, wie auch schon bei der Hough-Transformation, ermöglicht eine Berechnung des Mittelpunktes im zweidimensionalen. Für die Normalen $\boldsymbol{N}_1 = (N_{1x}, N_{1y})^T$ und $\boldsymbol{N}_2 = (N_{2x}, N_{2y})^T$, sowie Punkte $\boldsymbol{A} = (A_x, A_y)^T$ und $\boldsymbol{B} = (B_x, B_y)^T$ ergeben sich wieder die Formeln für dem Kreispittelpunkt $\boldsymbol{m}$ und den Radius $r_{(i,j)}$ des Kreises durch die 2D Projektion des Surfelpaares $\mathfrak{s}_{(i,j)}$:

$$m_x = B_x + \lambda_2 N_{2x} \tag{3.44}$$

$$m_y = B_y + \lambda_2 N_{2y} \tag{3.45}$$

$$r_{(i,j)} = \frac{dist(\boldsymbol{A}, \boldsymbol{m}) + dist(\boldsymbol{B}, \boldsymbol{m})}{2}. \tag{3.46}$$

Sollten keine Normaleninformationen vorliegen, so gibt es unterschiedliche Verfahren zur Modellgenerierung. Beder und Förstner [2] stellen in ihrer Arbeit hierzu mehrere Möglichkeiten dar. Zur Berechnung aller Parameter in einer geschlossenen Lösung sind demnach mindestens fünf Punkte nötig.

3.3 Multiresolutionsstrategie

Die Idee der Segmentierung von grob nach fein besteht darin, dass auf einer groben Auflösungsstufe der Punktwolke bereits grobe Strukturen erkannt werden, deren Punkte auf feineren Auflösungsstufen nicht mehr weiter betrachten müssen. Es wird der größtmögliche Kontext für die Segmentierung genutzt, so dass für die Surfelnormalen eine robustere Schätzung möglich ist. Algorithmus 3.1 bietet einen Überblick des Verfahrens.

Sowohl der Distanzschwellwert für RANSAC ϵ_{ransac}, als auch der Distanzschwellwert für die Hough-Transformation ϵ_{ht}, werden je nach aktuell bearbeiteter Oktalbaumtiefe angepasst. Je kleiner die Oktalbaumzellen und damit auch die Surfel, desto kleiner wird auch der jeweilige Distanzschwellwert gewählt.

Auf jeder Auflösungsstufe werden zuerst alle Surfel gegen die bereits gefundenen Formprimitive getestet und gegebenenfalls diesen zugeordnet. Es wird sowohl die Normaleninformation dieser Auflösungsstufe, als auch die Distanz zwischen Surfelschwerpunkt und Oberfläche des Formprimitivs betrachtet. Alle Surfel, deren Normalen zum Formprimitiv passen, und deren Distanzen gering genug sind, werden gemeinsam mit der bereits bestehenden Zusammenhangskomponente des Segments auf Zusammenhang geprüft und bei positivem Test dem Segment zugeordnet. Dieser Zuordnungsschritt dient sowohl der Eliminierung von Surfeln, deren Punkte durch Segmentierung auf groberer Auflösung bereits teilweise erklärt sind, als auch der Verfeinerung eines bereits gefundenen Segments.

Für den Distanztest gilt die Positionsabweichung ϵ_{ransac}, die bei der Erstellung des Formprimitivs festgelegt wurde. Hiermit wird erreicht, dass große und grobe Primitive, die auf grober Auflösungsstufe gefunden werden, Abweichungen relativ zu ihrer Größe zulassen können.

Alle Surfel, die sowohl von ihrer Orientierung, als auch von ihrer Position zum geprüften Formprimitiv passen, werden gemeinsam auf Zusammenhang geprüft. Hierzu wird das Zusammenhangsgitter, das während der Konstruktion des Segments angelegt wird, auf die aktuelle Auflösungsstufe vergrößert und die in Frage kommenden Surfel werden dann anhand ihres Schwerpunktes in das neue Gitter projiziert. Alle Surfel, die in die Zusammenhangskomponente fallen, die schon zuvor im Gitter eingetragen war, oder diese lückenlos erweitern, werden dem Segment zugeordnet. Dadurch, dass die Kantenlänge jeder Gitterzelle und die Größe der Surfels identisch sind, aber die Lage der Ebene beliebig ist, repräsentiert die Projektion des Surfelschwerpunktes nicht alle im Surfel liegenden Punkte. Daher können Diskretisierungsfehler entstehen, die koplanare, sich berührende Ebenensegmente zur Folge haben. Nachdem alle Oktalbaumstufen abgearbeitet wurden, soll ein Nachverarbeitungsschritt diese Ebenensegmente zuordnen.

Wird ein Surfel einem Segment zugewiesen, so müssen alle Punkte des Surfels gegen das Segment getestet werden. Hierzu wird geprüft, ob der Punkt wirklich innerhalb der maximalen Distanz ϵ_{ransac} zur Oberfläche des Formprimitivs liegt. Ist das der Fall, so wird der entsprechende Punkt des Surfels dem Segment zugeordnet. Werden

mehr als $ratio_{min}\%$ der Punkte eines Surfels einem Segment zugeordnet, so werden alle Kindknoten des dem Surfel entsprechenden Oktalbaumknotens als bereits segmentiert markiert. Auf diese Weise markierte Oktalbaumknoten werden bei der Segmentierung auf feineren Auflösungen nicht mehr betrachtet.

Nachdem alle Surfel entweder einem Primitiv zugeordnet werden konnten, oder erfolglos gegen alle Primitive getestet wurden, werden die nicht zugeordneten Surfel mit den oben beschriebenen Verfahren weiter segmentiert.

Algorithmus 3.1 : Multi-Resolutions Segmentierung

Data : Beobachtung $B = \boldsymbol{b}_1, .., \boldsymbol{b}_g$;
Result : Modelle M;
Beobachtungen in Oktalbaum T eintragen;
Auflösungsstufen S für jede Oktalbaum Tiefe d generieren;
Normale für alle Surfel berechnen;
for $d := 1 \,..\, d_{max}$ **do**
    ordne Surfel $s \in S(d)$ den Modellen in M zu;
    ordne alle Punkte der zugeordneten Surfel zu entsprechendem Modell zu;
    repeat
        $\hat{m}_s$:= Hough-Transformation($S(d)$);
        $\hat{m}_s$:= suche größte Zusammenhangskomponente $\hat{m}_s$;
        m_p := extrahiere Punkte aus Surfelmenge($\hat{m}_s$);
        $\hat{m}_p$:= RANSAC(m_p);
        if $|\hat{m}_p| >=$ *minPoints-Schwellwert* **then**
            $M := M \cup \{\hat{m}_p\}$;
        end
    until $|\hat{m}_p| >=$ *minPoints-Schwellwert* ;
end
M := Nachbearbeitung(B, M);

3.4 Nachbearbeitung der Segmentierung

In diesem Abschnitt werden die Nachbearbeitungsschritte der Segmentierung erläutert. Nachdem alle Oktalbaum-Auflösungsstufen abgearbeitet sind, werden zur Verbesserung der Zusammenhangskomponenten, die aufgrund von grober Auflösung nicht immer korrekt erkannt und zugeordnet werden können, koplanare Ebenensegmente miteinander verschmolzen. Manche Punkte können aufgrund von ungünstiger

Lage im Oktalbaum nicht den gefundenen Primitiven zugeordnet werden und müssen ohne Berücksichtigen von Normaleninformationen gegen die Primitive geprüft werden.

3.4.1 Verschmelzen von koplanaren Ebenensegmenten

Nachdem alle Auflösungsstufen abgearbeitet sind, werden die gefundenen Segmente auf Koplanarität geprüft. Das ist deswegen sinnvoll, weil die Zusammenhangstests während des Durchlaufens der Auflösungsstufen des Oktalbaumes nur auf Surfels und damit nur auf groben Auflösungsstufen, durchgeführt wird. Beim Zusammenhangstest auf grober Auflösung wird das Gitter in der Ebene mit einer Kantenlänge erstellt, die exakt der Kantenlänge jedes Volumens entspricht, welches die kleinsten beteiligten Surfel repräsentieren. Das kann zu Diskretisierungsfehlern und damit zu falschen Zusammenhangskomponenten führen. Das heißt, dass nach der Bearbeitung aller Auflösungsstufen, aneinander angrenzende Ebenensegmente existieren können, die eine sehr ähnliche Orientierung haben. Ziel dieses Schrittes der Nachverarbeitung ist die Verschmelzung solcher koplanaren Ebenensegmente, die nahe beieinander liegen.

Hierzu werden zuerst die Normalen der Ebenen (i, j) paarweise miteinander verglichen und nur Ebenenpaare weiter betrachtet, deren Normalen ($\boldsymbol{n}_i$, $\boldsymbol{n}_j$) in einem spitzen Winkel zueinander stehen,

$$|\langle \boldsymbol{n}_i, \boldsymbol{n}_j| >| \leq \cos(\alpha_{max}). \tag{3.47}$$

Es wird der gleiche Schwellwert für Winkel α_{max} benutzt, wie schon für die Prüfung der Ähnlichkeit von zwei Ebenen, die durch Hough-Transformation beziehungsweise RANSAC erzeugt wurden 3.2.1.

Anschließend wird geprüft, ob der Schwerpunkt einer der beiden Ebenen eine Distanz zur jeweils anderen Ebene besitzt, die so niedrig ist wie der Distanzschwellwert der Ebene. Nachdem die trivialen Tests abgeschlossen sind, muss noch ein Test auf Zusammenhang entscheiden, ob die Kandidatenebenen miteinander verbunden werden können. Dazu werden nacheinander alle Punkte einer Ebene in die Zusammenhangskarte der anderen Ebene projiziert. Falls ein Punkt auf ein bereits belegtes Gitterfeld abgebildet wird, so ist der Zusammenhang der beiden Kandidatenebenen belegt und alle Punkte werden in eine Ebene zusammen gefügt.

3.4.2 Verteilung der übrigen Punkte

Punkte in Randlagen, die während der Haupt-Segmentierung nicht zugeteilt werden konnten, müssen noch in die Segmentierung eingearbeitet werden. Das kann passieren, da die Diskretisierung des Oktalbaumes nicht schon die Segmentierung enthält, sondern die Punkte eines Surfels am Rand eines Segments meist nicht alle in ein Segment passen. Ebenso ist es möglich, dass Surfel, die am Rand eines Segmentes liegen keine stabile Normale bilden. In diesen Fällen ist eine Verteilung der noch nicht zugeteilten Punkte nötig.

Hierzu werden alle Einzelpunkte, die noch nicht zugeteilt sind, ohne Berücksichtigung jeglicher Normaleninformationen gegen die gefundenen Formprimitive geprüft und dem Segment mit geringster Distanz zugeordnet, falls diese Distanz unterhalb des Schwellwertes für maximale Distanz des Primitivs liegt.

Um die Zuteilung der Punkte an den Rändern zwischen mehreren Ebenen zu verbessern, werden für jeden Punkt, der in mehrere Ebenen passen könnte, zuerst alle Kandidatenebenen berechnet. Ein Punkt, der mehr als zwei entsprechende Ebenen als Kandidaten hat, wird entsprechend des geringsten Abstandes zugeordnet. Da die Normaleninformation nahe eines solchen Randes aufgrund der Verletzung der Planaritäts-Voraussetzung unzuverlässig ist, wird nur die Distanz zur Ebene als Kriterium betrachtet.

Gibt es für einen Punkt genau zwei benachbarte Ebenen, so führt eine Zuordnung nach Distanz zu Fehlern im Bereich der Kanten. Siehe Abbildung 3.3 für ein Beispiel. Statt dessen wird für jedes benachbarte Ebenenpaar (s_1, s_2) eine Mittelebene generiert, anhand der die Punkte zwischen den Ebenen aufgeteilt werden. Die Orientierung der Mittelebene berechnet sich wie folgt:

$$\boldsymbol{n}_{cut} = \left(\frac{\boldsymbol{n}_{s_1} + \boldsymbol{n}_{s_2}}{2} \right) \times (\boldsymbol{n}_{s_1} \times \boldsymbol{n}_{s_2}). \tag{3.48}$$

Die Distanz dieser Ebene zum Ursprung berechnet sich durch einen Punkt auf der Schnittgerade. Hierzu wird der Schwerpunkt $\boldsymbol{p}_1$ der ersten Ebene $\langle \boldsymbol{n}_{s_1}, \boldsymbol{p} \rangle - d_{s_2} = 0$ entlang der Ebene auf die andere Ebene $\langle \boldsymbol{n}_{s_2}, \boldsymbol{p} \rangle - d_{s_2} = 0$ projiziert. Der Richtungsvektor der Verschiebung von Punkt $\boldsymbol{p}_1$ errechnet sich aus den Ebenennormalen:

$$\boldsymbol{v} = \boldsymbol{n}_{s_1} \times (\boldsymbol{n}_{s_1} \times \boldsymbol{n}_{s_2}) \,. \tag{3.49}$$

Die Länge der Verschiebung λ ergibt sich aus dem Skalarprodukt,

$$cos(\beta) = \langle \boldsymbol{n}_{s_2}, (\boldsymbol{n}_{s_1} \times (\boldsymbol{n}_{s_1} \times \boldsymbol{n}_{s_2})) \rangle \,, \tag{3.50}$$

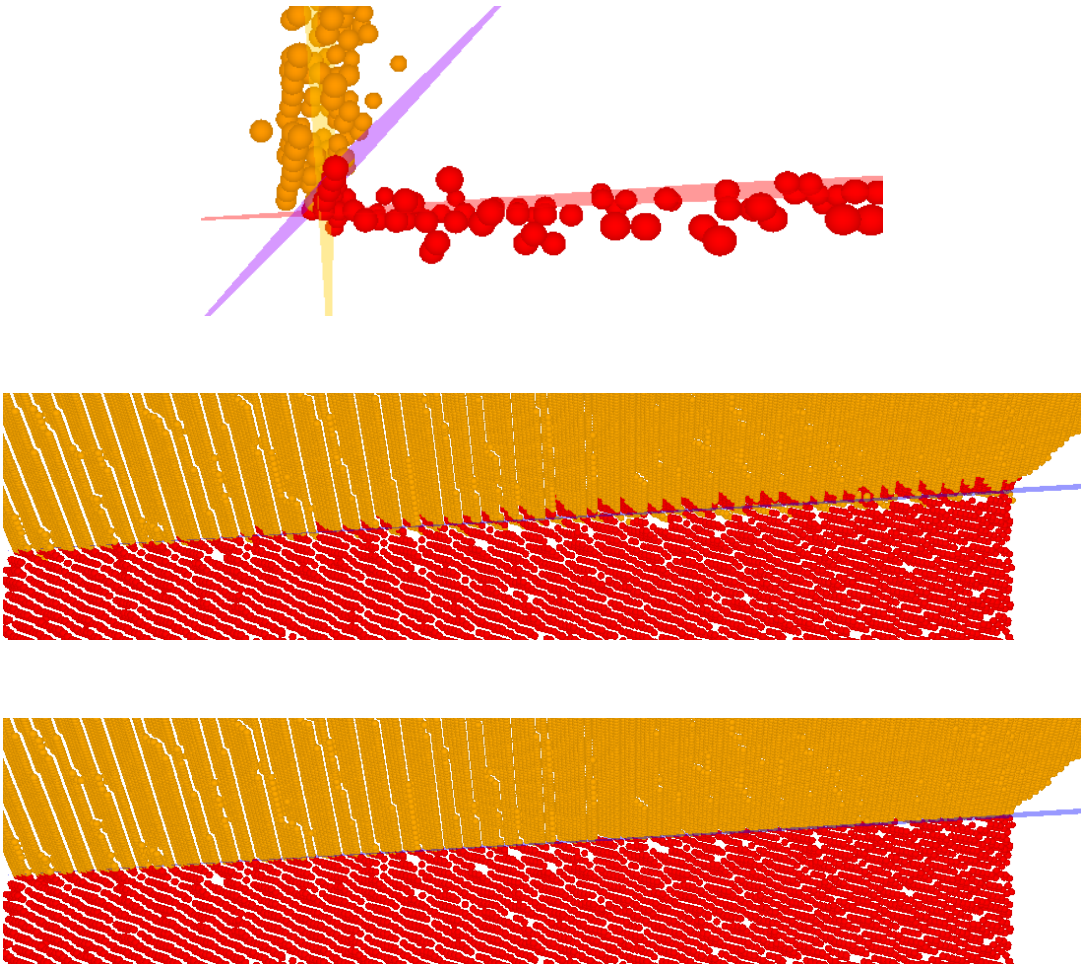


Abbildung 3.3: Aufteilung der Punkte zwischen zwei benachbarten Ebenen. Das obere Bild illustriert die Lage der Mitteleben und die Aufteilung der Punkte. Auf dem mittleren Bild wurden die Punkte ausschließlich nach der geringsten Distanz zur Ebene zugeordnet, was zu fehlerhaft zugeordneten Punkten entlang der Kante führte. Das untere Bild zeigt die gleiche Kante mit Aufteilung der Punkte nach Lage zur Mittelebene.

und der Definition des Kosinus im rechtwinkligen Dreieck,

$$cos(\beta) = \frac{\langle \boldsymbol{n}_{s_2}, \boldsymbol{p}_1 \rangle - d_{s2}}{\lambda}. \tag{3.51}$$

Durch Gleichsetzen und Umformung der Gleichungen wird eine Formel für die Länge der Verschiebung λ aufgestellt:

$$\lambda = \frac{\langle \boldsymbol{n}_{s_2}, \boldsymbol{p}_1 \rangle - d_{s_2}}{\langle \boldsymbol{n}_{s_2}, (\boldsymbol{n}_{s_1} \times (\boldsymbol{n}_{s_1} \times \boldsymbol{n}_{s_2})) \rangle}. \tag{3.52}$$

Die Distanz der Mittelebene d_{cut} errechnet sich mit den zuvor bestimmten Größen durch Einsetzen in die Ebenengleichung:

$$\langle \boldsymbol{n}_{cut}, \boldsymbol{p}_1 - \lambda \boldsymbol{v} \rangle - d_{cut} = 0. \tag{3.53}$$

Liegen die Schwerpunkte der beiden Ebenen auf unterschiedlichen Seiten der Mittelebene, so werden die Punkte im betrachteten Schnittbereich so verteilt, dass sie der Ebene zugeteilt werden, mit dessen Schwerpunkt sie auf der selben Seite liegen. Sollten die Schwerpunkte auf der gleichen Seite der Mittelebene liegen, so ist über dieses Kriterium keine Verbesserung zu erzielen, und die Punkte werden der Ebene zugeordnet, zu der sie die geringere Distanz besitzen. Abbildung 3.3 zeigt je ein Beispiel für die Punktzuordnung nach Lage zur Mittelebene und für die Zuordnung nach der geringeren Distanz.

