Institut für Informatik VII
Robotik und Telematik

Bachelorarbeit

Finden von Ebenen in 3D-Punktwolken mithilfe von RANSAC und Hough-Transformation

Tim Herrmann

April 2017

Erstgutachter: Prof. Dr. Andreas Nüchter

Zusammenfassung

Die vorliegende Arbeit behandelt das Erkennen von Ebenen in 3D-Punktwolken. Dabei wird der RANSAC-Algorithmus und verschiedene Arten der Hough-Transformation betrachtet und anhand der Qualität der Resultate und den zugehörigen Laufzeiten verglichen. Hierzu finden Implementierungen aus dem *3DTK - The 3D-Toolkit* Verwendung, welches verschiedene Methoden zum Arbeiten mit Punktwolken bereitstellt. Des Weiteren werden noch Datensätze benötigt, auf denen die Verfahren getestet werden können. Dazu nimmt ein Scanner fünf verschiedene Datensätze innerhalb der Universität auf. Durch die Anwendung der verschiedenen Algorithmen auf den Daten entstehen schließlich Testergebnisse, die deutlich machen welche Verfahren in der Praxis zum Finden von Ebenen in Punktwolken geeignet sind, und welche nicht.

Inhaltsverzeichnis

1 Einführung **1**
1.1 wissenschaftliche Arbeiten und Zielsetzung 1
1.2 Aufbau der Arbeit 2

2 Theorie **3**
2.1 RANSAC 3
2.1.1 Funktionsweise 3
2.1.2 Octtree 4
2.2 Hough-Transformation 5
2.2.1 Funktionsweise 6
2.2.2 Standard Hough-Transformation (SHT) 7
2.2.3 Probabilistische Hough-Transformation (PHT) 7
2.2.4 Adaptive Probabilistische Hough-Transformation (APHT) 7
2.2.5 Progressive Probabilistische Hough-Transformation (PPHT) 8
2.2.6 Randomisierte Hough-Transformation (RHT) 8

3 Vorgehensweise **9**
3.1 Vorbereitungen 9
3.2 Arbeiten am RANSAC-Algorithmus 9
3.2.1 Ebene einschränken und vorbereiten 11
3.2.2 Mehrere Ebenen finden 11
3.3 Testen der Verfahren auf den Beispieldatensätzen 13

4 Ergebnisse **15**
4.1 Standard Hough-Transformation 15
4.2 Probabilistische Hough-Transformation 17
4.3 Adaptive Probabilistische Hough-Transformation 17
4.4 Progressive Probabilistische Hough-Transformation 17
4.5 Randomisierte Hough-Transformation 20
4.6 RANSAC 20
4.7 Praktische Erfahrungen mit den Verfahren 21

5 Fazit **25**
5.1 Anhang zur Geschwindigkeit von RANSAC 25

Kapitel 1

Einführung

Zum Erkunden von schwer- oder unzugänglichen Terrain wird häufig ein Laser Scanning System genutzt. Hierbei misst ein Scanner die Distanz zu anderen Objekten mittels Laserstrahlen und speichert sie als 3D-Punkte ab. Auf diese Weise entstehen sogenannte Punktwolken, welche die ganzen gefundenen Punkte enthalten. Wird nun ein solcher Scanner an einem Flugobjekt wie zum Beispiel einem Flugzeug oder Helikopter befestigt, kann in kurzer Zeit eine große Fläche analysiert werden. Dieses Verfahren nennt sich Airbone Laserscanning und eignet sich für das Erstellen von dichten 3D-Punktwolken [4]. Um das Gescannte besser analysieren zu können, können die Daten vorab noch vorbereitet werden. So ist zum Beispiel eine Klassifizierung von Bereichen sehr hilfreich. Bei dem Scan eines Waldes kann beispielsweise zwischen dem Terrain und der Pflanzenwelt unterschieden werden [4]. Wird eine Stadt analysiert könnten die einzelnen Häuser oder Straßen von Bedeutung sein. Häufig ist dafür das Finden von primitiveren Formen notwendig. Häußerdächer und -wände ähneln häufig Ebenen, die zu finden sind. Obwohl auch andere Formen wie der Zylinder eine wichtige Rolle spielen können, beschäftigt sich diese Arbeit ausschließlich mit dem Erkennen von Ebenen in 3D-Punktwolken.

1.1 wissenschaftliche Arbeiten und Zielsetzung

In der Veröffentlichung *Hough-Transform and Extended RANSAC Algorithms for Automatic Detection of 3D Building Roof Planes from Lidar Data* wird die Hough-Transformation mit dem RANSAC-Algorithmus verglichen. Hierbei steht jedoch nicht das Finden von Ebenen generell, sondern das Erkennen von Häuserdächern in Punktwolken im Vordergrund. Nach den Tests der jeweils ersten Implementierung stellte sich RANSAC als schneller und genauer heraus und infolgedessen rückten Verbesserungen dieses Verfahrens in den Vordergrund.

3DTK - The 3D-Toolkit [3] bietet verschiedene Algorithmen rund um 3D-Punktwolken. Es ermöglicht unter anderem das Darstellen und automatisierte Verknüpfen verschiedener Scans und enthält bereits verschiedene Methoden zum Ausmachen von Ebenen.