3.5 Darstellung und Weiterverarbeitung

In diesem Abschnitt werden die Verfahren zur Berechnung der Texturen und Reliefkarten beschrieben, die zur Darstellung verwendet werden.

3.5.1 Belegtheitskarten und Texturen

Als Belegtheitskarte wird ein regelmäßiges Gitter bezeichnet, das anzeigt, ob eine Gitterzelle als Oberflächenelement eines Formprimitivs durch einen Punkt der zugrundeliegenden Punktwolke gestützt wird, oder nicht.

Eine Belegtheitskarte wird sehr ähnlich bestimmt, wie das Gitter zur Bestimmung der Zusammenhangskomponenten. Die Auflösung der Belegtheitskarte wird durch

die Anzahl der Punkte bestimmt, wobei eine Gleichverteilung der Punkte auf der Oberfläche angenommen wird. Als Korrektur zu dieser Approximation wurde ein zusätzlicher Faktor η eingefügt:

$$texelsize = \eta \sqrt{\frac{x \cdot y}{N}}. \tag{3.54}$$

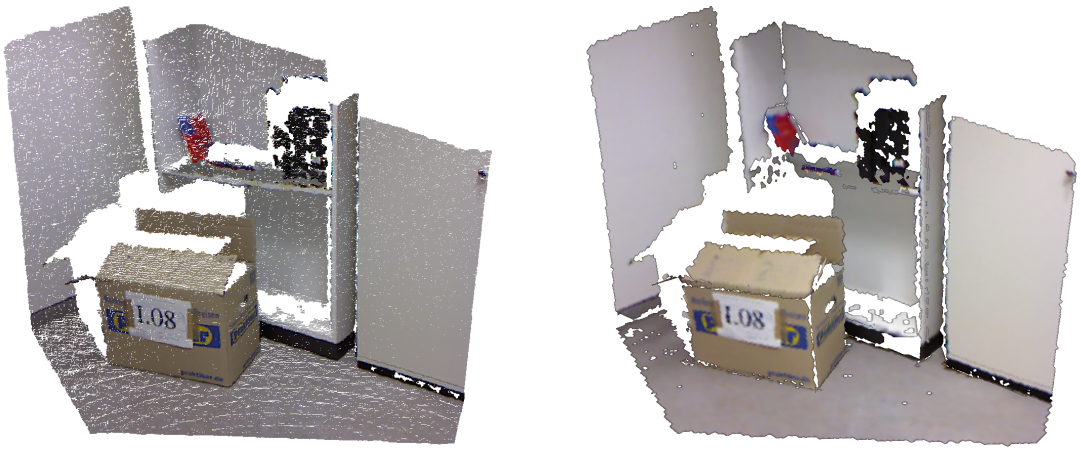


Abbildung 3.4: Kolorierte Kinect Punktwolke (links) und ihre texturierte planare Segmentierung (rechts).

Texturen sind Belegtheitskarten, die zusätzlich mit Farbinformationen gefüllt werden. Um die Farbe der zugrundeliegenden Punktwolke möglichst gut zu approximieren, wird die Farbe für eine Gitterzelle aus allen Punkten gemittelt, die das Gitterzellenzentrum als nächsten Nachbarn haben. Die Auflösung des Gitters wird wie bei den Belegtheitskarten ermittelt.

3.5.2 Reliefkarten

Bei Reliefkarten handelt es sich um Texturen, die Höhenabweichungen vom Formprimitiv kodieren. In der Reräsentation wird ein Skalierungsfaktor abgespeichert, der eine approximative Rückwärts-Abbildung zwischen Reliefkarte und zugrundeliegender Punktwolke ermöglicht. Allerdings werden Punkte, die in eine Gitterzelle fallen gemittelt und ihre Position durch das Gitter diskretisiert.

Der zu jeder Ebene gespeicherte Wert für die maximale Abweichung der zugehörigen Punktmenge ϵ_{plane} wird dazu genutzt, die Distanz eines jeden Punktes zur Ebene

zu diskretisieren. Zusätzlich wird betrachtet, auf welcher Seite der Ebene ein Punkt liegt. Diese Information wird in einem Vorzeichen kodiert. Die Punkte, die auf der Seite der Ebene liegen, auf der auch der Ursprung liegt, bekommen ein positives Vorzeichen und umgekehrt. Die Punkte der Ebene werden ins Koordinatensystem der Ebene transformiert, um dort ins Gitter projiziert zu werden. Damit ergibt sich eine Abbildung aus der Distanz $dist(\boldsymbol{p}, plane)$ eines transformierten Punktes $\boldsymbol{p}$ zur Ebene $plane$ mit Vorzeichen $sign(\boldsymbol{p}, plane)$ in einen Wertebereich $[0, 1]$ für die Reliefkarte:

$$r(\boldsymbol{p}) = \frac{dist(\boldsymbol{p}, plane)}{\epsilon_{plane}} \cdot \text{sign}(\boldsymbol{p}, plane) + \frac{\epsilon_{plane}}{2}. \tag{3.55}$$

Alle Punkte werden in das Gitter der Reliefkarte projiziert. Dort wird die Summe über alle Werte $r(\boldsymbol{p})$ in dieser Zelle gebildet und durch die Anzahl der Punkte in dieser Zelle $\#(\boldsymbol{p} \in (i, j))$ geteilt. Dadurch wird über alle Distanzwerte in dieser Gitterzelle das Mittel gebildet:

$$\forall(i \in I, j \in J) \sum_{\boldsymbol{p} \in (i,j))} r(\boldsymbol{p}) / \#(\boldsymbol{p} \in (i, j)). \tag{3.56}$$

Eine auf diese Weise erzeugte Reliefkarte kann in den Texturspeicher der Grafikkarte geladen und dort als Bumpmap benutzt werden. Hierzu wird die Höheninformation der Reliefkarte automatisch in Oberflächennormalen umgewandelt. Bei der Berechnung der Beleuchtung wird dann statt der Normalen des Ebenensegmentes die Oberflächennormale an jedem Pixel benuzt um den Farbwert dieses Pixels zu berechnen. Dadurch entsteht beim Betrachter der Eindruck, dass diese Fläche zusätzliche Details enthält, obwohl nur eine planare Fläche zu Grunde liegt.

Die Approximation der ursprünglichen Punktwolke kann nun dadurch erreicht werden, dass alle Reliefkartenwerte, die durch eine markierte Zelle in der Belegtheitskarte gekennzeichnet sind, als Höhenwert benutzt werden. Dieser Höhenwert muss dann mit dem Ebenenschwellwert ϵ_{plane} verrechnet werden. Anschließend wird der 3D Punkt mit der Rückprojektion der Ebene aus dem Koordinatensystem der Ebene ins globale Koordinatensystem projiziert.

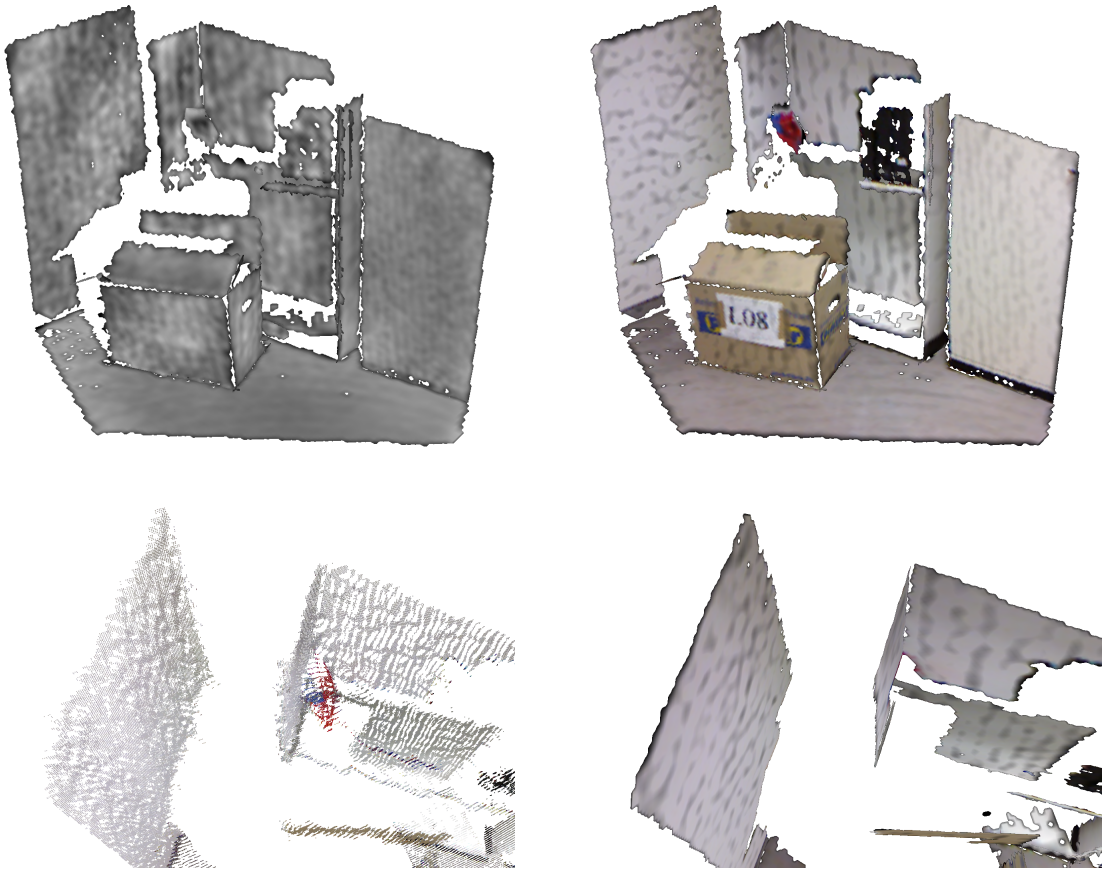


Abbildung 3.5: Reliefkarten der planaren Segmente der Szene aus Abbildung 3.4 als Graustufen-Textur (oben links) und ihre Visualisierung als Bumpmap mit eingeblendeter Farbtextur (oben rechts). Die unteren beiden Bilder zeigen eine Detailansicht der oberen linken Ecke der Szene als Punktmenge (unten links) und als texturierte Fläche mit Reliefkarte (unten rechts).

4 Experimente und Auswertung

Dieses Kapitel enthält Beschreibungen von durchgeführten Experimenten und die Auswertung des Segmentierungsverfahrens. Zuerst wird der Aufbau der Experimente geschildert, was die Spezifikationen der genutzten Sensoren und der Testdatensätze beinhaltet. Anschließend werden Spezialfälle von Datensätzen gezeigt und begründet, warum das hier vorgestellte Verfahren auf diesem Datensatz besonders gut funktioniert und wo die Schwierigkeiten liegen. Danach werden die erzielten Ergebnisse der einzelnen Datensätze im Detail ausgewertet. Am Ende des Kapitels wird die Zeiteffizienz der Implementierung exemplarisch untersucht.

Die Bewertung der Ergebnisse von Segmentierungsalgorithmen wurden bei Hoover [6] untersucht. Hierzu wurden die SegComp Datensätze zusammengestellt. Diese enthalten sowohl Tiefenbilder, die einem Segmentierungsalgorithmus als Eingabe dienen, als auch händisch vorgenommene Segmentierungen dieser Tiefenbilder, welche die Bewertungsgrundlage bilden. Für den Vergleich der Segmentierungen wurde von Hoover et al.[6] ein Auswertungsprogramm entwickelt, das in Bilddateien kodierte Segmentierungen miteinander vergleicht. Diese Bilddatein kodieren in Graustufen Fehlmessungen, Hintergrund und Abschattung als nicht segmentierbare Bildpunkte und bis zu 245 unterschiedliche Segmente.

Der ABW Testdatensatz von SegComp besteht aus 30 Tiefenbildern, die mit einer Kamera der Firma ABW aufgenommen wurden. Diese Kamera arbeitet ähnlich wie die Kinect der Firma Microsoft mit einem Projektor, der strukturiertes Licht in eine Szene abstrahlt. Das entstandene Muster wird mit der Kamera aufgenommen und in Tiefenwerte umgerechnet. Durch dieses Verfahren und eine relativ große Basis, dem Abstand zwischen Projektor und Kamera, gibt es im resultierenden Tiefenbild Abschattung von zwei Quellen. Der Datensatz besteht aus Bildern, die ausschließlich planare Segmente beinhalten.

Die 30 Bilder des SegComp Perceptron Datensatzes wurden im CESAR lab des Oak Ridge National Laboratory (Tennessee) mit einer Perceptron Laser Radar Kamera aufgenommen. Die Tiefenbilder haben 512×512 Bildpunkte und weisen insgesamt höheres Rauschen in den Tiefenwerten auf. Außerdem befinden sich Fehlmessungen an Sprungkanten zwischen Objekten und der Rückwand. Auch auf den Bildern dieses Datensatzes sind nur Ebenensegmente abgebildet.

Zusätzlich zu den beiden SegComp ABW und Perceptron Datensätzen wurden Szenen, die mit einer Microsoft Kinect aufgenommen wurden, händisch segmentiert. Diese Szenen enthalten nicht nur die in den verwendeten SegComp Datensätzen vorkommenden Flächen, sondern auch zylindrische Segmente. Daher wurde ein Kinect Ebenen Datensatz und ein Kinect Zylinder Datensatz erzeugt. Diese beiden aus je 30 Szenen bestehenden Datensätze beinhalten Punktwolken, Farbbilder und händische Segmentierungen. Im Kinect Ebenen Datensatz wurden alle Punkte, die auf einem Zylinder liegen als Fehlmessung markiert, so dass diese beim Vergleich nicht als Ebenensegmente gezählt werden. Die Punkte im Zylinderdatensatz, die auf Ebenen liegen wurden aus dem gleichen Grund als Fehlmessung markiert.

Zur Bewertung der vom entwickelten Algorithmus ausgegebenen Segmentierungen wurde das von Hoover et al. [6] entwickelte Auswertungsprogramm benutzt. Dieses bestimmt zu einem gegebenen Toleranzwert $0,51 \leq T \leq 1$ eine Klassifikation für die Segmente der beiden Segmentierungen.

Anhand der Unterschiede zwischen der Segmentierung des zu untersuchenden Algorithmus, folgend maschinelle Segmentierung genannt, und der händischen Segmentierung wird die Güte des Verfahrens ermittelt. Bei der Bewertung der maschinellen Segmentierung wird zuerst die Überlappung aller Segmente der händischen Segmentierung mit den Segmenten der maschinellen Segmentierung auf Basis der Punkte festgestellt. Hierzu wird zu jedem Einzelpunkt der händischen Segmentierung das entsprechende Segment der maschinellen Segmentierung ermittelt und umgekehrt. Nachdem alle Einzelpunkte abgearbeitet wurden, ist eine Tabelle entstanden, die zu jedem Segment der händischen Segmentierung alle Teilüberschneidungen in der Anzahl der Bildpunkten mit jedem Segment der maschinellen Segmentierung enthält. Anhand dieser Tabelle läßt sich schnell für jede Paarung aus Segmenten der beiden Segmentierungen die Überlappung berechnen.

Zwei Segmente A und B überlappen sich gegenseitig, wenn Bildpunkte, die in der händischen Segmentierung in Segment A liegen und in der maschinellen Segmentierung in Segment B. Die Anzahl der Bildpunkte, die in beiden Segmenten enhalten

sind wird im weiteren Text als $overlap(A, B)$ bezeichnet. Wenn sich zwei Segmente überlappen, so werden zwei Überlappungskoeffizienten k_1 und k_2 berechnet, um die Klassifikation vorzunehmen. Koeffizient k_1 wird aus dem Verhältnis der Anzahl Punkte der Überlappungsmenge $|overlap(A, B)|$ und der Grundmenge des ersten Segments $|A|$ bestimmt, k_2 entspricht dem Verhältnis der Anzahl der Punkte des zweiten Segments $|B|$ und der Grundmenge:

$$k_1 = \frac{overlap(A, B)}{|A|}, \tag{4.1}$$

$$k_2 = \frac{overlap(A, B)}{|B|}. \tag{4.2}$$

Sind beide Koeffizienten größer oder gleich dem gegebenen Schwellwert T, so bilden diese Segmente (A, B) eine genügend große Punktüberlappung bezüglich des Schwellwertes T.

Die Klassifikationen ergeben sich aus den Punktüberlappungen:

Korrekt Wenn ein Segment der maschinellen Segmentierung und ein Segment der händischen Segmentierung eine genügend große Punktüberlappung bezüglich Schwellwert T bilden, wird das Segment der maschinellen Segmentierung als korrekt klassifiziert.

Übersegmentiert Mehrere Segmente der maschinellen Segmentierung und ein Segment der händischen Segmentierung bilden eine genügend große Punktüberlappung bezüglich Schwellwert T. Das Segment der händischen Segmentierung gilt als übersegmentiert.

Untersegmentiert Ein Segment der maschinellen Segmentierung und mehrere Segmente der händischen Segmentierung bilden eine genügend große Punktüberlappung bezüglich Schwellwert T. Das Segment der maschinellen Segmentierung wird als untersegmentiert klassifiziert.

Nicht gefunden Ein Segment der händischen Segmentierung wird als nicht gefunden klassifiziert, wenn es mit keinem Segment der maschinellen Segmentierung eine genügend große Punktüberlappung bildet.

Rauschen Ein Segment der maschinellen Segmentierung, das keine genügend große Punktüberlappung mit einem Segment der händischen Segmentierung hat, wird als Rauschen klassifiziert.

4.1 Ebenensegmentierung

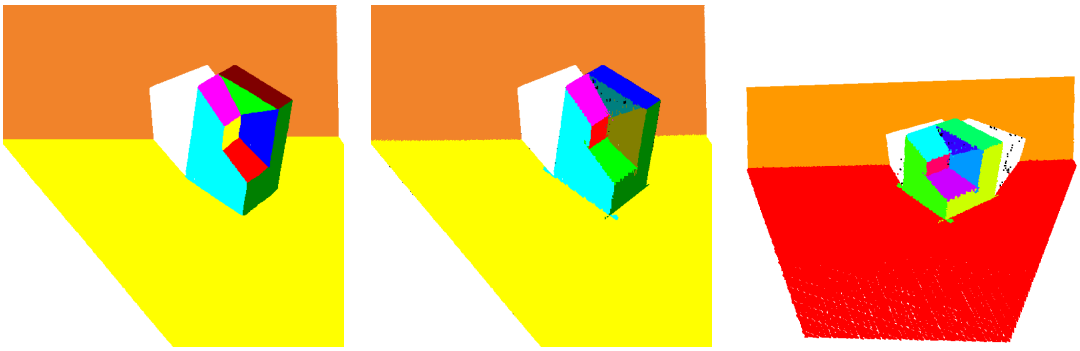
In disem Abschnitt wird eine Auswertung der Ebenensegmentierung vorgenommen. Hierzu werden als erstes die Ergebnisse auf dem ABW Datensatz und im Anschluss auf dem Perceptron Datensatz gezeigt und mit anderen Verfahren verglichen. Danach werden Segmentierungen auf dem Kinect Datensatz und anschließend einiger unsortierter 3D Punktwolken diskutiert.