Ziel dieser Arbeit ist es die verschiedenen Verfahren der Hough-Transformation und RANSAC

zum Finden von Ebenen zu beurteilen und vergleichen. Im Gegensatz zu der bereits genannten Veröffentlichung stehen dabei jedoch nicht nur Häuserdächer im Fokus. Außerdem werden dabei auch von der Standard Hough-Transformation abgeleitete Methoden in Betracht gezogen. Änderungen und Erweiterungen der dort bereits implementierten Algorithmen sollen nach dem Abschluss der Arbeit, sofern es sinnvoll ist, in das 3D-Toolkit mit aufgenommen werden.

1.2 Aufbau der Arbeit

Zunächst sollen Grundlagen erläutert werden, die für das Verständnis dieser Arbeit notwendig sind. Hierbei sind natürlich der RANSAC-Algorithmus und die Hough-Transformation von größter Bedeutung. Anschließend wird, bei den Vorbereitungen beginnend, das Vorgehen beschrieben. Dabei wird deutlich, welche Arbeiten im 3D-Toolkit noch zu erledigen waren und wie es zu den Testergebnissen kam. Diese werden danach analysiert, verglichen und beurteilt, damit zum Schluss ein Fazit gezogen werden kann.

Kapitel 2

Theorie

2.1 RANSAC

Grundsätzlich erstellt der RANdom SAmple Consensus Algorithmus (RANSAC) anhand von zufällig gewählten Stichproben eine Art Modell, welches anschließend geprüft und bewertet wird. Dieses Verfahren wird viele Male wiederholt, wobei das beste Modell gespeichert wird.

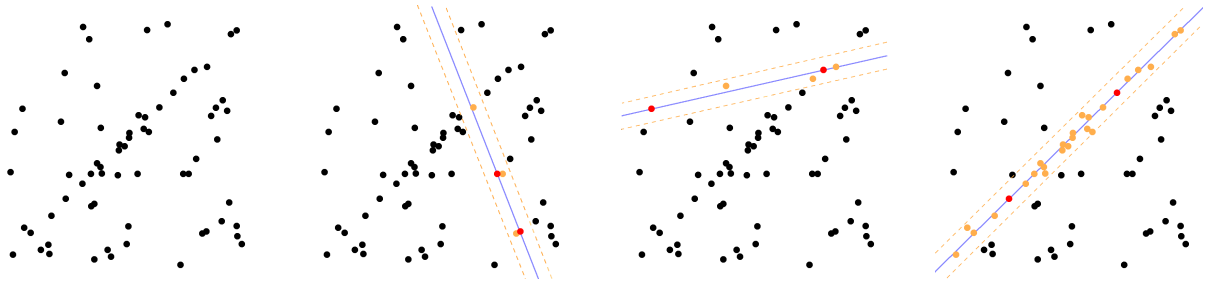
Um dem Leser das Eintauchen in die Theorie leichter zu machen, wird das Verfahren vorerst vereinfacht im Zweidimensionalen erklärt. Anstatt Ebenen, die die meisten Punkte enthalten, sollen nun dementsprechend Linien gefunden werden.

2.1.1 Funktionsweise

Um die beste Linie zu finden werden zufällig zwei Punkte ausgewählt und die Gerade berechnet, die beide verbindet. Anschließend werden die Distanzen aller anderen Punkte zur Linie berechnet. Ist sie kleiner als ein gewisse vorher festgelegte Grenze, so ist der entsprechende Punkt nahe genug an der Linie, passt zum Modell und zählt als sogenannter Inliner. Ansonsten ist er ein Outliner. Viele Iterationen und das Abspeichern der Linie mit den meisten Inlinern ermöglicht dann das Finden einer guten Linie (siehe Abbildung 2.1). Je mehr Iterationen es gibt, desto wahrscheinlicher ist es, die beste zu finden, folglich steigt aber natürlich auch die Berechnungszeit an.

Die Anzahl der Wiederholungen ist frei wählbar und kann auch mit Bedingungen verknüpft werden. So ist es beispielsweise denkbar abzubrechen, wenn entweder 50% der Punkte auf der Linie liegen oder 3000 Iterationen durchlaufen wurden. Zum Schluss wird die Linie noch einmal angepasst, indem für ausschließlich die Inliner die beste Linie berechnet wird.

Dieses Prinzip lässt sich leicht auf die 3D-Welt übertragen. Anstatt zwei zufälligen Punkten werden jetzt drei benötigt, um eine Ebene aufstellen zu können. Ansonsten ändert sich nur



(a) Beispieldatensatz **(b)** Erste Iteration **(c)** Zweite Iteration **(d)** Dritte Iteration

Abbildung 2.1: Beispiel einiger RANSAC-Iterationen. Die zufällig gewählten Punkte sind rot und Inliner orange gekennzeichnet.

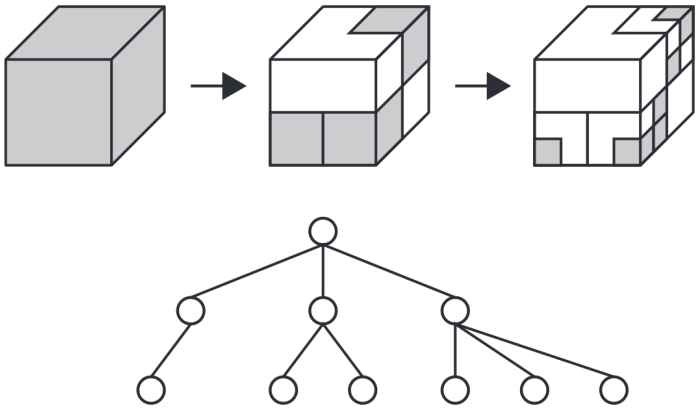


Abbildung 2.2: Struktur eines Octtrees [2]

der mathematische Teil. Die Abstandsberechnung von Punkten zu Ebenen ist dementsprechend etwas anders als zu Linien, aber das grundsätzliche Vorgehen bleibt unverändert.

2.1.2 Octtree

Der größte Zeitaufwand des Algorithmus pro Iteration ist das Auswerten des Modells. Schließlich muss dafür jeder Punkt, der nicht zu den drei zufällig gewählten gehört, in In- und Outliner kategorisiert werden. Um hier die Laufzeit zu verbessern, werden die Punkte deshalb in einem zuvor erstellen Octtree gespeichert. Alle für diesen Abschnitt nowendige Informationen sind in *One billion points in the cloud – an octree for efficient processing of 3D laser scans* [2] zu finden.

Aufbau

Bei einem Octtree handelt es sich um eine Baumstruktur, die sich sehr gut für das effektive Speichern von 3D-Punktwolken eignet. Jeder Knoten des Octtrees gleicht dem Volumen eines

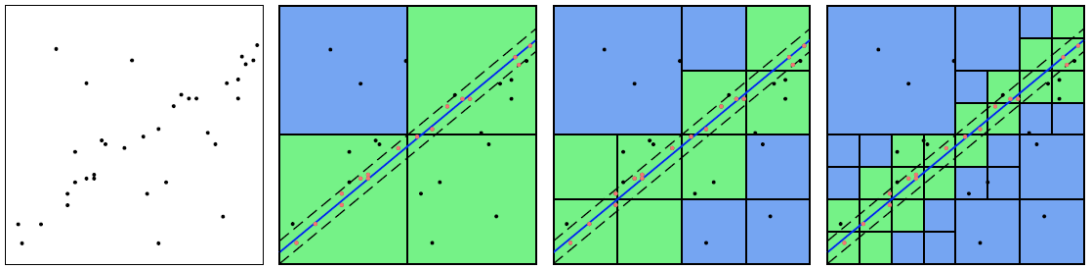


Abbildung 2.3: Verschnellerte Auswertung des Modells

Würfels, der meist wie die Achsen des Koordinatensystems orientiert ist. Jeder Knoten wird rekursiv in acht Kinderknoten aufgeteilt, wobei jedes Kindknoten einen Würfel innerhalb des Elternknoten mit genau einem Achtel seiner Größe repräsentiert. Ein Kind existiert allerdings nur dann, wenn mindestens ein Punkt in dem repräsentierenden Würfel eingeschlossen ist (siehe Abbildung 2.2). Somit hat ein Knoten häufig weniger, jedoch maximal acht Kinder, und unnötige Rekursionen im leeren Raum werden vermieden.

Da die einzelnen 3D-Punkte dimensionslos sind, würde der Baum, wenn keine Gegenmaßnahmen ergriffen werden unendlich in die Tiefe wachsen. Um dies zu verhindern wird eine maximale Baumtiefe und eine minimale Knotenanzahl definiert. Wenn also die maximale Baumtiefe überschritten wird, werden die Knoten ab dieser Tiefe zu Blättern. Diese beinhalten dann letztendlich eine Liste der Punkte, die sich im repräsentierten Raum befinden. Falls die Anzahl der Punkte in den vom Knoten dargestellten Würfel kleiner wird als die minimale Knotenanzahl wird der jeweilige Knoten auch zu einem Blatt.

Ersparnisse mithilfe des Octtrees

Bei einer unvorbereiteten unsortieren 3D-Punktwolke muss für die Evaluation des Modells über jeden Punkt der Wolke iteriert werden. Durch die Verwendung des Octtrees kann dieser Prozess deutlich verschnellert werden. Dafür wird das aufgestellte Modell rekursiv mit den Knoten des Octtrees geschnitten. Ist die Schnittmenge leer, so ist kein Punkt innerhalb dieses Knotens Teil des Modells und die Rekursion wird hier gestoppt. Auf diese Art und Weise können viele Punkte gleichzeitig als Outliner definiert werden ohne sie genauer betrachten zu müssen. In Abbildung 2.3 wird dieses Konzept für das Finden von Linien im zweidimensionalem Raum aufgezeigt.