Zusätzlich wurden fünf Abwandlungen des hier beschriebenen Verfahrens implementiert um zu zeigen, dass die Kombination aus RANSAC und Hough-Transformation in der hier gezeigten Weise besser und effizienter ist als jedes dieser Verfahren ohne das andere.

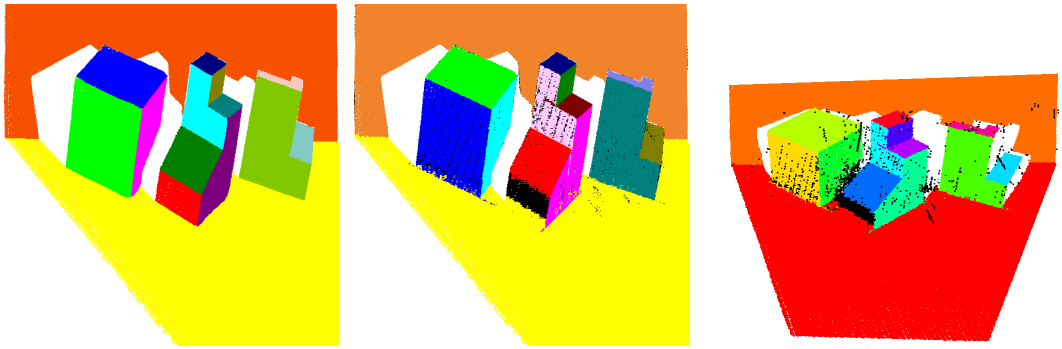
4.1.1 ABW Datensatz

Sowohl der ABW als auch der Perceptron Datensatz sind zusätzlich mit Informationen über die Winkel zwischen zwei angrenzenden Flächen versehen. Wurden in der maschinellen Segmentierung zwei angrenzende Ebenensegmente korrekt erkannt, so wird der Winkel zwischen ihren Normalen bestimmt. Ein Vergleich dieser berechneten Winkel der maschinellen Segmentierung und der gemessenen Winkel im Datensatz ermöglich eine Aussage über die Güte der geschätzten Ebenen. Der Kinect Ebenen Datensatz hat keine solchen Winkelinformationen, so dass die Güte der Ebenenparameter nur auf dem ABW Datensatz und dem Perceptron Datensatz verglichen werden kann.

Abbildung 4.1 zeigt zwei der 30 SegComp Datensätze mit händischer Segmentierung und der Ausgabe des hier präsentierten Algorithmus. Mit einem Schwellwert von 80% Punktüberlappung wurden alle Ebenensegmente in den oberen Bildern korrekt segmentiert. In der Szene auf den unteren Bildern wurde ein kleines Segment nicht gefunden (rot in der händischen Segmentierung (Abb. 4.1d)), was an der relativ kleinen Größe und einer ungünstigen Lage in der Oktalbaum Diskretisierung liegt. Einige Punkte im gelben Segment der untere 3D Ansicht (Abb. 4.1f) wurden nicht korrekt zugeordnet, da ihre Distanz zur geschätzten Ebene zu groß waren. Das gleiche Problem tritt auch beim türkisen Segment in der Mitte der Szene auf. Zusätzlich wurden einige Punkte der türkisen Ebene dem lilanen angrenzenden Segment falsch zugeordnet. Dies geschieht, da bei der Nachverarbeitung die Verteilung der Punkte anhand der Mittelebene 3.4.2 keine bessere Zuordnung zuläßt. Die restlichen



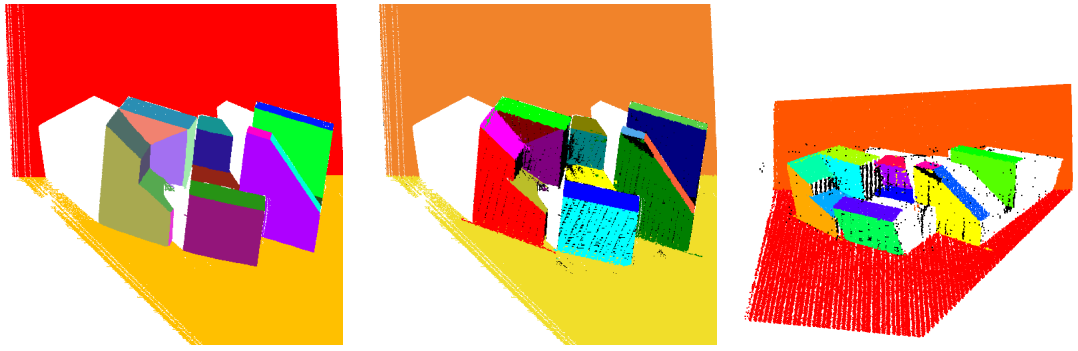
(a) händische Segmentierung **(b)** maschinelle Segmentierung **(c)** 3D Ansicht der maschinellen Segmentierung



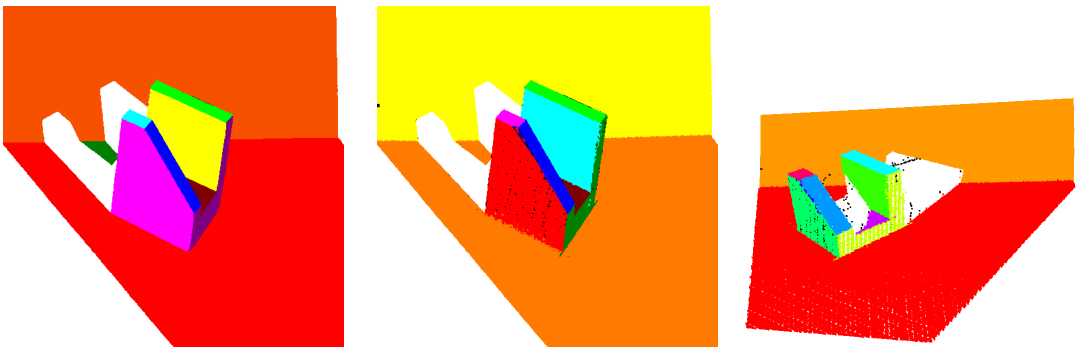
(d) händische Segmentierung **(e)** maschinelle Segmentierung **(f)** 3D Ansicht der maschinellen Segmentierung

Abbildung 4.1: Zwei Szenen des SegComp ABW Datensatzes. Die händischen Segmentierungen wurden links, die maschinellen Segmentierungen in der Mitte und die 3D Ansichten wurde rechts abgebildet. Die Farben kodieren die verschiedenen Segmente. Fehlmessungen und Hintergrund sowie Bildpunkte ohne Segmentzuordnung wurden in den linken und mittleren Bildern weiß markiert. In den 3D Ansichten wurden unsegmentierte Punkte mit schwarz gekennzeichnet.

Ebenen wurden korrekt detektiert, was bei insgesamt 15 Segmenten der händischen Segmentierung eine Detektionsrate von 80% ergibt.



(a) händische Segmentierung **(b)** maschinelle Segmentierung **(c)** 3D Ansicht der maschinellen Segmentierung



(d) händische Segmentierung **(e)** maschinelle Segmentierung **(f)** 3D Ansicht der maschinellen Segmentierung

Abbildung 4.2: Segmentierung zweier Szenen des ABW Datensatzes. Auf beiden Szenen wurden etwa 77% (oben) bzw. 70% (unten) der Flächen mit 80% Punktüberlappung korrekt segmentiert. Die Farbwerte der 3D Punkte kodieren die Segmentierung. In den linken und mittleren Bilder sind Hintergrund und Fehlmessungen weiß, in den rechten Bildern wurden nicht zugeordnete Punkte schwarz markiert.

Auf Abbildung 4.2 sind einige Fehler in der Segmentierung zu erkennen. Auf beiden Szenen werden einige Segmente nicht gefunden. In der Szene auf den oberen Bildern wurden einige Segmente deswegen nicht erkannt, weil die Punkte sich im dreidimensionalen Raum aufgrund von Diskretisierungsfehlern bei der Aufnahme und einem Winkel von nahezu 90° nicht gleichmäßig auf der Fläche verteilen (vgl. schwarze Punkte in Abb. 4.2c). In der unteren Szene wurden das dunkelgrüne und

das rote Segment des Bodens vom hier präsentierten Algorithmus als ein Segment detektiert. Durch die Prüfung des Zusammenhangs auf einer groberen Auflösung konnte die Lücke zwischen den beiden Teilen der Bodenebene überbrückt werden, was bei der Detektion der Bodenebene Sinn macht. Zwischen der hellgrünen Fläche und dem graublauen Segment des rechten unteren Bildes ist noch ein kleines dunkelblaues Segment zu erkennen. Dieses Segment entstand durch die Lage im Oktalbaum und die daraus resultierende Normalenschätzung. An dieser Stelle befinden sich ähnlich viele Punkte von der graublauen und von der hellgrünen Fläche in mehreren benachbarten Oktalbaum Knoten und werden zu je einem Surfel mit ähnlicher Orientierung zusammengefasst. Wie auf dem Bild deutlich zu sehen ist, umfasst die dadurch entstandene Ebene aber nur wenige Bildpunkte.

Tabelle 4.1 zeigt die Ergebnisse der Auswertung des SegComp ABW Datensatzes für eine Überlappungstoleranz von 80%. Das hier in der Arbeit beschriebene Verfahren segmentierte durchschnittlich 11,1 von 15,2 Flächen pro Bild korrekt, was einer Erkennungsrate von 73% entspricht. Die Verfahren, zu denen ein Winkelfehler veröffentlicht wurde, haben alle einen sehr geringen Winkelfehler von $1,3°$ bis $1,6°$. Das hier vorgestellte Verfahren liegt mit einem Winkelfehler von $1,4°$ in diesem Bereich. Die obere Hälfte der Tabelle zeigt die Ergebnisse anderer Verfahren, die mit dem SegComp Datensatz verglichen wurden. Hier erzielen vier der Verfahren (USF, UB, UE und UFPR) bessere Ergebnisse gemessen an der Anzahl der durchschnittlich korrekt segmentierten Flächen. Diese Verfahren arbeiten ausschließlich auf den bereitgestellten Tiefenbildern und nutzen die Pixelnachbarschaft im Tiefenbild aus. Dadurch können sie bessere Ergebnisse erzielen. Das hier vorgestellte Verfahren arbeitet statt dessen auf einer daraus berechneten unsortierten Punktwolke und ohne Nutzung von Pixelnachbarschaften und ist damit allgemeiner anwendbar. Die anderen Verfahren könnten eine unsortierte 3D Punktwolke nicht segmentieren. Im Vergleich der Ergebnisse des hier präsentierten Verfahrens mit den Ergebnissen der Universität Süd Florida (USF), so wird neben der etwa 10% größeren Erkennungsrate deutlich, dass das Verfahren wesentlich weniger untersegmentiert. Ein Beispiel für eine solche Untersegmentierung ist in Abbildung 4.2 zu sehen, wo die Tischebene (in der händischen Segmentierung rot und grün) durch den hier vorgestellten Algorithmus als ein Segment detektiert wird. Die Aufteilung dieser Ebene in der händischen Segmentierung mag auf einem Tiefenbild sinnvoll erscheinen, macht in der 3D Punktwolke allerdings wenig Sinn, da es sich um eine einzige Fläche handelt.

Die beiden Verfahren, die als *ransacOnly* in Tabelle 4.1 eingetragen wurde, ist eine einfache Implementierung des RANSAC-Verfahrens, bei der nach und nach Ebenen mit RANSAC in der vorliegenden Punktwolke gesucht und aus der Punktmenge

Gruppe	korrekt	über-segmentiert	unter-segmentiert	nicht gefunden	Rauschen	Winkel-fehler
USF	12,7 (83,5%)	0,2	0,1	2,1	1,2	1,6°
WSU	9,7 (63,8%)	0,5	0,2	4,5	2,2	1,6°
UB	12,8 (84,2%)	0,5	0,1	1,7	2,1	1,3°
UE	13,4 (88,1%)	0,4	0,2	1,1	0,8	1,6°
OU	9,8 (64,4%)	0,2	0,4	4,4	3,2	-
PPU	6,8 (44,7%)	0,1	2,1	3,4	2,0	-
UA	4,9 (32,2%)	0,3	2,2	3,6	3,2	-
UFPR	13,0 (85,5%)	0,5	0,1	1,6	1,4	1,5°
ransacOnly	3,1 (20,4%)	3,2	1,2	6,2	6,6	1,7°
noRansac	4,4 (28,9%)	0,2	0,4	9,7	3,0	3,1°
octree8	5,6 (36,8%)	0,0	0,5	8,6	1,1	1,4°
octree9	10,0 (65,8%)	0,2	0,3	4,4	1,2	1,5°
octree10	7,2 (47,4%)	1,1	0,1	6,8	7,0	1,2°
AIS/FKIE	11,1 (73,0%)	0,2	0,7	2,2	0,8	1,4°

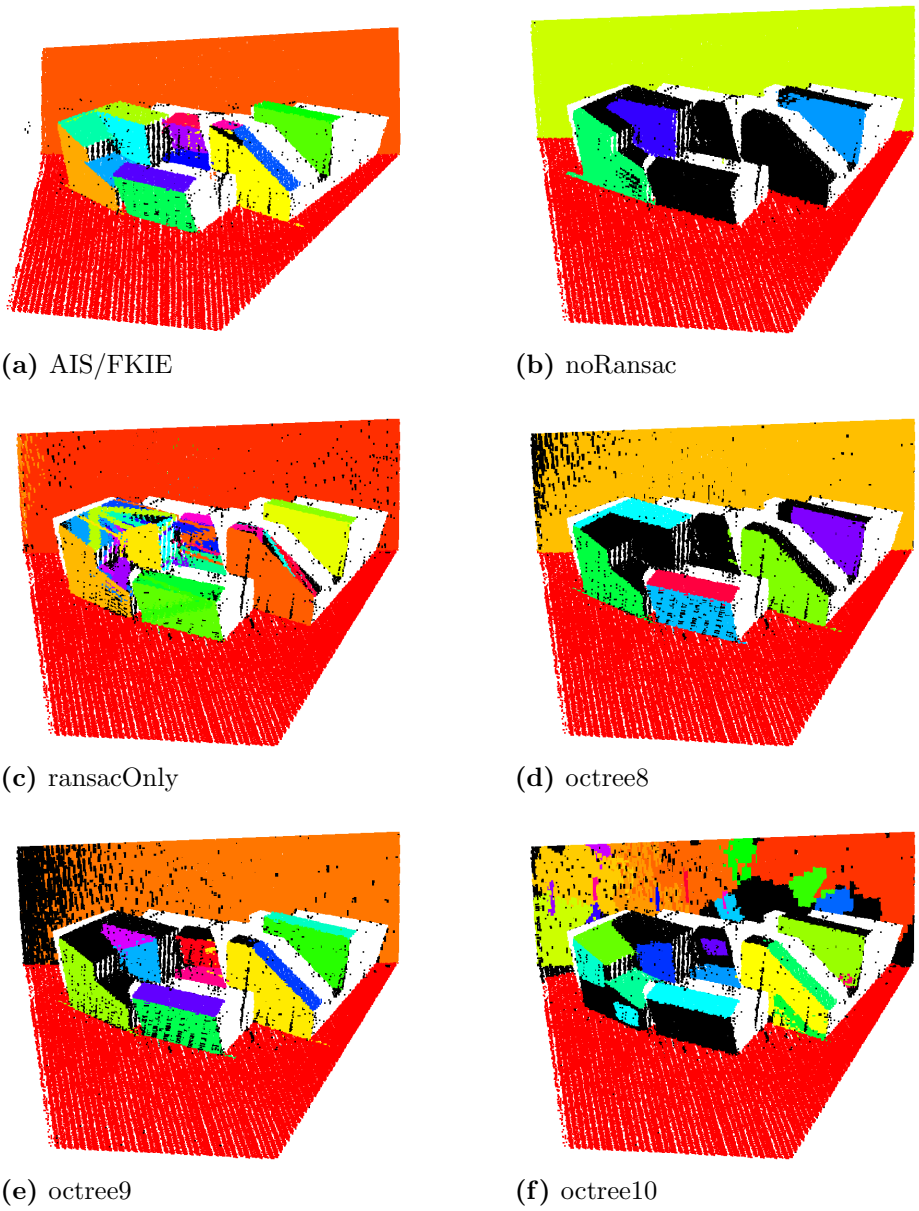
Tabelle 4.1: Vergleich mit anderen Segmentierungsverfahren (aus [4]) auf dem SegComp ABW Datensatz mit 80% Überlappungstoleranz. Im Durchschnitt über alle 30 Bilder sind auf jedem $15,2$ Regionen abgebildet. Die erste Spalte gibt anhand eines Kürzels die jeweilige Universität der Arbeitsgruppe an, in der das Segmentierungsverfahren entwickelt wurde. *noRansac*, *ransacOnly* und die drei mit *octree[i]* benannten Verfahren wurden zum Vergleich implementiert oder sind Abwandlungen des in dieser Arbeit beschriebenen Verfahrens. Der Winkelfehler gibt den durchschnittlichen Fehler der Winkel zwischen je zwei benachbarten korrekt klassifizierten Flächen an. Bei drei der Verfahren ist der Winkelfehler nicht bekannt und wurde daher nicht angegeben. Die anderen Werte sind durchschnittliche Anzahlen von Flächen, die übersegmentiert, untersegmentiert, als nicht gefunden oder als Rauschen klassifiziert worden sind.

enfernt werden. *NoRansac* ist eine Variation des hier vorgestellten Verfahrens, bei welcher der RANSAC Schritt nach der Suche der Zusammenhangskomponenten entfernt wurde. Statt dessen werden aus den Punkten der zugrundeliegenden Punktmenge aller zusammenhängenden Surfeln diejenigen bestimmt, die in einer geringen Distanz zur Ebene liegen, deren Parameter in der Hough-Transformation bestimmt wurden. Die drei Varianten, die mit *octree[i]* bezeichnet wurden, sind Varianten des in dieser Arbeit beschriebenen Verfahrens, bei denen nur eine Oktalbaumtiefe betrachtet wurde. Auf den Surfeln der entsprechenden Auflösung wurden dann nacheinander Hough-Transformation, dann Zusammenhangskomponentensuche und anschließend RANSAC angewendet, um Ebenensegmente zu finden. Wie der Tabelle zu entnehmen ist, erzielen sowohl das *noRansac*-Verfahren, als auch *ransacOnly* keine annähernd guten Ergebnisse, wie das hier vorgestellte Verfahren.

Abbildung 4.3 zeigt die ABW Szene aus Abblidung 4.2 und ihre Segmentierung durch alle fünf Vergleichsverfahren. Durch die nötige Verkleinerung des Distanzschwellwertes ϵ_{ht} bei der Hough-Transformation (siehe Kapitel 3 Abschnitt 3.2) bei *noRansac* (Abb. 4.3b) wurden viel weniger Ebenen gefunden als bei der Segmentierung mit RANSAC-Schritt. *RansacOnly* (Abb. 4.3c) findet die Hauptebenen und ein paar der größeren Objektflächen Segmente, viele kleine Ebenen werden allerdings nicht gefunden sondern nach kleinster Distanz den bereits gefundenen Segmenten zugeordnet. *octree8* (Abb. 4.3d) findet die Hauptebenen und einige der großen Objektebenen (ähnlich wie *ransacOnly*), aber durch die Beschränkung auf nur diese eine Auflösungsstufe (Volumen des Oktalbaumknotens mit 16cm Kantenlänge) konnten die kleinen Ebenen nicht erkannt werden. Auf den kleineren Auflösungsstufen (8cm Kantenlänge in Abb. 4.3e bzw. 4cm Kantenlänge in Abb. 4.3f) wurden die kleineren Objektflächen besser erkannt, allerdings konnten die großen Ebenen aufgrund von immer ungenaueren Normaleninformation auf feinerer Auflösung nicht mehr gefunden werden.

Die Beschränkung auf eine einzige Oktalbaum Tiefe (*octree[i]*) scheint das Ergebnis wesentlich weniger zu beeinflussen, wenn man die richtige Tiefe wählt. Die Wahl der richtigen Tiefe hängt allerdings ausschließlich von der Szene ab, die segmentiert werden soll.

Die Ergebnisse für unterschiedliche Überlappungstoleranzen beim Vergleich der hier vorgestellten Segmentierung mit den Daten der händischen Segmentierung wurden in Tabelle 4.2 zusammengefasst. Die Tabelle zeigt, dass sich die Ergebnisse auch bei Überlappungsschwellwerten von 51% bis hin zu 80% kaum verändern. Das bedeutet, dass fast alle Segmente, die korrekt gefunden werden auch eine Punktüber-



(a) AIS/FKIE **(b)** noRansac

(c) ransacOnly **(d)** octree8

(e) octree9 **(f)** octree10

Abbildung 4.3: Sechs verschiedene Segmentierungen der gleichen ABW Szene. Die Farbe eines Punktes zeigt die Segmentierung an. Schwarze Punte wurden nicht segmentiert. Oben von links nach rechts zeigen die Bilder folgende Segmentierungsverfahren: Verfahren aus Kapitel 3, *noRansac*, *ransacOnly*. In der unteren Reihe wurden von links nach rechts die Segmentierungen der Verfahren *octree8*, *octree9* und *octree10* abgebildet.

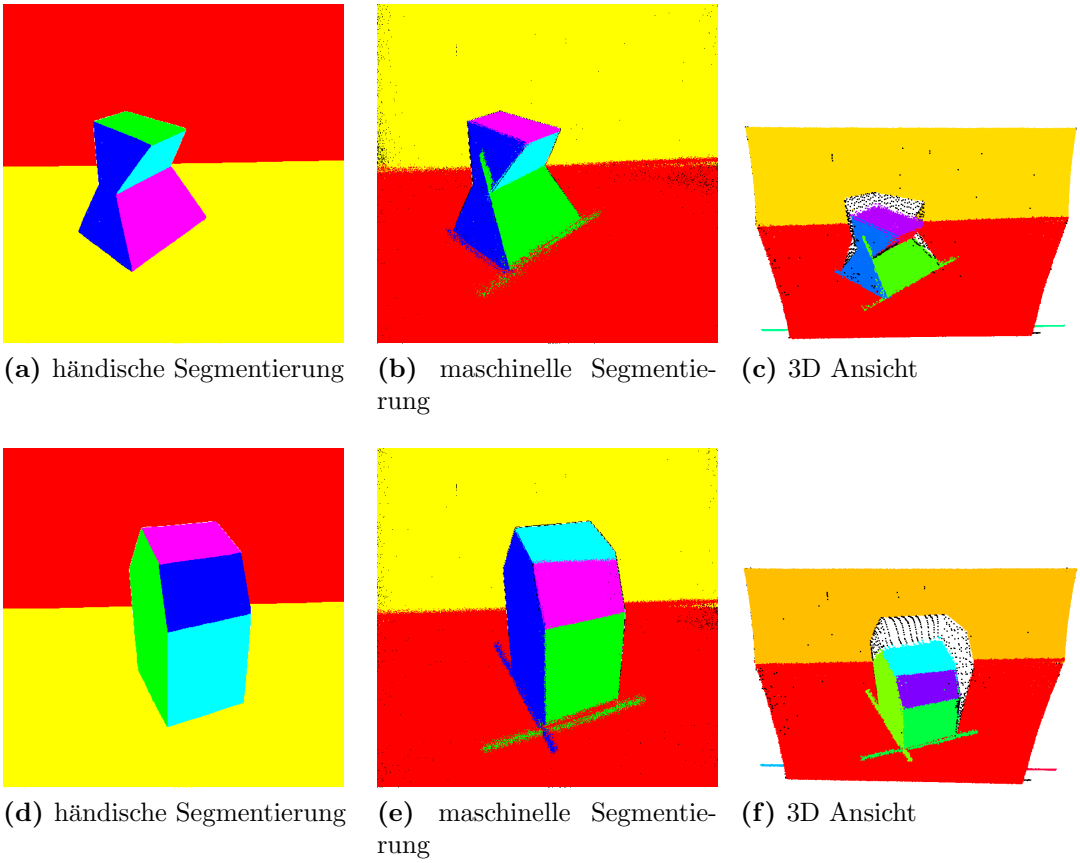
Überlappungstoleranz	51%	60%	70%	80%	90%
korrekt	11,4	11,5	11,4	11,1	8,8
übersegmentiert	0,2	0,2	0,2	0,2	0,1
untersegmentiert	0,9	0,8	0,8	0,7	0,7
nicht gefunden	1,4	1,5	1,7	2,2	4,8
Rauschen	0,2	0,2	0,3	0,8	3,4
Winkelfehler	1,4°	1,4°	1,4°	1,4°	1,3°
Standardabweichung	0,9°	0,9°	0,8°	0,8°	0,6°

Tabelle 4.2: Segmentierungsqualität für unterschiedliche Überlappungstoleranzen auf dem SegComp ABW Datensatz mit 30 Bildern und durschnittlich $15,2$ Ebenensegmenten in der händischen Segmentierung.

lappung von über 80% besitzen. Wie bei den anderen Verfahren (siehe Gotardo [4]), werden die Ergebnisse für einen Überlappungsschwellwert von 90% wesentlich schlechter. Die Flächen der maschinellen Segmentierung, für welche die Punktüberlappung nicht ausreichend ist, werden dann als Rauschen klassifiziert. Flächen der händischen Segmentierung ohne entsprechende Punktüberlappung gelten als nicht gefunden. Der durchschnittliche Winkelfehler zwischen zwei benachbarten Flächen ist für alle Überlappungstoleranzen fast gleich gut, beziehungsweise verbessert sich nur für eine Überlappung von 90% um 0,1°.

4.1.2 Perceptron Datensatz

Wie bereits erwähnt, ist beim SegComp Perceptron Datensatz das Rauschen der Punktwolke größer als beim ABW Datensatz, daher wurde der Distanzschwellwert für Ebenen etwas größer gewählt.



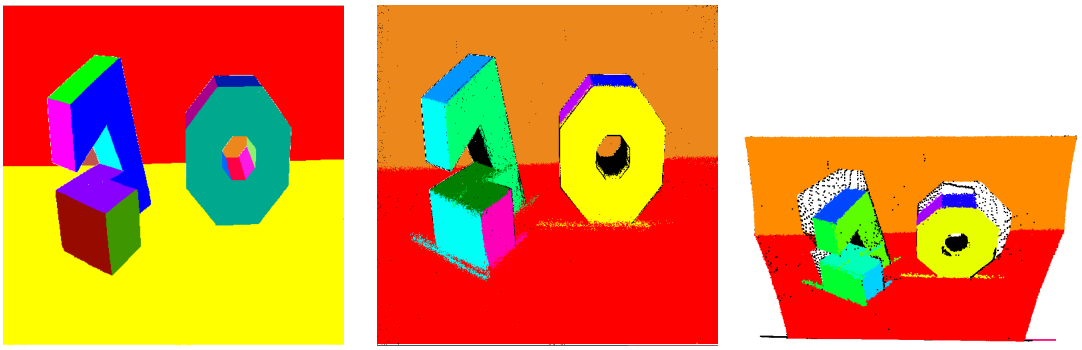
(a) händische Segmentierung **(b)** maschinelle Segmentierung **(c)** 3D Ansicht

(d) händische Segmentierung **(e)** maschinelle Segmentierung **(f)** 3D Ansicht

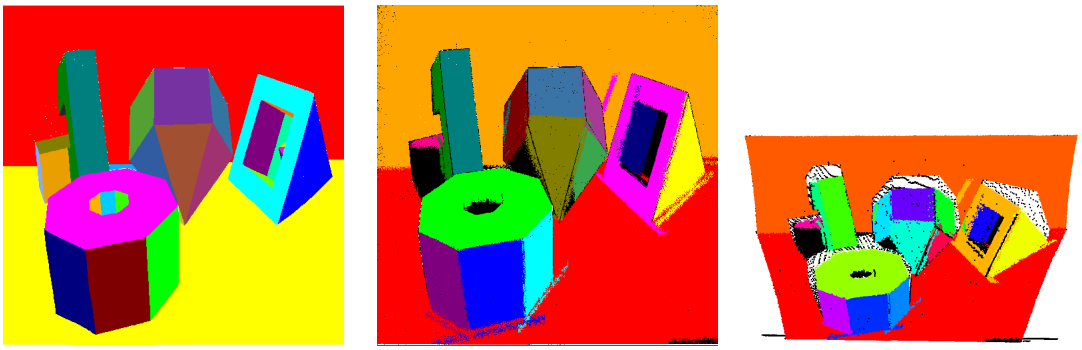
Abbildung 4.4: Zwei Szenen aus dem Perceptron Datensatz (oben und unten). Händische Segmentierung links, maschinelle Segmentierung in der Mitte und segmentierte 3D Ansicht der Szene rechts. Die Segmentzugehörigkeit wird durch die Farbe gegeben. Schwarze Bildpunkte sind Fehlmessungen, die keinem Segment zugeordnet wurden. In den linken und mittleren Bildern wurden auch diese Punkte weiß eingefärbt.

Abbildung 4.4 zeigt beispielhafte Segmentierungen von zwei Perceptron Szenen. Mit 80% Überlappungstoleranz werden alle Segmente in diesen gezeigten Bildern korrekt detektiert. Allerdings ist auf diesen Bildern zu sehen, dass ein paar Punkte außerhalb

der eigentlichen Begrenzung einer Fläche falsch zu dieser zugeordnet werden. Die Anzahl der falsch zugeordneten Punkte ist allerdings gering im Gegensatz zur Anzahl der korrekt zugeordneten Punkte. Weiterhin ist auf den Bildern zu erkennen, dass zwischen Objekt und Hintergrund Sprungkanten in der Punktwolke des Perceptron Datensatzes sind.



(a) händische Segmentierung **(b)** maschinelle Segmentierung **(c)** 3D Ansicht



(d) händische Segmentierung **(e)** maschinelle Segmentierung **(f)** 3D Ansicht

Abbildung 4.5: Zwei segmentierte Szenen des Perceptron Datensatzes mit geringerer Erkennungsrate. Die Farbe kodiert die Zugehörigkeit zu einem Segment. Schwarze Punkte in den linken Bildern markieren unsegmentierte Punkte. In den mittleren Bildern wurden diese Punkte weiß eingefärbt. Auf den linken Bildern sind nur Fehlmessungen als weiße Bildpunkte dargestellt.

Abbildung 4.5 zeigt zwei Szenen mit einer geringeren korrekten Erkennungsrate (bei 80% Punktüberlappung). In der oberen Szene wurde auf dem linken Objekt eine kleine Ebene nicht detektiert, auf dem rechten Objekt wurden die Flächensegmente im

Inneren aufgrund von zu geriger Größe nicht gefunden. Außerdem wurde die Rückwand (händische Segmentierung (Abb. 4.5b) orange und rot, maschinelle Segmentierung (Abb. 4.5c) orange) als ein Ebenensegment detektiert, also untersegmentiert. Auf der unteren Szene sind einige Flächen mit sehr wenigen Punkten in der händischen Segmentierung (Abb. 4.5d) eingetragen. Diese Segmente können aufgrund des insgesamt recht großen Punktrauschens im Schritt der Hough Transformation nicht gefunden werden.

Gruppe	korrekt	über-segmentiert	unter-segmentiert	nicht gefunden	Rauschen	Winkel-fehler
USF	8,9 (60,9%)	0,4	0,0	5,3	3,6	2,7°
WSU	5,9 (40,4%)	0,5	0,6	6,7	4,8	3,3°
UB	9,6 (65,7%)	0,6	0,1	4,2	2,8	3,1°
UE	10,0 (68,4%)	0,2	0,3	3,6	2,1	2,6°
UFPR	11,0 (75,3%)	0,3	0,1	3,0	2,5	2,5°
noRansac	2,1 (14,4%)	0,0	0,6	10,9	4,2	0,8°
octree8	2,9 (19,9%)	0,1	0,2	11,2	2,3	9,6°
octree9	6,7 (45,9%)	0,6	0,2	6,9	3,0	7,4°
octree10	1,2 (8,2%)	0,1	0,0	13,3	10,2	0,1°
ransacOnly	2,6 (17,8%)	0,3	0,1	11,4	16,1	2,1°
AIS/FKIE	7,4 (50,1%)	0,3	0,4	6,2	3,9	5,2°

Tabelle 4.3: Vergleich mit anderen Segmentierungsverfahren (aus [4]) auf dem SegComp Perceptron Datensatz mit 80% Überlappungstoleranz. Im Durchschnitt über alle 30 Bilder sind auf jedem $14,6$ Segmente abgebildet. Die meisten planaren Segmente ($75,3\%$) werden vom Algorithmus von Gotardo [4] (UFPR) korrekt detektiert.

Tabelle 4.3 zeigt einen Vergleich der Segmentierungsergebnisse unterschiedlicher Arbeitsgruppen bei einer Punktüberlappung von mindestens 80% auf dem SegComp Perceptron Datensatz. Die Ergebnisse zeigen, dass der hier vorgestellte Algorithmus auf dem Perceptron Datensatz nicht so gut arbeitet, wie die meisten anderen Verfahren, denn es werden nur $50,1\%$ der durchschnittlich $14,6$ Flächen je Bild korrekt erkannt. Das Verfahren mit den meisten korrekt erkannten Ebenensegmenten von der UFPR erkennt $75,3\%$. Im Gegensatz zu den Bildern des ABW Datensatzes ist der Rauschanteil bei den Perceptron Bildern wesentlich höher, außerdem gibt es viel mehr Flächen mit nur sehr wenigen Punkten auf den Perceptron Bildern. Die Verfahren der anderen Arbeitsgruppen, die alle auf Tiefenbildern arbeiten, sind deutlich dadurch im Vorteil, dass bei der Erkennung kleiner Flächen die Zusammenhangsin-

formation durch die Pixelnachbarschaft im Tiefenbild genutzt werden kann.

Die beiden zum Vergleich implementierten Verfahren *ransacOnly* und *noRansac* zeigen auch auf diesem Datensatz keine guten Segmentierungen und finden im Durchschnitt keine 20% der Flächen mit 80% Punktüberlappung. Das Verfahren *octree9*, das sich von dem ingesamt hier vorgestellten Verfahren nur dadurch unterscheidet, dass es auf eine Oktalbaumauflösung von 14, 4 cm beschränkt wurde, zeigt deutlich die besten Ergebnisse der Vergleichsimplementierungen. Wie schon beim ABW Datensatz bleibt allerdings anzumerken, dass diese Auflösung vollständig von der zu segmentierenden Szene abhängig ist. Durch die geschickte Kombination der verschiedenen Auflösungsstufen in einem Verfahren, nutzt der Algorithmus aus dieser Arbeit die Ergebnisse aller relevanten Auflösungsstufen unabhängig von der Größe der Szene.