2.2 Hough-Transformation

Die Hough-Transformation wird üblicherweise zum Erkennen von Linien und Kreisen genutzt, kann aber auch für andere Formen verwendet werden, wie in unserem Fall die Ebene. Obwohl der Input meist aus Pixeln eines Bildes besteht, kann dabei problemlos mit 3D-Punktwolken gearbeitet werden. Die nun folgende Darstellung orientiert sich an der wissenschaftlichen Arbeit

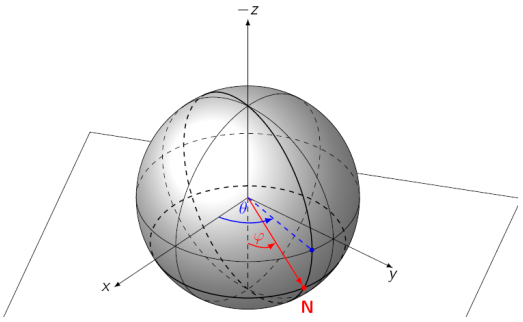


Abbildung 2.4: Normalenvektor dargestellt in Polarkoordinaten [1]

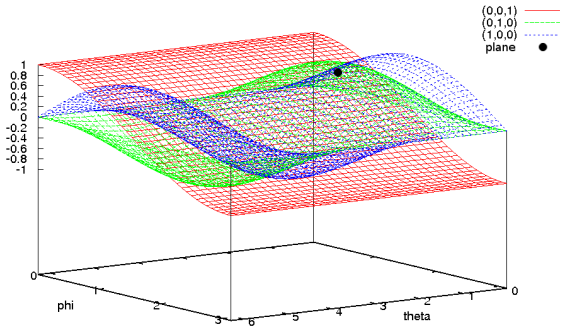


Abbildung 2.5: Hough-Raum nach Tranformation von drei Punkten. Der Schnittpunkt der Sinuskurven ist schwarz markiert. [1]

The 3D Hough Transform fo Plane Detection in Point Clouds: A Review and a new Accumulator Design [1].

2.2.1 Funktionsweise

Eine Ebene kann auf verschiedene Arten dargestellt werden. In der Hesse-Normalform wird eine Ebene durch einen Punkt p auf der Ebene, dem Normalenvektor n und dem Abstand ρ zum Ursprung definiert.

$$\rho = p \cdot n = p_x n_x + p_y n_y + p_z n_z$$

Werden nun die Winkel zwischen dem Normalenvektor und des Koordinatensystems betrachtet, können die Koordinaten von n folgendermaßen faktorisiert werden:

$$\rho = p_x \cdot \cos\theta \cdot \sin\varphi + p_y \cdot \sin\varphi \cdot \sin\theta + p_z \cdot \cos\varphi \tag{2.1}$$

Bei θ handelt es sich um den Winkel des Normalenvektors auf der xy-Achse und bei φ um den Winkel zwischen der xy-Ebene und dem Normalenvektor in z-Richtung, wie es in Abbilung 2.4 zu sehen ist. Somit definieren φ, θ und ρ den sogenannten Hough-Raum (θ, φ, ρ), sodass jeder Punkt im Hough-Raum eine Ebene im $\mathbb{R}^3$ darstellt.

Um schließlich Ebenen in der 3D-Point-Cloud zu finden wird für jeden Punkt die Hough-Transformation berechnet. Für einen Punkt p müssen dafür alle Ebenen gefunden werden, die p beinhalten, dass heißt alle Kombinationen von θ,φ und ρ, sodass die Gleichung 2.1 erfüllt ist. Diese werden in den Hough-Raum übertragen, wodurch dort Sinuskurven entstehen. Falls also drei Punkte transformiert werden, entstehen dementsprechend drei Sinuskurven die sich schneiden. Der Schnittpunkt definiert die Ebene die die drei Punkte aufspannen (siehe Abbildung 2.5). Je mehr Kurven sich im Hough-Raum in einem Punkt schneiden, desto mehr Punkte liegen auf der Ebene die durch den Schnittpunkt repräsentiert wird.

2.2.2 Standard Hough-Transformation (SHT)

Der Hough Space wird diskretisiert. Dabei bestimmen θ', φ' und ρ' die Größe der entstehenden Zellen in der entsprechenden Richtung im Hough Space. Jede dieser Zelle speichert einen Score. Für jeden Punkt der Wolke wird der Score der Zellen, die von der Hough Transformation berührt werden, um eins erhöht. Dieser Prozess wird häufig als "voting"(Abstimmung, Wahl) bezeichnet, das heißt jeder Punkt wählt alle Kombinationen der Parameter (θ, φ, ρ), die eine Ebene definieren auf die er liegt. Ein Punkt liegt dabei genau dann auf der Ebene, wenn der Abstand zwischen Punkt und Ebene kleiner ist als die vorher festgelegte Grenze T.

Die Zellen die am Schluss den höchsten Wert besitzen repräsentieren die Ebenen, welche die meisten Punkte einschließen. Beim Auswählen der besten Ebenen ist es aufgrund der Diskretisierung und Störanfälligkeiten in der Punktwolke jedoch sinnvoll, nicht nach der Zelle mit dem höchsten Score, sondern nach der maximalen Summe der Scores über einen kleinen zusammenhängenden Bereich zu suchen.

2.2.3 Probabilistische Hough-Transformation (PHT)

Die Standard Hough-Transformationen kann in zwei Schritte eingeteilt werden. Im ersten werden die Punkte der 3D-Punktwolke P in den Hough-Raum transformiert und die Scores der entsprechenden Zellen werden erhöht. Dies benötigt $O(|P| \cdot N_\varphi \cdot N_\theta)$ Operationen, wobei N_φ die Anzahl der Zellen in φ-Richtung, N_θ die Anzahl in θ-Richtung und N_ρ die Anzahl in ρ-Richtung entspricht. Die anschließende Suche nach den maximalen Scores benötigt $O(N_\rho \cdot N_\varphi \cdot N_\theta)$ Operationen. Da die Größe der Wolke $|P|$ für gewöhnlich deutlich größer ist als die Anzahl der Zellen $N_\rho \cdot N_\varphi \cdot N_\theta$ können größere Laufzeitverbesserungen durch eine Verringerung der Punkte, statt einer Anpassung der Diskretisierung erreicht werden.