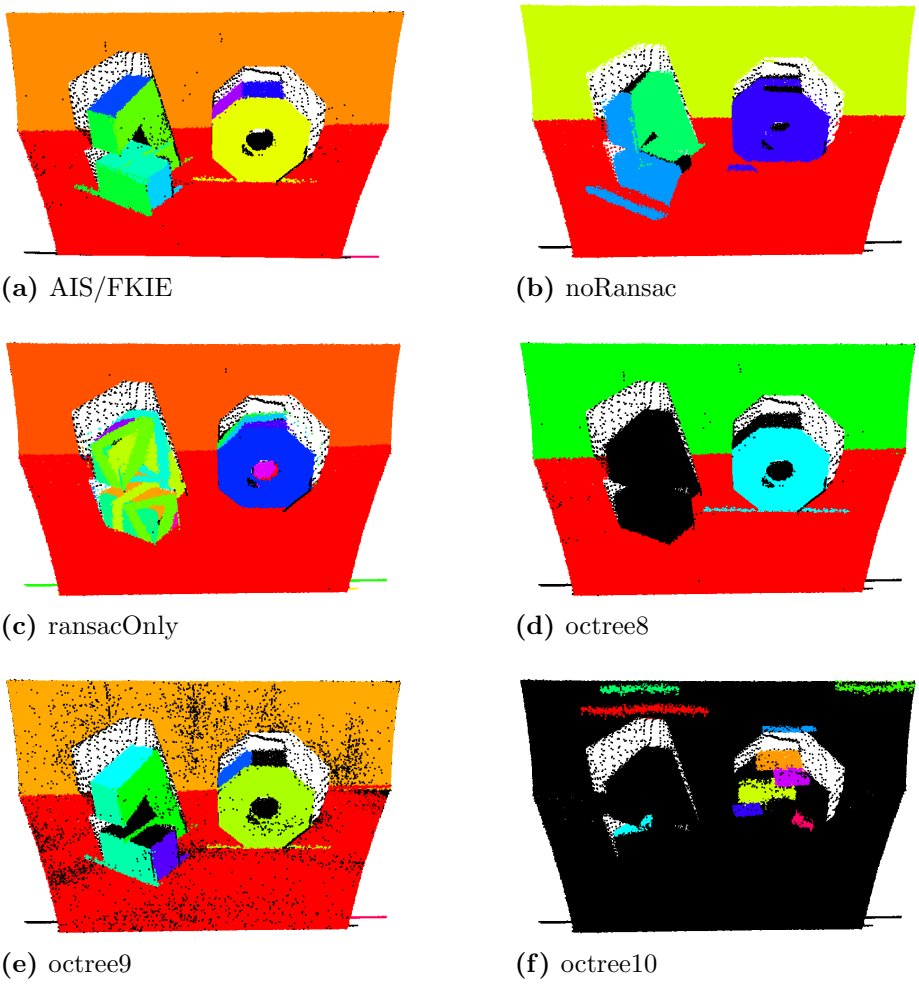
Abbildung 4.6 zeigt die Segmentierungen der Vergleichsverfahren eines Perceptron Datensatzes. Aufgrund von hohem Punktrauschen mussten die Parameter der Hough-Transformation beim Verfahren *noRansac* (Abb. 4.6b) so gewählt werden, dass auch Punkte mit größerem Punktabstand (als beim oben beschriebenen Verfahren mit RANSAC Parameterverbesserung) zur Ebene hinzugefügt werden. Dadurch werden die Hauptebenen richtig erkannt, die Flächen der Objekte werden allerdings untersegmentiert. Das Verfahren *ransacOnly* (Abb. 4.6c) detektiert die Hauptebenen richtig, kann aber dann die meisten Flächen der Objekte nicht mehr korrekt segmentieren. *octree8* (Abb. 4.6d) findet die Hauptebenen (Tisch und Rückwand) richtig, sowie die vordere Fläche des rechten Objekts, die Diskretisierung im Oktalbaum auf dieser Auflösungsstufe von 16 cm erlaubt keine Segmentierung der feineren Strukturen. *octree9* (Abb. 4.6e) arbeitet auf den Hauptebenen aufgrund des Punktrauschens nicht mehr so gut, erkennt dafür allerdings kleinere Ebenensegmente als *octree8*. *octree10* detektiert ausschließlich die orangene Fläche in Abbildung 4.6d. Aufgrund des großen Rauschens der Punktposition können die Normalen auf dieser Oktalbaumauflösung (4cm) nicht mehr richtig bestimmt werden, da die Punkte im Volumen eines Oktalbaumknotens nicht durch eine Fläche angenähert werden können. Die kleinen Segmente im Inneren des rechten Objektes werden in keinem der Vergleichsverfahren detektiert.

In der Tabelle 4.4 werden die durchschnittlichen Anzahlen der vorkommenden Klassifikation in einem Bild dargestellt. Es ist deutlich zu sehen, dass die drei linken Spalten recht ähnlich zueinander sind, was zeigt, dass die meisten korrekt klassifizierten Ebenensegmente eine Punktüberlappung von 70% bis 80% aufweisen. Nur etwa die Hälfte aller, mit einer Überlappungstoleranz von 60%, korrekt klassifizier-

Überlappungstoleranz	51%	60%	70%	80%	90%
korrekt	8,4	8,5	8,2	7,4	4,2
übersegmentiert	0,5	0,4	0,4	0,3	0,2
untersegmentiert	0,7	0,6	0,4	0,4	0,2
nicht gefunden	4,0	4,4	5,0	6,2	9,7
Rauschen	2,0	2,2	2,7	3,9	7,3
Winkelfehler	6,0°	5,9°	5,9°	5,2°	6,2°
Standardabweichung	7,8°	7,9°	7,5°	5,8°	1,0°

Tabelle 4.4: Segmentierungsqualität für unterschiedliche Überlappungstoleranzen auf dem SegComp Perceptron Datensatz.

ten Ebenen erreicht eine Punktüberlappung von 90%. Insgesamt werden für alle Überlappungstoleranzen nur wenige Segmente über- oder untersegmentiert.



(a) AIS/FKIE **(b)** noRansac

(c) ransacOnly **(d)** octree8

(e) octree9 **(f)** octree10

Abbildung 4.6: Sechs verschiedene Segmentierungen der Perceptron Szene von Abbildung 4.5a. Die Farbe eines Punktes zeigt die Segmentierung an. Schwarze Punkte wurden nicht segmentiert.

4.1.3 Kinect Ebenen Datensatz

Der Kinect Ebenen Datensatz besteht aus 30 verschiedenen Punktwolken und ihrer händischen Segmentierung. Die zylindrischen Objekte der Szenen wurden als Fehlmassungen markiert, so dass sie bei der Berechnung der Gesamtzahl an vorkommenden Segmenten der händischen Segmentierung nicht berücksichtigt werden. Da in den Bildern dadurch einige Punkte als Fehlmessungen markiert sind, ist der Rauschanteil der maschinellen Segmentierung auf diesem Datensatz besonders hoch.

Bei der Segmentierung der Bilder des Kinect Ebenen Datensatzes wurde das Rauschen des Sensors durch ein Modell approximiert. Dazu wird für jeden Punkt $\boldsymbol{p}$ je nach seiner quadratischen Distanz zum Kamerazentrum $dist(\boldsymbol{p})^2$ die kleineste Oktalbaumauflösung ermittelt, in die dieser Punkt eingetragen wird. Der Faktor γ_{dist} legt dabei fest, wie stark das Rauschen zunimmt und wurde für den Kinect Ebenen Datensatz auf $\gamma_{dist} = 0,007$ festgelegt.

$$\epsilon_{min}(p) = \max(\epsilon_{min}, \gamma_{dist} * dist(p)^2) \tag{4.3}$$

Abbildungen 4.7 und 4.8 zeigen zwei Szenen des Kinect Ebenen Datensatzes, die beide einen hohen Anteil an korrekt segmentierten Flächen enthalten. Die Segmentierung der Szene aus Abbildung 4.7 enthält 12 korrekte Segmente, bei einer Punktüberlappung von 80%. In der händischen Segmentierung wurden 14 Flächen markiert. Damit ergibt sich eine Erkennungsrate von $85,7\%$ auf dieser Szene. Falsch zugeordnete Punkte befinden sich zum Beispiel auf der Vorderkante der rechten Regal Seitenwand. Hier wurden einige Punkte zum Segment der Schranktür zugeordnet. Der größte Fehler in der Segmentierung von Abbildung 4.8 ist die Teilung der Rückwand in zwei Segmente, da der schmale Streifen oberhalb des Monitors nicht zugeordnet wird. In diesem Bereich der unsegmentierten Punkte konnten keine stabilen Normalen berechnet werden, da in groben Auflösungen nur eine nahezu eindimensionale Ausdehnung der Punktmenge vorliegt. Bei der Normalenfilterung (siehe Kapitel 3 Abschnitt 3.1) wird die Normale als unstabil erkannt und diese Oktalbaumzelle nicht weiter zur Segmentierung benutzt. Aufgrund der Beschränkung der feinsten Auflösung je nach Entfernung eines Punktes, werden keine Knoten auf einer genügend feinen Auflösung generiert.

Abbildungen 4.9 und 4.10 zeigen durchschnittlich gut segmentierte Szenen des Kinect Ebenen Datensatzes. In beiden Abbildungen werden die Hauptebenen erkannt. In der rechten oberen Ecke der Szene aus Abbildung 4.9 ist nur in der 3D Ansicht erkennbar, dass die gemessenen Punkte dort nicht in einer Ebene mit den Punkten

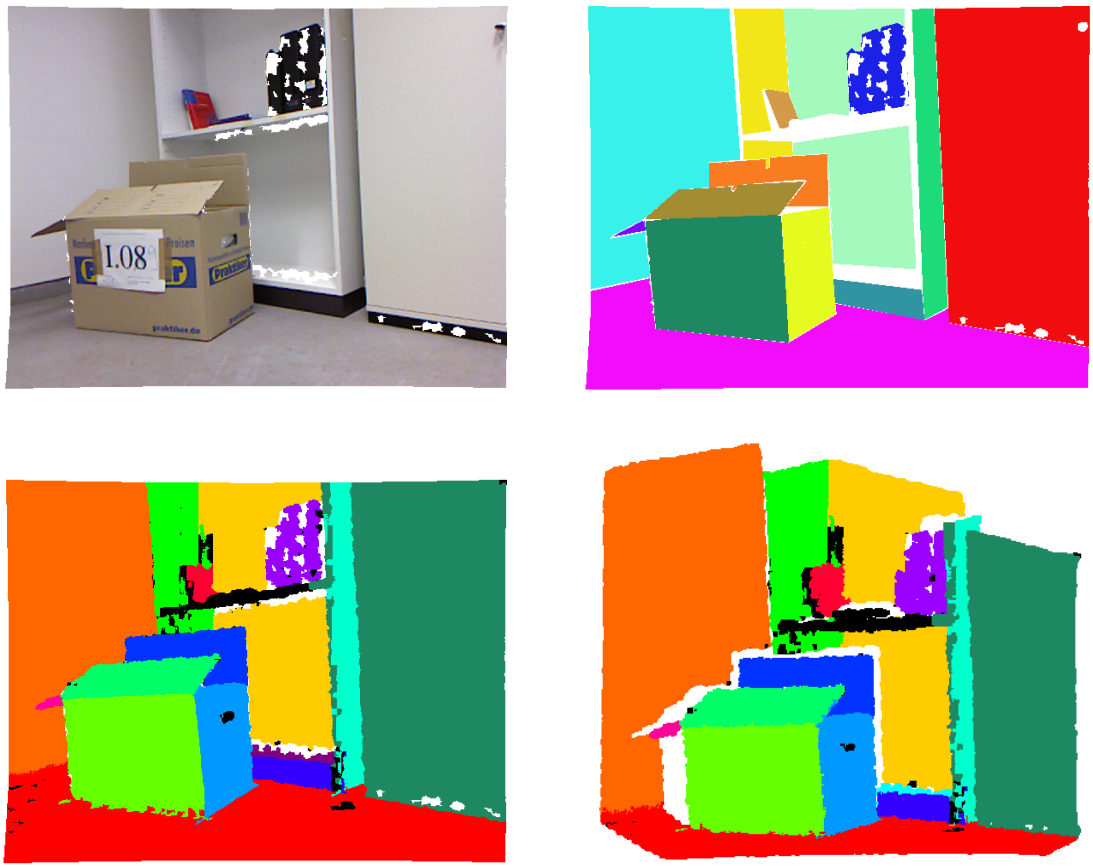


Abbildung 4.7: Gute Segmentierung einer Szene des Kinect Ebenen Datensatzes. Farbbild und händische Segmentierung (oben), maschinelle Segmentierung und 3D Bild (unten). Im Farbbild und in der händischen Segmentierung wurden Fehlmessungen schwarz markiert, die in der Punktwolke, die zur Segmentierung verwendet wurde, nicht enthalten sind. Daher sind diese Punkte in den unteren beiden Bildern nicht eingefärbt, also weiß. Weiße Punkte in der händischen Segmentierung sind Punkte auf nicht planaren Flächen. In der 3D Ansicht wurden unsegmentierte Punkte schwarz dargestellt. Da sie beim Vergleich mit der händischen Segmentierung nicht anders behandelt werden, als Fehlmessungen, wurden sie im Bild der maschinellen Segmentierung ebenfalls weiß eingefärbt.

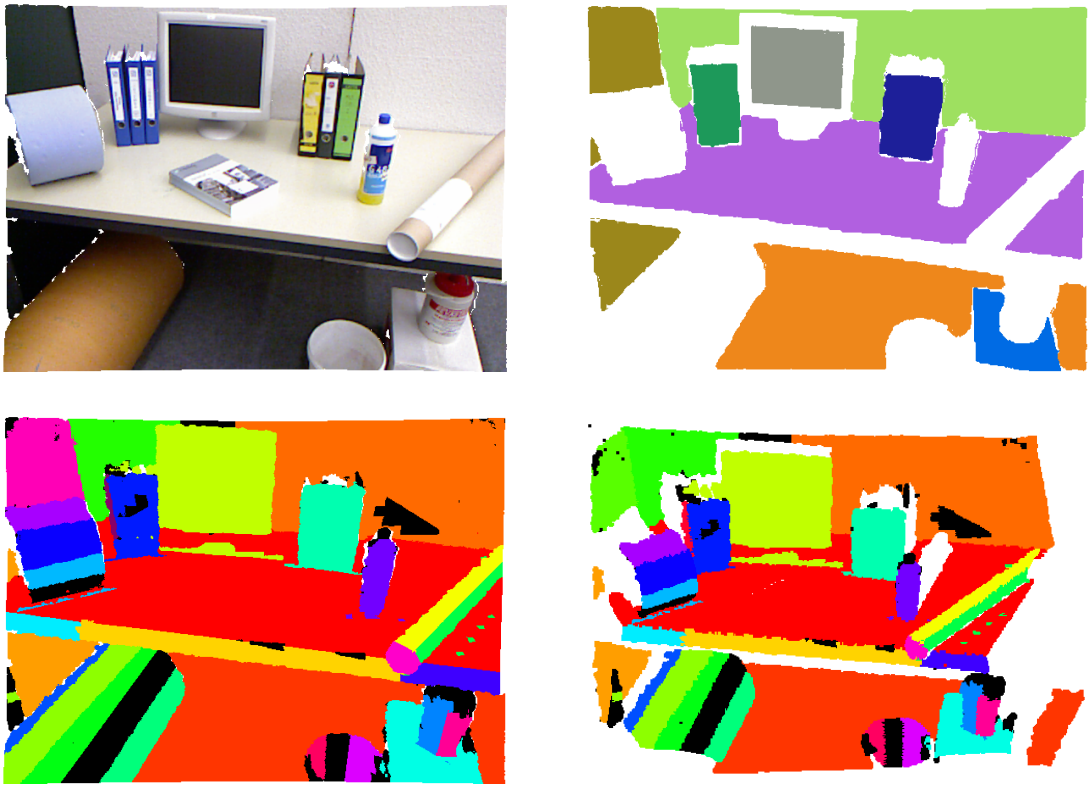


Abbildung 4.8: Gute Segmentierung einer Szene des Kinect Ebenen Datensatzes. Farbbild und händische Segmentierung (oben), maschinelle Segmentierung und 3D Bild (unten). Die Farbkodierung der Segmentierungsbilder ist wie in Abbildung 4.7 gewählt worden. Zylindrische Objekte wurden in der händischen Segmentierung als Fehlmessungen markiert. In der maschinellen Segmentierung wurden sie durch kleine planare Segmente angenähert.

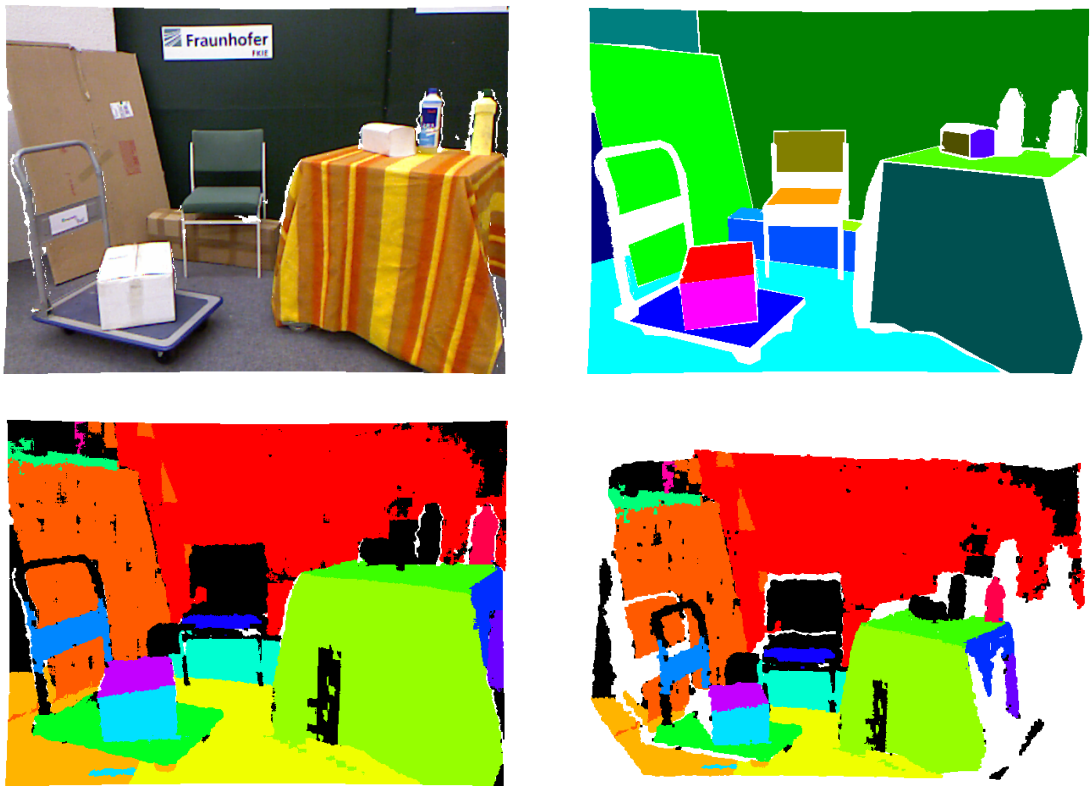


Abbildung 4.9: Durchschnittliche Segmentierung einer Szene des Kinect Ebenen Datensatzes. Farbbild und händische Segmentierung (oben), maschinelle Segmentierung und 3D Bild (unten). Die Farbkodierung der Segmentierungsbilder ist wie in Abbildung 4.7.

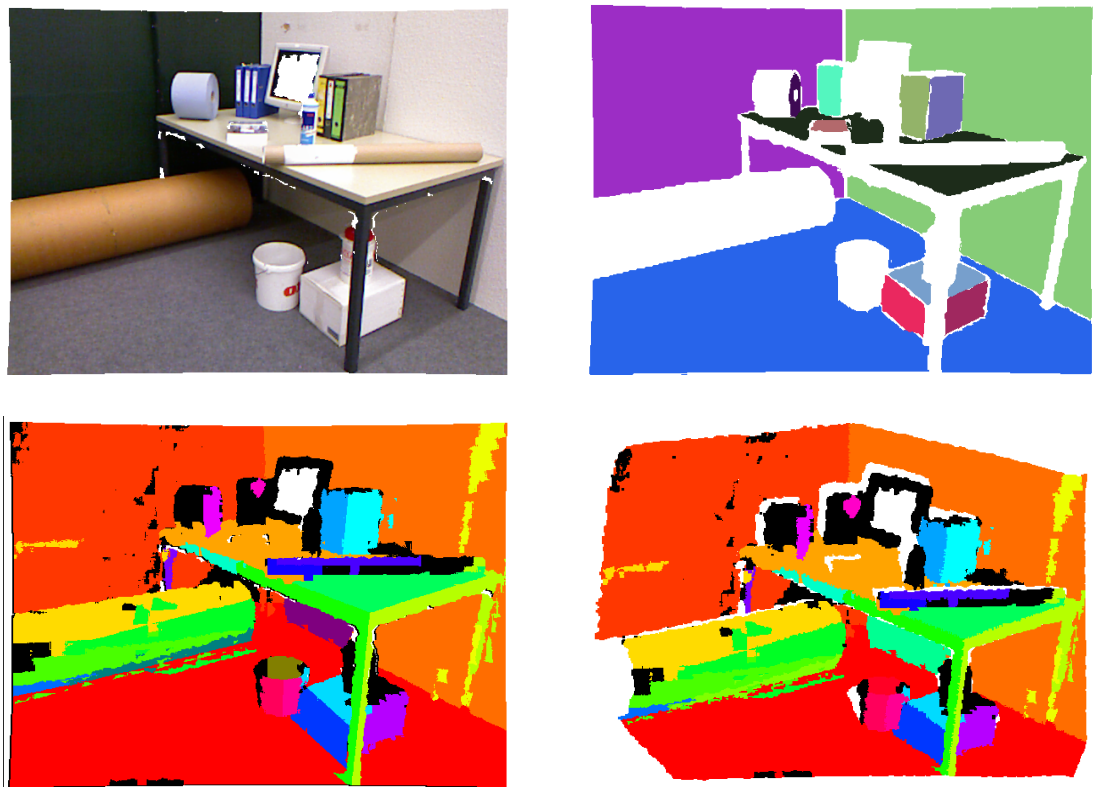


Abbildung 4.10: Durchschnittlich gute Segmentierung einer Szene des Kinect Ebenen Datensatzes. Farbbild und händische Segmentierung (oben), maschinelle Segmentierung und 3D Bild (unten). Die Farbkodierung der Segmentierungsbilder ist wie in Abbildung 4.7.

der roten Rückwandebene liegen. Die kleineren Flächen von Objekten im Vordergrund werden gut erkannt. Zum Teil verdeckte und kleine Flächen, die in größerer Entfernung liegen, werden nicht so gut detektiert. Auf Abbildung 4.10 ist zu sehen, dass die Tischebene übersegmentiert, also fälschlicherweise als zwei Segmente in der maschinellen Segmentietung (unten) detektiert wird. Einige der kleineren Flächen von den Objekten in der Szene werden ganz gut und richtig erkannt, bilden aber nicht in allen Fällen eine genügend große Punktüberlappung.

Gruppe	korrekt	über-segmentiert	unter-segmentiert	nicht gefunden	Rauschen
noRansac	1,5 (12,1%)	0,4	0,0	10,4	7,1
octree8	2,1 (16,9%)	0,6	0,0	9,7	7,0
octree9	3,2 (25,8%)	0,8	0,0	8,4	13,2
octree10	2,9 (23,4%)	1,1	0,0	8,3	15,1
ransacOnly	1,4 (11,3%)	1,5	0,0	9,5	33,2
AIS/FKIE	4,5 (36,3%)	0,6	0,4	6,8	16,2

Tabelle 4.5: Vergleich mit alternativen Segmentierungsverfahren auf dem Kinect Ebenen Datensatz mit 80% Überlappungstoleranz. Im Durchschnitt über alle 30 Bilder sind auf jedem $12,4$ Segmente abgebildet. Der Vergleich des Winkelfehlers entfällt, da zum vorliegenden Datensatz keine Winkelinformationen vorliegen. Die hohen Werte beim Rauschen resultieren daraus, dass in den Szenen vorkommende Zylinder in der händischen Segmentierung als Fehlmessung markiert wurden.

In Tabelle 4.5 ist die Güte der Segmentierung des Algorithmus auf dem Kinect Ebenen Datensatz zu sehen. Das hier vorgestellte Verfahren (AIS/FKIE) erkennt auf dem Kinect Ebenen Datensatz mehr Segmente korrekt, als alle Vergleichsverfahren. Durchschnittlich werden bei 80% Überlappungstoleranz $4,5$ von $12,4$ Flächen erkannt (AIS/FKIE). Das Vergleichsverfahren mit dem zweitbesten Ergebnis bei den korrekt detektierten Segmenten ist *octree9*. Die beiden Verfahren *ransacOnly* und *noRansac*, die nicht mit der Kombination aus RANSAC und Hough Transformation arbeiten, haben mit durschnittlich 1,4 beziehungsweise 1,5 korrekt segmentierten Flächen nur ca. 10% der Segmente der händischen Segmentierung mit einer Punktüberlappung von 80% oder mehr detektiert.

Die große Anzahl an Flächen, die als Rauschen klassifiziert worden sind, begründet darauf, dass nicht planare Teile der Szene in der händischen Segmentierung als Fehlmessungen markiert wurden. Dadurch wurde erreicht, dass diese Segmente nicht in die Anzahl der Segmente der händischen Segmentierung eingerechnet wird.

Abbildung 4.8 zeigt deutlich, dass ein einzelner Zylinder während der Ebenendetektion in mehrere planare Segmente unterteilt wird. Diese Segmente zählen alle in die Gesamtzahl der Segmente, die als Rauschen klassifiziert worden sind. Der hier vorgestellte Algorithmus detektiert $36,3\%$ durchschnittlichen Anzahl der händisch segmentierten Flächen bei einer Punktüberlappung von 80%.

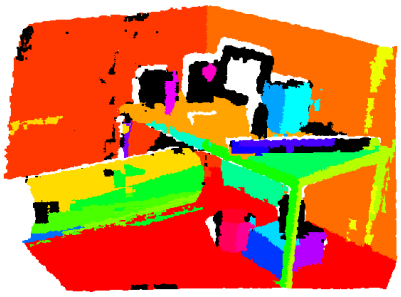
Abbildung 4.11 zeigt die Segmentierung der Szene aus Abbildung 4.10 durch die Vergleichsverfahren. Das Verfahren *noRansac* (Abb. 4.11b) erkennt die Hauptebenen mit einer geringeren Punktüberdeckung, da auch für diese Szene der Distanzschwellwert für Ebenen bei der Hough-Transformationen kleiner gewählt werde musste als bei dem Verfahren mit RANSAC Verbesserung aus Kapitel 3. Dadurch wurden auch insgesamt weniger Segmente detektiert. *RansacOnly* erkennt die Hauptebenen und die Tischebene gut, auch wenn im Segment der linken hinteren Wand (vgl. Abb. 4.11c) einige Punkte falsch zugeordnet wurden. Die kleineren Flächen der Objekte werden garnicht erkannt oder durch wenige Segmente untersegmentiert. Die Segmentierungen der Verfahren *octree[i]*, die auf einer Auflösungsstufe der Oktalbaumes Hough-Transformation, Zusammenhangskomponentensuche und RANSAC anwenden, sehen sich sehr ähnlich. Das liegt daran, dass durch die Beschränkung der Oktalbaumstufe beim Eintragen der Punkte (abhängig vom Quadrat der Distanz) einige Knoten des Oktalbaumes nicht in so feinen Auflösungsstufen vorliegen.

Überlappungstoleranz	51%	60%	70%	80%	90%
korrekt	6,7	6,3	5,5	4,5	2,5
übersegmentiert	1,3	1,1	0,9	0,6	0,4
untersegmentiert	0,7	0,6	0,6	0,4	0,1
nicht gefunden	3,3	4,1	5,3	6,8	9,8
Rauschen	9,1	10,0	11,4	13,2	16,2

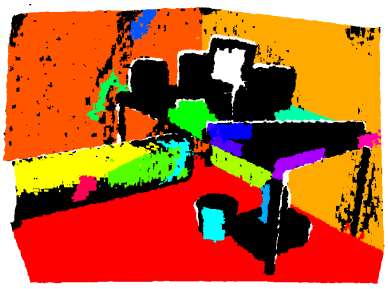
Tabelle 4.6: Durchschnittliche Anzahl an entsprechend klassifizierten Flächen pro Bild für unterschiedliche Überlappungstoleranzen auf den 30 Bildern des Kinect Ebenen Datensatzes. Durchschnittlich waren $12,8$ Segmente in der händischen Segmentierung enthalten.

Tabelle 4.6 zeigt die Klassifikation der maschinellen Segmentierung gegenüber der händischen Segmentierung im Durchschnitt über die 30 Szenen des Kinect Ebenen Datensatzes. Wie die Abbildungen 4.7, 4.8, 4.7 und 4.8 schon gezeigt haben, werden die großen Segmente fast immer mit hoher Punktüberlappung korrekt detektiert, wohingegen die kleinen Flächen oft nur eine geringe relative Punktüberlappung erreichen. Daher werden kleine Flächen, die erkannt werden, bei einer Überlappungsto-

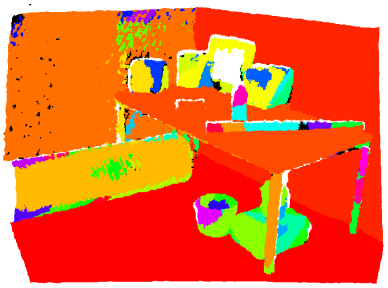
leranz von 80% nicht als korrekt, sondern als nicht gefunden klassifiziert. Die Tabelle zeigt, dass bei geringeren Punktüberlappungen wesentlich mehr Segmente als korrekt bewertet werden. Die durchschnittliche Anzahl der Segmente, die als nicht gefunden klassifiziert werden, halbiert sich für eine Überlappungstoleranz von 51% gegenüber einer Toleranz von 80%.



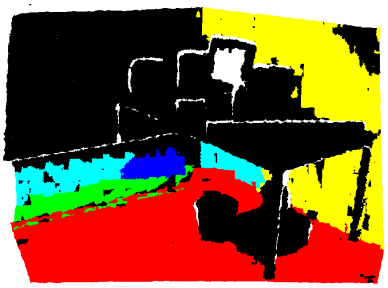
(a) AIS/FKIE



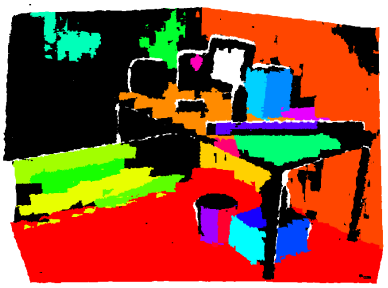
(b) noRansac



(c) ransacOnly



(d) octree8



(e) octree9

(f) octree10

Abbildung 4.11: Sechs verschiedene Segmentierungen der Kinect Ebenen Szene von Abbildung 4.10. Die Farbe eines Punktes zeigt die Segmentierung an. Schwarze Punkte wurden nicht segmentiert.

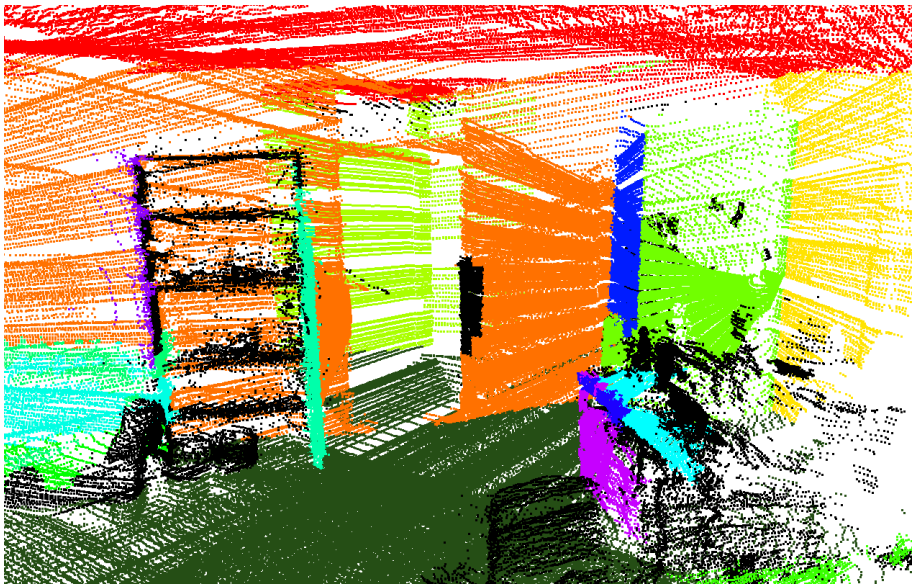


Abbildung 4.12: Planare Segmentierung einer unsortierten 3D Punktwolke. Alle Punkte eines Segmentes haben die gleiche Farbe. Schwarze Punkte sind unsegmentiert. Deutlich zu sehen ist, dass alle großen Ebenensegmente korrekt gefunden wurden. Viele kleine beziehungsweise nicht planare Segmente, wie die Regalbretter zwischen der rechten (mintgrün) und der linken (lila) Seitenwand, wurden nicht segmentiert. Im Türrahmen, an der orangenen Wandfläche, fehlen einige Punkte.

4.1.4 unsortierte Punktwolken

In diesem Abschnitt wird gezeigt, dass mit dem hier vorgestellten Verfahren unsortierte Punktwolken segmentiert werden können. Da für diese Art von ungeordneter Punktwolke keine händische Segmentierung von Tiefenbildern möglich ist, konnte auf diese Weise keine qualitative Auswertung der Ergebnisse gemacht werden.

Abbildung 4.12 zeigt die planare Segmentierung einer unsortierten 3D Punktwolke. Auf dem Bild ist zu sehen, dass alle großen Ebenen detektiert wurden.

Abbildung 4.13 zeigt eine zweite unsortierte 3D Punktwolke, die mit einem Laser aufgenommen wurde, der sich im Greifer eines Manipulatorarmes befindet. Durch die Winkelstellung der Gelenke des Armes wurde die 3D Position und Aufnahmerichtung des Laserscanners ermittelt und damit die Punktwolke erstellt. Aufgrund der Bewegungen des Armes ist eine Interpretation der zeitlich aufeinander folgenden Scans als Tiefenbild nicht sinnvoll. Alle großen Ebenen wurden in der Segmentie-

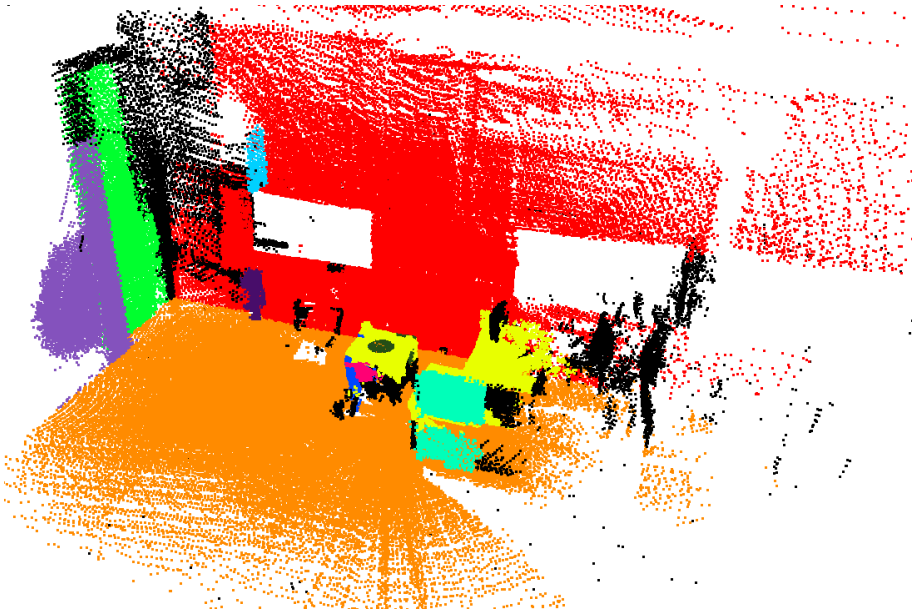


Abbildung 4.13: Segmentierte unsortierte 3D Punktwolke. Die Farbe der Punkte kodiert die Zugehörigkeit zum Segment. Schwarze Punkte wurden keinem Segment zugeordnet. Orangene Punkte bilden den Fußboden. Punkte auf der Rückwand (mit zwei Fenstern) wurden im roten Segment zusammengefasst. Der Manipulatortisch und die Tische daneben wurden in ein Segment mit gelber Farbe gruppiert. Die Wand auf der linken Seite des Bildes ist in mehrere Teile aufgeteilt, die großen Teile dieser gestückelten Wand wurden richtig segmentiert (grün und lila), die anderen Teile wurden nicht gefunden.

rung gefunden. Die Punkte, die keinem Ebenensegment zugeordnet wurden, liegen entweder weit auseinander, oder bilden keine planare Fläche.

4.1.5 Empirische Laufzeitbestimmung

Die Laufzeiten der Verfahren wurden auf einem Intel Core 2 Duo Processor mit 2,4 GHz ermittelt. Es wurde die durchschnittliche Laufzeit über die je 30 Szenen aller drei Datensätze berechnet und auch die Standardabweichung bestimmt.

Tabelle 4.7 zeigt die durchschnittlichen Laufzeiten der Segmentierung der einzelnen Verfahren. Das hier präsentierte Verfahren benötigt auf dem ABW Datensatz durchschnittlich $1,8$ Sekunden, auf dem Perceptron Datensatz $2,6$ Sekunden und auf dem Kinect Ebenen Datensatz $2,1$ Sekunden. Die Standardabweichung der Laufzeit ist

	ransacOnly	noRansac	octree8	octree9	octree10	AIS/FKIE
ABW						
mittlere Laufzeit	13,200s	3,163s	0,555s	0,655s	1,001s	1,824s
std. Abw.	12,411s	0,462s	0,048s	0,046s	0,092s	0,212s
Perceptron						
mittlere Laufzeit	2,559s	2,800s	0,740s	0,896s	0,517s	2,592s
std. Abw.	1,894s	0,462s	0,057s	0,076s	0,032s	0,162s
Kinect Ebenen						
mittlere Laufzeit	6,623s	4,436s	0,494s	0,711s	0,872s	2,134s
std. Abw.	2,241s	1,155s	0,041s	0,081s	0,162s	0,340s

Tabelle 4.7: Laufzeiten und Standardabweichungen der unterschiedlichen Verfahren auf den drei Ebenen Datensätzen in Sekunden. Diese durchschnittlichen Zeiten und die Standardabweichung wurden beim Erzeugen der Ergebnisse aus den Tabellen 4.1, 4.3 und 4.5 gemessen.

bei diesem Verfahren auf allen Datensätzen weniger als 16% des Mittelwertes. Dies zeigt an, dass die Laufzeit des Verfahrens abhänging von der individuellen Komplexität der Szene ist. Die Abwandlungen der hier vorgestellten Segmentierungen, die nur auf einer Oktalbaumstufe arbeiten (*octree[i]*) konnten die Segmentierungen dieser einzelnen Auflösungen zwar schneller bewältigen, aber auch nur weniger korrekt klassifizierte Segmente finden (vergleiche Tabellen 4.1, 4.3 und 4.5). Wie bereits beschrieben, musste für die Segmentierung mit *noRansac* der Parameter für die Punktdistanz zur Ebene kleiner gewählt werden (im Vergleich zu AIS/FKIE), um die nicht durchgeführte Verbesserung durch RANSAC zu kompensieren, was die Laufzeit deutlich erhöhte. Es konnte dennoch keine ähnlich gute Segmentierung, bezüglich eines Überlappungsschwellwertes von 80%, erreicht werden. Aufgrund der höheren Ausreißerquote (gegenüber AIS/FKIE) beim *ransacOnly*-Verfahren musste eine größere Gesamtzahl an Iterationen gewählt werden (vergleiche Tabelle 2.1 in Kapitel 2). Die höhere Ausreißerquote erklärt sich daher, dass keine Vorsegmentierung stattfindet. Wegen der erhöhten Anzahl an Iterationen ergab sich eine wesentlich höhere Laufzeit des Verfahrens. Aufgrund eines Konvergenz Kriteriums in der hier verwendeten RANSAC-Implementierung, bricht das Verfahren manchmal nach wenigen Iterationen ab. Daher ist die Standardabweichung der Laufzeit bei diesem Verfahren besonders groß (94% der mittleren Laufzeit auf dem ABW Datensatz).