Deshalb werden bei der Probabilistischen Hough-Transformation zufällig $m < |P|$ Punkte ausgewählt und für den Algorithmus verwendet. Dementsprechend wird dadurch die hohe Laufzeit des ersten Schrittes auf $O(m \cdot N_\varphi \cdot N_\theta)$ verringert. Um hierbei ähnliche gute Ergebnisse zu erzielen muss m und T gut gewählt sein. Die optimale Lösung hängt jedoch immer von dem verwendeten Datensatz ab.

2.2.4 Adaptive Probabilistische Hough-Transformation (APHT)

Häufig werden bei der Probabilistischen Hough-Transformation immer noch deutlich mehr Punkte verwendet als notwendig, um Fehler zu vermeiden. Aus diesem Grund wurden weitere Methoden entwickelt. Bei der Adaptiven Probabilistischen Hough-Transformation werden die einzelnen Zellen beobachtet. Sobald sich stabile Strukturen herauskristallisieren und sich relevante Maxima bilden wird das transformieren der Punkte gestoppt. Außerdem müssen dabei nur die Zellen, deren Score verändert wurde, untersucht werden und die Zelle mit dem größten Score wird als Ebene in Betracht gezogen. Falls mehrere Zellen den gleichen Score haben wird eine davon ausgewählt. Zusätzlich wird eine sortierte Liste der potentiellen Maxima gespeichert. Diese wird,

um den Prozess zu verschnellern, nicht nach jedem Punkt, sondern nach einem kleinen Stoß von Punkten aktualisiert. Beim Updaten wird auch die Position der Höchstwerte berücksichtig. Liegen beispielsweise zwei Maxima in näherer Umgebung so wird der kleinere der beiden aus der Liste gelöscht. Der kontinuierliche Vergleich der aufeinanderfolgenden Listen ermöglicht das Überprüfen der Konsistenz der Maxima. In der Anfangsphase des Algorithmus wird es typischerweise viele Änderungen in dieser Liste geben, aber je mehr Punkte transformiert werden, desto klarer werden die gefunden Maxima. Die Abbruchbedingung des Verfahrens hängt von der Stabilität der größten Maxima ab.

2.2.5 Progressive Probabilistische Hough-Transformation (PPHT)

Hier wird nach der Transformation jedes Punktes überprüft ob das Verhältnis zwischen dem Score der besten Zelle und der Anzahl der bisher transformierten Punkten einen gewissen Grenzwert überschreitet. Tritt dieser Fall ein, so wird unter allen Punkten, welche auf der Ebene dieses Maxima liegen, die größte zusammenhängende Region gesucht. Die Punkte der Region P_{Region} werden nun aus der Punktwolke P entfernt. Wenn die Anzahl der Punkte in der Region größer ist als ein vorher festgelegte Mindestgröße dann werden sie zudem noch zu der Ausgabe hinzugefügt. Anschließend wird die Transformation aller Punkte in P_{Region} rückgängig gemacht.

2.2.6 Randomisierte Hough-Transformation (RHT)

Anstatt in jedem Schritt zufällig einen Punkt zu wählen und diesen zu Transformieren, werden jetzt immer gleich drei betrachtet. Dadurch kann direkt die Ebene, die durch sie aufgespannt wird, berechnet werden. Im Hough-Raum wird nun der Score der Zelle, der diese Ebene repräsentiert, um eins erhöht. Wenn sich in der Pointwolke an der Stelle (θ, φ, ρ) eine Ebene befindet, so wird es nach einer gewissen Zeit ein großen Score in dieser Zelle geben. Demnach werden sich die Zellen, die eine Ebene in der Wolke darstellen, im Laufe der Zeit von den anderen abheben. Falls der Score einer Zelle einen festgelegten Grenzwert überschreitet, wird die Ebene parametrisiert und zur Ausgabe hinzugefügt. Anschließend müssen noch die Punkte die nahe an der Ebene liegen aus 3D-Point-Cloud ausgeschlossen und der Hough-Raum zurückgesetzt werden. Da Punkte, die sehr weit voneinander entfernt sind, höchst wahrscheinlich nicht auf einer Ebene liegen, wird noch ein Kriterium für die Distanz zwischen den drei ausgewählten Punkten eingeführt. Demnach werden nach dem Wählen der Punkte zuerst die zugehörigen Abstände berechnet. Ist einer davon zu groß oder zu klein werden zufällig neue Punkte bestimmt.

Kapitel 3

Vorgehensweise

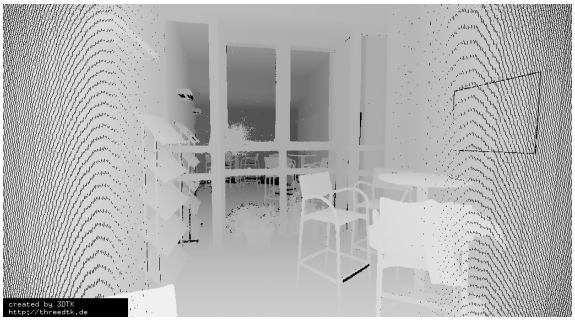
3.1 Vorbereitungen

Zuerst wurde „3DTK - The 3D Toolkit“ von `http://slam6d.sourceforge.net/` heruntergeladen und installiert. Es ist für jeden zugänglich und stellt Algorithmen zum Arbeiten mit 3D-Punktwolken bereit. Hier sind unter anderem bereits die verschiedenen Methoden zur Hough-Transformation implementiert. Auch der Octtree und eine unfertige Version des RANSAC-Algorithmus sind vorzufinden. Des Weiteren bietet es eine Methode zum Anzeigen der Daten in dem auch die gefundenen Ebenen mit einbezogen werden können, weshalb sich das 3D Toolkit sehr gut für diese Arbeit eignet.

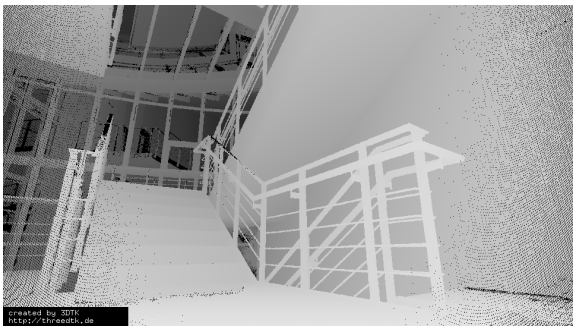
Anschließend erstellt ein Scanner innerhalb der Universität Würzburg eigene Datensätze, auf denen später die verschiedenen Methoden zum Finden von Ebenen ausgeführt und verglichen werden sollen. So entstehen insgesamt fünf Scans, die in Abbildung 3.1 zu sehen sind.

3.2 Arbeiten am RANSAC-Algorithmus

Da die Methoden zur Hough-Transformation bereits vollständig implementiert sind, bleibt nur noch das Fertigstellen des RANSAC-Algorithmus. Bisher führt er das Verfahren wie in Kapitel 2.1 beschrieben unter Verwendung eines Octtrees genau einmal mit 5000 Iterationen aus. Die gefundene Ebene wird dabei allerdings noch in keinster Weise eingegrenzt oder für den Datensatz vorbereitet und kann dementsprechend nicht mit angezeigt werden.

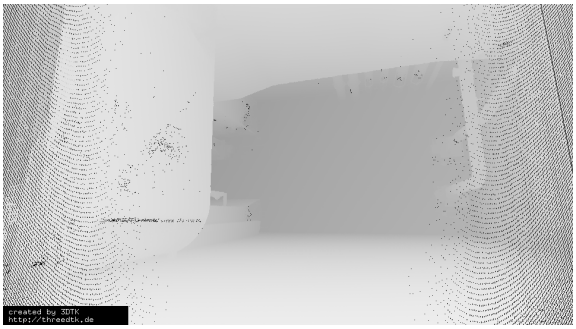


(a) kleiner Aufenthaltsbereich

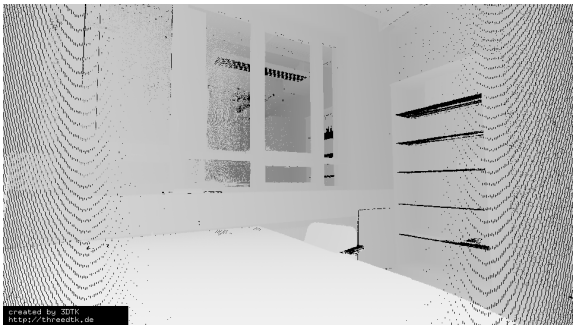


(b) im Gang vor einer Treppe

(c) Aufenhaltsbereich der Studenten



(d) leerstehender Keller



(e) kleines Büro

Abbildung 3.1: Einblicke in die fünf aufgenommen Datensätze

3.2.1 Ebene einschränken und vorbereiten

Um es zu ermöglichen die gefundene Ebene mit den Punkten zusammen darzustellen muss sie begrenzt und abgespeichert werden. Dazu wird die größte zusammenhängende Bereich an Punkten gesucht, die auf der Ebene liegen. Danach kann für diese Region eine konvexe Hülle berechnet werden, die dazu dient die Ebene einzuschränken. Zum Schluss wird das Ergebnis noch mithilfe von Text-Dateien festgehalten.

Alle notwendigen Funktionen für dieses Vorgehen sind bereits für die Hough-Transformation implementiert und können auch für den RANSAC-Algorithmus genutzt werden. Dabei ist lediglich das Verwenden der richtigen Vorzeichen zu beachten. In Abbildung 3.2 ist der Datensatz des leerstehenden Kellers mit der gefundenen Ebene zu sehen.

3.2.2 Mehrere Ebenen finden

Falls nun mehrere Ebenen gesucht sind, hilft der momentane Algorithmus jedoch nicht. Bei mehrfachem Ausführen auf den gleichen Daten wird wahrscheinlich meistens die gleiche Ebene berechnet. Wie viele verschiedene Ebenen auf diese Weise entdeckt werden hängt natürlich von dem jeweiligen Datensatz ab. Fakt ist allerdings, dass es in der Hinsicht noch viel Potential gibt.