Um die Laufzeit des Verfahrens auch auf den kleineren Auflösungen der Microsoft Kinect zu zeigen, wurden 30 Punktwoken der Szene aus Abbildung 4.7 aufgenom-

Auflösung	mittlere Laufzeit	Standard-abweichung
VGA	2,587s	0,648s
QVGA	0,350s	0,039s
QQVGA	0,105s	0,024s

Tabelle 4.8: Mittlere Laufzeiten für verschiedene Kinect Auflösungen in Sekunden.

men. Während der Aufnahmen wurde die Kamera nicht bewegt, so dass die Szene sich nicht geändert hat. Die Zeit wurde für die Segmentierung von 30 Punktwolken der gleichen Szene bestimmt und im Anschluss wurde die mittlere Laufzeit berechnet. Tabelle 4.8 zeigt die Laufzeiten gemittelt über 30 Szenen und die zugehörige Standardabweichung. Um Punktwolken dieser Szene in der vollen VGA Auflösung (640×480) segmentieren zu können wurden 2,5 Sekunden benötigt. Innerhalb von einer Sekunden können knapp drei Punktwolken mit 320×240 Punkten (QVGA Auflösung) vom hier vorgeschlagenen Verfahren segmentiert werden. In Punktwolken der QQVGA Auflösung (160×120 Punkte) können mit einer Frequenz von knapp 10Hz planare Segmente detektiert werden.

4.1.6 Diskussion der Teilergebnisse

Anhand von verschiedenen Datensätzen wurde gezeigt, dass das hier vorgestellte Verfahren zur Ebenendetektion effizient und robust Ebenensegmente in unsortierten 3D Punktwolken erkennt. Dünn mit Punkten besetzte Flächen werden dabei schlechter erkannt, starkes Rauschen der Punkte erschwert die Berechnung der Normalen auf feinen Auflösungsstufen und damit die Segmentierung zusätzlich. Die durchschnittliche Erkennungsrate bei 80% Punktüberlappung war mit 73% auf dem ABW Datensatz und $50,1\%$ auf dem Perceptron Datensatz zwar etwas niedriger als die Erkennungsraten der Tiefenbild basierten Verfahren, aber dafür ist das hier vorgestellte Verfahren allgemeiner und kann auch unsortierte Punktwolken segmentieren. Der Winkelfehler war ähnlich gering, wie bei den anderen Verfahren, was zeigt, dass die Ebenenparameter der korrekt segmentierten Flächen ebenso genau geschätzt wurde, wie bei den Verfahren mit etwas höherer Erkennungsrate. Auf dem Kinect Ebenen Datensatz wurde die Segmentierung auf sehr verrauschten Daten gezeigt. Auf diesem sehr schwierigen Datensatz wurden kleinere Objektflächen auf

größere Entfernung aufgrund des Punktrauschens nicht oder nur unvollständig gefunden. Die Hauptebenen wurden vom hier vorgestellten Verfahren meistens auch mit 80% Überlappungsschwellwert korrekt detektiert.

4.2 Zylindersegmentierung

Zur Auswertung der Zylindersegmentierung standen keine Ergebnisse von Segmentierungen anderer Verfahren zur Verfügung, daher wurden hier ausschließlich die bereits bei den Ebenen verwendeten Varianten des hier vorgestellten Verfahrens zum Vergleich der Segmentierungsqualität benutzt. Das Verfahren *noRansac* ist eine Variante, bei welcher der RANSAC-Schritt zur Verbesserung der Zylinderparameter entfällt. *RansacOnly* bezeichnet eine Implementierung des RANSAC-Verfahrens, bei der das beste Vorkommen eines Zylinders in der Punktwolke gesucht und die beteiligten Punkte aus der Punktwolke entfernt werden. Anschließend wird der nächstbeste Zylinder gesucht. Die anderen drei Vergleichsverfahren *octree[i]* nutzen wie das in dieser Arbeit vorgestellte Verfahren Hough-Transformation und RANSAC. Allerdings sind diese Verfahren auf genau eine Oktalbaumtiefe (und alle Blätter von darüber liegenden Oktalbaumtiefen) beschränkt.

4.2.1 Kinect Zylinder Datensatz

Der Kinect Zylinder Datensatz besteht aus 30 Szenen, die mit einer Microsoft Kinect aufgezeichnet wurden. In der händischen Segmentierung wurden alle Bildpunkte, die auf der Mantelfläche eines zylindrischen Objektes liegen, als Segment markiert. Alle Bildpunkte auf nicht-zylindrischen Oberflächen wurden als Fehlmessung gekennzeichnet. Damit ist der Anteil der Bildpunkte der händischen Segmentierung, die ein Segment markieren, wesentlich geringer als der Anteil der Bildpunkte, die als Fehlmessung gekennzeichnet worden sind. Um die Anzahl der Falschdetektionen zu reduzieren, wurde ein Test implementiert, der bewirkt, dass Zylinder mit weniger als 10% belegter Mantelfläche verworfen werden.

Da die Szenen beziehungsweise die Zylinder in den Szenen nicht vermessen wurden, stehen zu diesem Datensatz keine Informationen zur Verfügung, die eine qualitative Auswertung der geschätzten Parameter möglich macht.

Auch bei der Segmentierung auf diesem Datensatz wurde das Fehlermodell für die Punktmessungen der Kinect benutzt, bei dem die minimale Volumengröße im Oktalbaum begrenzt wird, in die ein Punkt eingetragen wird. Dieses minimale Volumen berechnet sich aus der quadratischen Distanz zum Kamerazentrum.

Durch diese Beschränkung des kleinsten Volumens für einen Punkt im Oktalbaum kann es passieren, dass einige Auflösungsstufen die gleichen Blattknoten beinhalten und sich dadurch wenig unterscheiden. In diesem Fall werden aber bei der Bildung der Paare für die Hough-Transformation von Zylindern (siehe Kapitel 3 Abschnitt 3.2), die Knoten nicht mehr berücksichtigt, die in darüberliegenden Auflösungsstufe schon Paare mit all ihren Nachbarn gebildet haben.

Abbildung 4.14 zeigt eine Szene des Kinect Zylinder Datensatzes mit zwei korrekt segmentierten Zylindern (bei 80% Punktüberlappung). Auf beiden Zylindern fehlen ein paar Punkte. Das blaue Segment in der maschinellen Segmentierung wurde falsch auf die Punkte des Fußbudens ausgeweitet. Dies ist die einfachste Szene des Datensatzes, da nur zwei große Zylinder in relativ geringer Distanz zum Kamerazentrum aufgenommen wurden. Dadurch ist das Sensorrauschen in dieser Szene besonders gering.

Die Szene aus Abbildung 4.15 enthält drei Zylinder, die in der händischen Segmentierung markiert wurden. Die maschinelle Segmentierung detektiert zwei dieser Zylinder. Der vom Verfahren geschätzte Radius des roten Zylinders (maschinelle Segmentierung) ist etwas zu groß, aber dennoch klein genug, so dass die Punkte korrekt zugeordnet werden können. Der dritte Zylinder wird aufgrund seiner geringen Größe und der Deformierung der Oberfläche nicht als Zylinder erkannt. Das blau markierte Segment in der maschinellen Segmentierung von Abbildung 4.15 ist kein Zylinder, sondern ein quaderförmiges Objekt, das fälschlicherweise als Zylinder erkannt wird.

Abbildung 4.16 zeigt eine Szene des Kinect Zylinder Datensatzes mit fünf markierten zylindrischen Objekten in der händischen Segmentierung. Vier der Zylinder werden vom in diser Arbeit vorgestellten Verfahren erkannt, auch wenn die Punktüberlappung beim länglichen Objekt (blau in der maschinellen Segmentierung) nicht groß ist, also einige der Objektpunkte nicht korrekt zugeordnet werden. Der kleine Eimer auf dem Fußboden (lila in der händischen Segmentierung), wird nicht erkannt. Das Objekt ist zu weit entfernt, so dass wegen des Modells für das Sensorrauschen der Kinect, der Oktalbaum in dieser Entfernung nicht fein genug aufgelöst wird, um in der Surfelmenge Zylinder zu erkennen. Zusätzlich werden in dieser Szene viele kleine Zylinder falsch detektiert. Vier Zylinder werden auf dem Rand eines korrekt

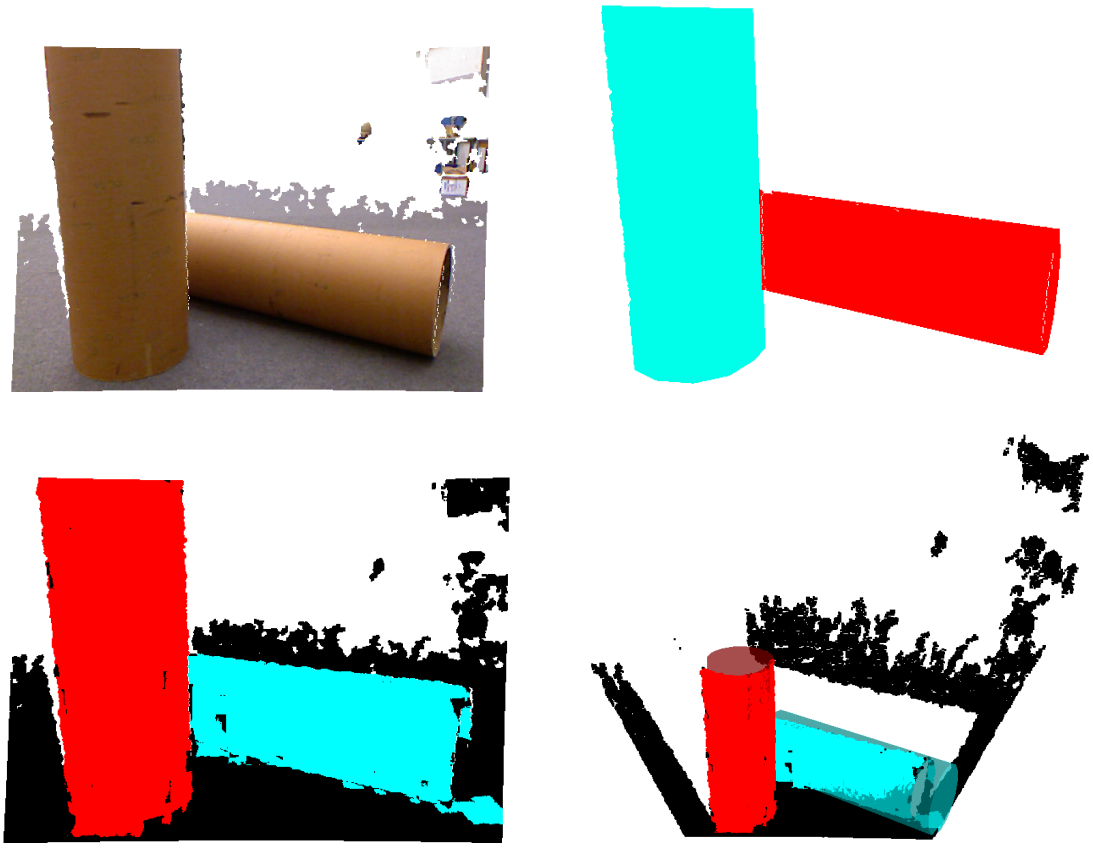


Abbildung 4.14: Segmentierung einer Szene des Kinect Zylinder Datensatzes. Farbbild und händische Segmentierung (oben), maschinelle Segmentierung und 3D Bild (unten). Weiße Pixel markieren in den Bildern, dass dort keine Messung vorliegt, oder kein Segment markiert wurde (händische Segmentierung). In der maschinellen Segmentierung und der 3D Ansicht wurden alle Punkte, die zu keinem zylindrischen Segment gehören schwarz markiert. In der 3D Ansicht wurden die geschätzten Zylinder leicht transparent abgebildet.

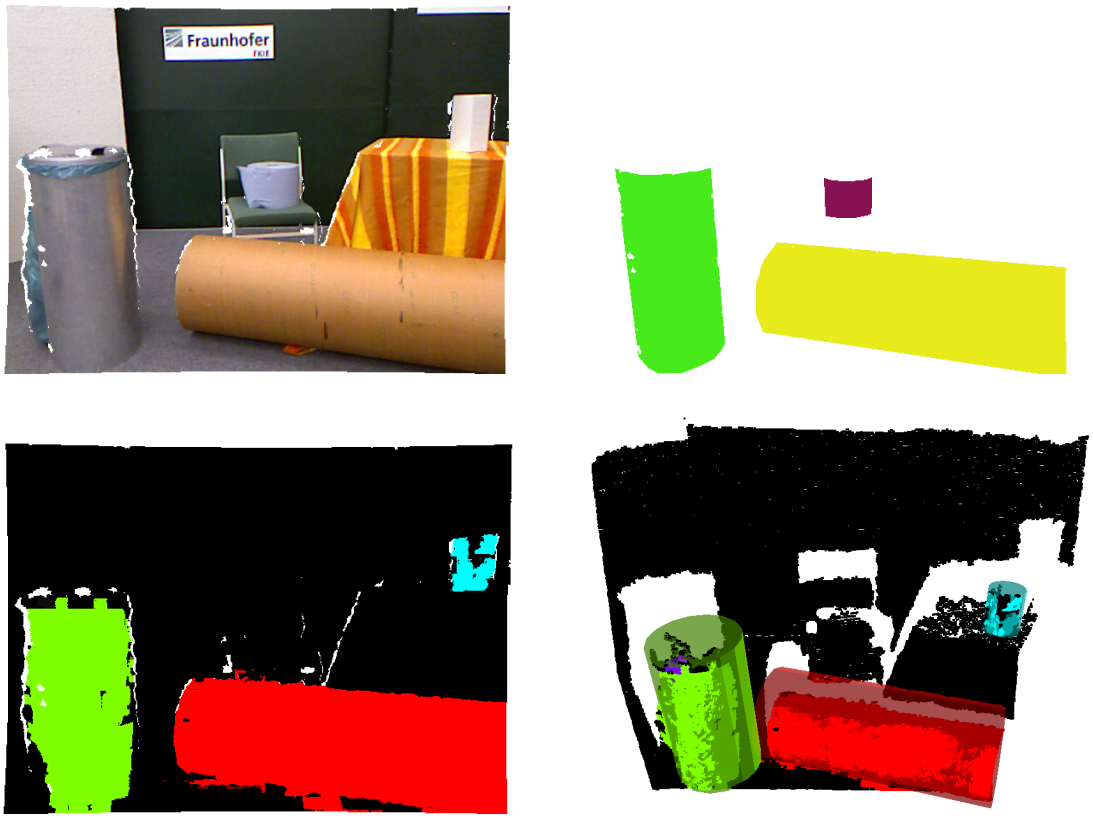


Abbildung 4.15: Segmentierung einer zweiten Szene des Kinect Zylinder Datensatzes. Farbbild und händische Segmentierung (oben), maschinelle Segmentierung und 3D Bild (unten). Die Farbkodierung der einzelnen Bilder ist identisch zu Abbildung 4.14.

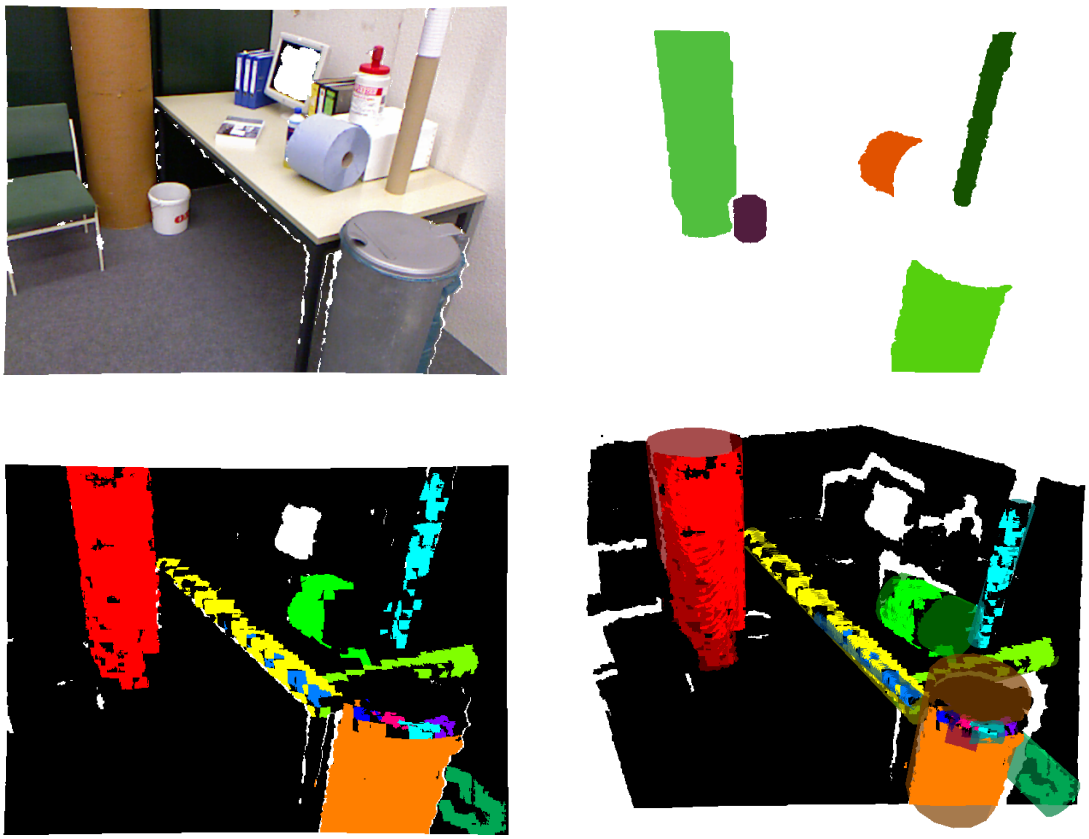


Abbildung 4.16: Segmentierung einer Szene des Kinect Zylinder Datensatzes. Farbbild und händische Segmentierung (oben), maschinelle Segmentierung und 3D Bild (unten). Die Farbkodierung der einzelnen Bilder ist identisch zu Abbildung 4.14.

gefundenen Zylinder erkannt, drei weitere Segmente liegen durch die Vorder- und Seitenkante des Tisches und ein Zylinder wird zwischen der rechten Wand und dem Fußboden detektiert. Diese falsch erkannten Segmente werden als Rauschen klassifiziert und müssten durch einen Nachverarbeitungsschritt gefiltert werden.