Um zu verhindern die gleiche Ebene mehrmals zu finden müssen alle Punkte, die auf bereits gefundenen Ebenen liegen, ignoriert werden. Dazu erstellt der Algorithmus nach jedem vollständigen Durchgang eine neue Scan-Datei, die nur die Punkte enthält die nicht auf der soeben berechneten Ebene liegen. So wird sichergestellt, dass bei der nächsten Wiederholung des Verfahrens eine neue Ebene gefunden wird. Die dabei entstehenden aufeinanderfolgenden Scan-Dateien sind in Abbildung 3.3 veranschaulicht.

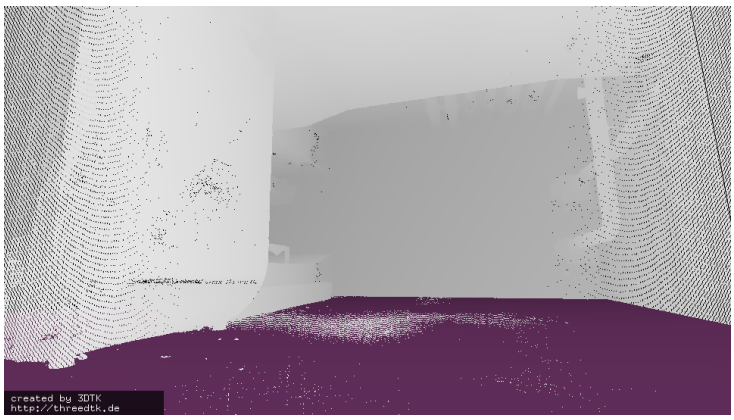
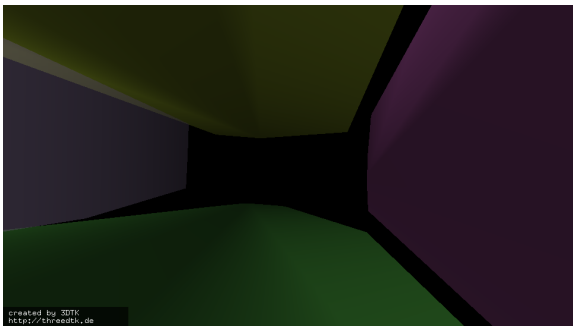


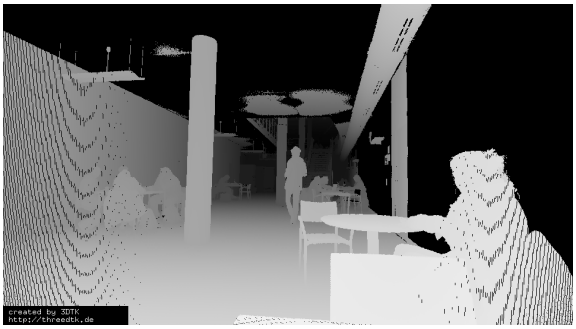
Abbildung 3.2: leerstehender Keller mit lila markierter Ebene

(a) Aufenthaltsbereich für Studenten



(b) gefundene Ebenen

(c) Scan nach dem ersten Durchgang



(d) Scan nach dem zweiten Durchgang

(e) Scan nach dem dritten Durchgang

(f) Scan nach dem vierten Durchgang

Abbildung 3.3: Verschiedene Scandateien während der Suche nach Ebenen

Verbesserung beim Entfernen der Punkte

Bisher wird durch dem Eingabescan im RANSAC ein Octtree erstellt und anschließend damit eine Ebene gesucht. Danach wird eine neue Scan-Datei ohne die Punkte der Ebene erstellt. Demnach muss für das Erkennen der nächsten Ebene wieder ein neuer Octtree mit den Daten des neuen Scans erzeugt werden. Dieses scheinbar umständliche Verfahren sorgte dafür, dass an der Implementierung des Octtrees nichts verändert werden musste und war daher einfacher und dienten eher als Zwischenschritt.

Letztendlich können die unnötigen Berechnungen jedoch verkürzt werden, wenn dem Algorithmus ein Octtree und nicht ein Scan als Eingabe mitgegeben wird. Jedes Mal wenn RANSAC eine Ebene findet, werden die Punkte dann direkt aus dem Octtree herausgelöscht und ein mehrfaches Erstellen von Scan-Dateien und Octtrees wird vermieden. Die dadurch entstehende Zeitersparnis beim Suchen von fünf Ebenen ist in Tabelle 3.1 zu sehen. Je mehr Ebenen gesucht werden, desto größer wird der Unterschied der Berechnungszeit. Abbildung 3.4 zeigt den zeitlichen Verlauf für das Finden von unterschiedlich vielen Ebenen auf dem gleichen Datensatz. Die erste Ebene benötigt im Verhältnis am meisten Zeit da hier der Octtree erst einmal aufgebaut werden muss. Danach kann dieser bearbeitet weiterverwendet werden, weshalb das Finden der anderen Ebenen schneller läuft. Des Weiteren sinkt der Zeitaufwand im Verlauf noch ein wenig, da die Größe des Octtrees nach und nach abnimmt.