Gruppe	korrekt	über-segmentiert	unter-segmentiert	nicht gefunden	Rauschen
noRansac	0,73 (22,1%)	0,00	0,00	2,56	13,97
octree8	0,53 (16,0%)	0,00	0,00	2,77	1,53
octree9	0,50 (15,1%)	0,00	0,00	2,80	2,20
octree10	0,20 (6,0%)	0,00	0,00	3,10	3,73
ransacOnly	0,60 (18,2%)	0,00	0,00	2,7	0,20
AIS/FKIE	1,13 (34,2%)	0,07	0,00	2,10	6,30

Tabelle 4.9: Vergleich mit alternativen Segmentierungsverfahren auf dem Kinect Zylinder Datensatz mit 80% Überlappungstoleranz. Im Durchschnitt über alle 30 Bilder sind auf jedem $3,3$ Segmente abgebildet. Die hohen Werte beim Rauschen resultieren daraus, dass in den Szenen hauptsächlich vorkommende Ebenen in der händischen Segmentierung als Fehlmessung markiert wurden.

In Tabelle 4.9 wurde die Segmentierungsqualität der implementierten Varianten miteinander verglichen. Die beste relative Erkennungsrate von $34,2\%$ erzielt auf diesem Datensatz das in dieser Arbeit entwickelte Verfahren (AIS/FKIE), das durchschnittlich 1,13 Zylinder in der Szene korrekt detektiert. *ransacOnly* findet bei sehr geringerem Rauschen noch 0,6 (18,2%) Segmente, die korrekt klassifiziert werden. Das Verfahren *noRansac*, bei dem kein filternder RANSAC-Schritt durchgeführt wird, weist mit Abstand die meisten Segmente auf, die als Rauschen klassifiziert wurden, ohne dass besonders viele zylindrische Segmente korrekt gefunden werden. Die *octree[i]*-Verfahren erzielen geringe Erkennungsraten, da sie die Szene nur auf einer Auflösungsstufe betrachten, die Zylinder aber sehr unterschiedlich groß sind.

Abbildung 4.17 zeigt 3D Ansichten der Segmentierungen der Szene aus Abbildung 4.15 durch die Vergleichsverfahren. Die Segmentierung des in dieser Arbeit gezeigten Algorithmus (AIS/FKIE, Abbildung 4.17a) wurde bereits erläutert. Das Verfahren *noRansac* (vgl. Abb. 4.17b) findet in dieser Szene sehr viele zylindrische Segmente. Nur das rote und das orangene Segment sind korrekt segmentiert, die anderen Segmente enthalten fast ausschließlich Punkte von planaren Flächen. Der dunkelblaue Zylinder in der Mitte der Szene liegt zwar an einer richtigen Stelle, allerdings ist er

falsch orientiert, so dass nur sehr wenige Punkte korrekt zugeordnet werden. *RansacOnly* findet in der Szene nur ein zylindrisches Segment mit sehr wenig Punktüberlappung (siehe Abb. 4.17c). Bei *octree8* (Abb. 4.17d) und *octree9* (Abb. 4.17e) werden die beiden großen Zylinder der Szene richtig erkannt, allerdings fehlen einige der Punkte. Auf der nächstfeineren Auflösung (vgl. Abb. 4.17f) sind die Oktalbaumzellen so klein, dass die berechneten Normalen zu unsicher werden. Dadurch ist die Detektion eines Zylinders schwierig. Das Verfahren findet noch einige Punkte, die einen Zylinder bilden, allerdings werden nur so wenig Punkte in ein Segment zugeordnet, dass keine 51% Punktüberlappung erreicht werden.

Überlappungstoleranz	51%	60%	70%	80%	90%
korrekt	1,9	1,73	1,60	1,13	0,60
übersegmentiert	0,27	0,17	0,13	0,07	0,00
untersegmentiert	0,00	0,00	0,00	0,00	0,00
nicht gefunden	1,13	1,40	1,57	2,10	2,70
Rauschen	5,13	5,50	5,70	6,3	6,97

Tabelle 4.10: Durchschnittliche Anzahl an entsprechend klassifizierten Flächen pro Bild für unterschiedliche Überlappungstoleranzen auf den 30 Szenen des Kinect Zylinder Datensatzes. Durchschnittlich waren $3,3$ Segmente in der händischen Segmentierung enthalten. Verglichen wurde mit der maschinellen Segmentierung des in dieser Arbeit vorgeschlagenen Algorithmus (AIS/FKIE).

Tabelle 4.10 zeigt die Veränderung der Segmentierungsgüte bei unterschiedlichen Überlappungstoleranzen. Das Verfahren findet im Kinect Zylinder Datensatz durchschnittlich $1,13$ Segmente bei 80% Überlappungstoleranz. Das ergibt bei durchschnittlich $3,3$ Segmenten pro Bild eine Erkennungsrate von $34,2\%$. Bei 70% Punktüberlappung erreicht das Verfahren sogar eine Erkennungsrate von $48,5\%$. Die Anzahl der Übersegmentierten Oberflächen ist gering, untersegmentiert wurde kein Segment. Insgesamt ist die Anzahl der Flächen in der maschinellen Segmentierung pro Szene, die als Rauschen klassifiziert wurde, sehr hoch. Bei 51% Punktüberlappung sind dies $5,13$ Flächen, also knapp 3 mal so viele, wie korrekt gefundene Zylinder ($1,9$ Flächen). Dies liegt unter anderem daran, dass die Parameter der Filterschritte, welche die Anzahl der falsch erkannten Zylinder verringern, so gewählt wurden, dass keine korrekt erkannten Zylinder verworfen wurden.

Tabelle 4.11 zeigt die mittleren Laufzeiten und Standardabweichungen des Verfahrens auf dem Kinect Zylinder Datensatz. Das Verfahren *ransacOnly* benötigte im

	ransacOnly	noRansac	octree8	octree9	octree10	AIS/FKIE
mittlere Laufzeit	393,330s	1,761s	75,747s	31,050s	10,637s	15,394s
std. Abw.	303,807s	0,252s	38,566s	15,170s	2,677s	8,667s

Tabelle 4.11: Laufzeiten und Standardabweichungen der unterschiedlichen Verfahren auf dem Kinect Zylinder Datensatz in Sekunden. Diese durchschnittlichen Zeiten und die Standardabweichung wurden beim Erzeugen der Ergebnisse aus Tabelle 4.9 gemessen. Für die Durchführung der Experimente über die 30 Szenen des Datensatzes wurde ein Intel Core 2 Duo Prozessor mit 2,4 GHz verwendet.

Durchschnitt 6,5 Minuten für eine Szene. Für die zweite Szene des Datensatzes wurde eine Laufzeit von insgesamt 2045,85 Sekunden, also ca. 34 Minuten. Diese Laufzeit wurde erreicht, weil sehr viele Iterationen durchgeführt wurden. Da aber sehr viele Punkte auf planaren Flächen statt auf Zylindern liegen, konnten nur kleine Punktmengen aus der Punktwolke entfernt werden. Das in dieser Arbeit vorgestellte Verfahren benötigte auf dem Datensatz durchschnittlich 15,394 Sekunden. Ein Großteil der Zeit wurde dabei auf die Berechnung der Normalen verwendet, die für den RANSAC-Schritt erforderlich sind. Das Verfahren *noRansac*, welches keinen solchen Schritt macht, brauchte im Vergleich nur durchschnittlich 1,76 Sekunden pro Szene des Datensatzes.

4.2.2 Diskussion der Teilergebnisse

In diesem Abschnitt wurde gezeigt, dass mit dem hier vorgestellten Verfahren Zylinder in komplexen Szenen mit hohem Punktrauschen gefunden werden können. Die Erkennungsrate des in dieser Arbeit beschriebenen Verfahrens bei $34,1\%$ bezüglich einer Überlappungstoleranz von 80%. Weil die Szenen im Durchschnitt nur wenige Zylinder enthielten, und da viele der Zylinder in den Szenen recht klein im Vergleich zum Rauschen des Sensors waren, konnten keine höheren Erkennungsraten erzielt werden.

Die Normalenberechnung für den RANSAC-Schritt wurde für jeden Ausschnitt der Punktwolke, den die Hough-Transformation vorsegmentiert hat, neu gestartet. Die Punkte für die Normalenberechnung wurden dabei in einem ähnlich großen Bereich ausgewählt, wie die Oktalbaumknotengröße auf dieser Berechnungsstufe vorgab. Dadurch wurden auf groberen Auflösungen sehr viele Punkte ausgewertet, was sehr

viel Zeit in Anspruch nahm. Durch Beschleunigung der Normalenberechnung mit Bereichsanfragen im Oktalbaum könnte die Laufzeit verkürzt werden.

(a) AIS/FKIE **(b)** noRansac

(c) ransacOnly **(d)** octree8

(e) octree9 **(f)** octree10

Abbildung 4.17: 3D Ansichten der sechs verschiedenen Segmentierungen der Kinect Zylinder Szene von Abbildung 4.15. Die Farbe eines Punktes zeigt die Segmentierung an. Schwarze Punkte wurden nicht segmentiert.

5 Diskussion

5.1 Zusammenfassung

Es wurde in dieser Arbeit ein Verfahren entwickelt, um in 3D Punktwolken von komplexeren Szenen Objekte finden zu können und um kolorierte 3D Punktwolken effizient übertragen zu können. Das Verfahren detektiert in unsortierten 3D Punktwolken entweder zylindrische, oder planare Segmente.

Es wurde ein Oktalbaum zur Repräsentation der Punktwolke durch Surfels auf verschiendenen Auflösungsstufen und zur effizienten Berechnung der Surfelnormalen benutzt. Die Segmentierung wurde von grob nach fein durchgeführt, wobei auf jeder Auflösungsstufe bereits gefundene Segmente durch Zuordnung der Surfels unter Berücksichtigung der Normalen und der Schwerpunkte verfeinert wurden, bevor die Detektion weiterer Segmente begonnen wurde. Bei der Erkennung von Formprimitiven auf einer Auflösungsstufe wurden die beiden populären Verfahren RANSAC und Hough-Transformation kombiniert. Die Hough-Transformation wurde verwendet, um eine Vorsegmentierung der Surfel vorzunehmen, welche die Modellparameter nur grob festlegt und größere Abweichungen davon zulässt. Mithilfe des RANSAC-Verfahrens wurden die Modellparameter verfeinert und die Punkte bestimmt, die auf der Modell Oberfläche liegen. Diese Punktzuordnung wurde mit geringen Abweichungen der Punktpositionen durchgeführt.

Bei der Segmentierung der Szene in Ebenen wurden zwischen der Vorsegmentierung und der Verfeinerung der Parameter die Zusammenhangskomponenten auf einem Gitter in der Ebene bestimmt. Die Ebenensegmentierung wurde nach der Detektion aller Segmente auf allen Auflösungsstufen noch durch zwei Nachverarbeitungsschritte verbessert. Im ersten Schritt wurden koplanare Ebenensegmente, die aneinander angrenzten, zu einem Segment zusammengefasst. Anschließend wurden bisher unsegmentierte Punkte zu bereits gefundenen Ebenen zugeordnet, ohne dabei Normaleninformationen zu berücksichtigen. Um die Zuordnung zwischen zwei Ebenen im

Bereich der Schnittgeraden zu verfeinern, wurde eine Mittelebene benutzt, die den Winkel zwischen den Ebenen gleichmäßig teilt.

Farbwerte der 3D Punkte und ihre Position wurden bezüglich der Ebenen diskretisiert und in einem Gitter gespeichert, so dass sie als Textur und Reliefkarte angezeigt werden können. Die Reliefkarte kann auch dazu verwendet werden, eine Approximation der Punktwolke zu erhalten.

Die Experimente haben gezeigt, dass die Ebenensegmentierung effizient und auf gering verrauschten Daten robust arbeitet. Auf stärker verrauschten Daten werden die Hauptebenen aber dennoch zuverlässig detektiert. Anhand der vorgeführten Erkennung von zylindrischen Segmenten wurde gezeigt, dass auch höherwertige Formprimitive mit diesem Verfahren detektiert werden können. Allerdings ist die Detektion von Zylindern empfindlicher gegenüber verrauschten Daten.

5.2 Ausblick

Bei der Zylindersegmentierung wurde noch kein Test auf Zusammenhangskomponenten implementiert, die Auswertung hat ergeben, dass dadurch noch eine Verbesserung der Segmentierung erreicht werden könnte. Außerdem wurde für die Zylindersegmentierung noch kein Nachverarbeitungsschritt implementiert. Die Verteilung der übrigen Punkte könnte anhand der Distanz zur geschätzten Oberfläche des Zylinders erfolgen. Dadurch könnten nicht zugeordnete Punkte der Mantelfläche noch verteilt werden. Texturen und Reliefkarten wurden für die Zylinder noch nicht implementiert.

Erweiterungen des Algorithmus ließen sich durch Ergänzung weiterer Formprimitive, wie zum Beispiel Kugeln oder Tori, schaffen. Außerdem könnte die kombinierte Suche nach mehreren Formprimitiven gleichzeitig untersucht werden, wie in [14], statt der nacheinander erfolgenden unabhängigen Segmentierung, die hier vorgestellt und untersucht wurde. Unter der Bedingung, dass die falsche Erkennung von Zylindern durch Nachverarbeitungsschritte und Suche nach Zusammenhangskomponenten deutlich gesenkt werden kann, könnte nach der Zylindersegmentierung die Ebenensegmentierung auf der unsegmentierten Punktmenge ausgeführt werden.

Die kombinierte Suche nach mehreren Formprimitiven wurde in dieser Arbeit nicht gelöst. Ebenen und Zylinder können ausschließlich getrennt voneinander detektiert

werden. Die kombinierte Suche, wie in [14], wäre ein möglicher nächster Arbeitsschrit. Zusätzliche Erweiterungen des Verfahrens ließen sich durch Ergänzung weiterer Formprimitive, wie zum Beispiel Kugeln oder Tori, schaffen.

Die Datenkompression wurde in dieser Arbeit bisher wenig untersucht. Es könnten zum einen die Kompression der Texturen wie bei [13] und ihre Auswirkungen auf die benötigte Bandbreite zur Übertragung der Daten untersucht werden. Zum anderen könnten Beschränkungen der minimalen Auflösung des Oktalbaumes oder unterschiedliche Methoden zur Formprimitiv erhaltenden Verkleinerung der Punktwolke geprüft werden.

Als weitere Möglichkeit zur Fortsetzung der Arbeiten an diesem Verfahren ist die Berechnung der Fehlerfortpflanzung für die Hough-Transformation, ähnlich wie bei Shapiro et al. [15] [16] und [17]. Statt alle benachbarten Orientierungen des Orientierungshistogramms, ausgewählt nache inem festgelegten Radius, mit Gewichten zu versehen, könnte die Auswahl statt dessen durch eine Fehlerellipse gegeben werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] C. Beder and W. Förstner. Direct solutions for computing cylinders from minimal sets of 3d points. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proceedings of the European Conference on Computer Vision*, number 3951 in LNCS, pages 135–146. Springer, 2006.

[3] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981.

[4] P. F. U. Gotardo, O. R. P. Bellon, and L. Silva. Range image segmentation by surface extraction using an improved robust estimator. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II – 33–8, 2003.

[5] A. Harati, S. Gächter, and R. Siegwart. Fast range image segmentation for indoor 3d-slam. *6th IFAC Symposium on Intelligent Autonomous Vehicles*, 2007.

[6] A. Hoover, G. Jean-Baptiste, X. Jiang, P.J. Flynn, H. Bunke, D.B. Goldgof, K. Bowyer, D.W. Eggert, A. Fitzgibbon, and R.B. Fisher. An experimental comparison of range image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):673 –689, 1996.

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26:71–78, July 1992.

[8] P. Hough. Method and means for recognizing complex patterns. U.S. Patent 3.069.654, December 1962.

[9] X. Jiang and H. Bunke. Fast segmentation of range images into planar regions by scan line grouping. *Mach. Vision Appl.*, 7:115–122, April 1994.

[10] B. Oehler, J. Stückler, J. Welle, D. Schulz, and S. Behnke. Efficient multi-resolution plane segmentation of 3d point clouds. In *4th International Conference on Intelligent Robotics and Applications*, to appear December 2011.

[11] T. Rabbani. *Automatic reconstruction of industrial installations using point clouds and images*. PhD thesis, TU Delft, 2006.

[12] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, August 2006.

[13] R. Schnabel. *Efficient point-cloud processing with primitive shapes*. Dissertation, Universität Bonn, 2010.

[14] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[15] S. D. Shapiro. Transformations for the computer detection of curves in noisy pictures. *Computer Graphics and Image Processing*, 4(4):328–338, 1975.

[16] S. D. Shapiro. Feature space transforms for curve detection. *Pattern Recognition*, 10(3):129–143, 1978. The Proceedings of the IEEE Computer Society Conference.

[17] S. D. Shapiro. Properties of transforms for the detection of curves in noisy pictures. *Computer Graphics and Image Processing*, 8(2):219–236, 1978.

[18] R. W. Taylor, M. Savini, and A. P. Reeves. Fast segmentation of range imagery into planar regions. *Computer Vision, Graphics, and Image Processing*, 45(1):42 – 60, 1989.

[19] F. van den Heuvel and T. Rabbani. Efficient hough transform for automatic detection of cylinders in point clouds. In *Proceedings of the ISPRS Workshop Laser scanning 2005*, volume XXXVI, pages 60 – 65, 2005.

[20] G. Vosselman, B. G. H. Gorte, G. Sithole, and T. Rabbani. Recognising structure in laser scanner point clouds. In *ISPRS - Laser-Scanners for Forest and Landscape Assessment*, volume XXXVI - 8/W2, pages 33 – 38, 2004.