3.3 Testen der Verfahren auf den Beispieldatensätzen

Um die Praxisrelevanz der einzelnen Methoden zu beurteilen müssen noch Testergebnisse gesammelt werden. Aus diesem Grund berechnen die verschiedenen Algorithmen jetzt Ebenen zu den einzelnen Datensätzen. Die Ergebnisse werden unter Angabe der Berechnungszeit strukturiert abgespeichert und analysiert. Infolgedessen entstehen für jeden Scan und jedes Vorgehen mehrere Resultate, die im nächsten Kapitel genauer erläutert werden.

Tabelle 3.1: Zeit in Millisekunden für das Finden von fünf Ebenen mit RANSAC bei jeweils 5000 Iterationen mit und ohne der Verbesserung

Datensatz	ohne Verbesserung	mit Verbesserung
kleiner Aufenthaltsbereich	8940	6380
im Gang vor einer Treppe	8801	6407
Aufenthaltsbereich der Studenten	11647	9088
leerstehender Keller	87890	44720
kleines Büro	45119	38444

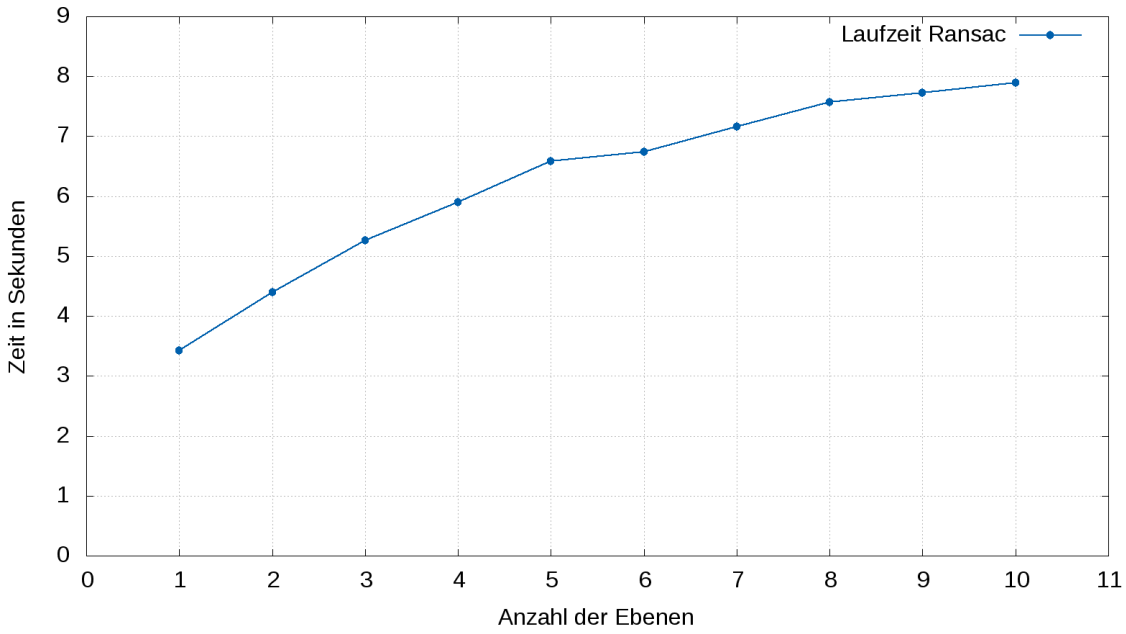


Abbildung 3.4: zeitlicher Verlauf für das Finden von Ebenen im RANSAC

Kapitel 4

Ergebnisse

Im Folgenden sollen die Resultate der verschiedenen Verfahren präsentiert und verglichen werden. Dafür ist natürlich die Geschwindigkeit (siehe Tabelle 4.1) und Qualität ausschlaggebend. Zuerst werden die verschiedenen Hough-Methoden und anschließend der RANSAC-Algorithmus untersucht. Zum Schluss werden dann noch einige Aussagen über ihre Bedienbarkeit in der Praxis getroffen.

4.1 Standard Hough-Transformation

Die Standard Hough-Transformation ist nach der Theorie das langsamste Verfahren. Dies spiegelt sich auch in der Praxis wieder, denn der Zeitaufwand ist hierbei mit Abstand am längsten.

Die Ergebnisse sind trotz dessen nicht immer optimal. Unter Verwendung der richtigen Parameter sind zwar sehr gute Resultate möglich (siehe Abbilung 4.1), jedoch nicht immer leicht erreichbar. In Bild 4.2 wird beispielsweise nicht die linke Wand als vierte große Ebene erkannt, sondern der Tisch und die Lampe, die von deutlich weniger Bedeutung sind.

Tabelle 4.1: Überblick über die Laufzeiten beim Finden von fünf Ebenen mithilfe der verschiedenen Verfahren auf reduzierten Versionen der Datensätze

Datensatz	SHT	PHT	APHT	PPHT	RHT	RANSAC
Aufenthaltsbereich	7484533	1514885	7048877???	618497	439	179
Gang	23063714	4845744	4695419???	4674808	1364	662
Studentenbereich	27796365	5620056	5585208???	4711965	1509	936
Keller	7544622	1523785	???	423662	489	879
Büro	12770472	2579524	27853156???	717134	947	1287

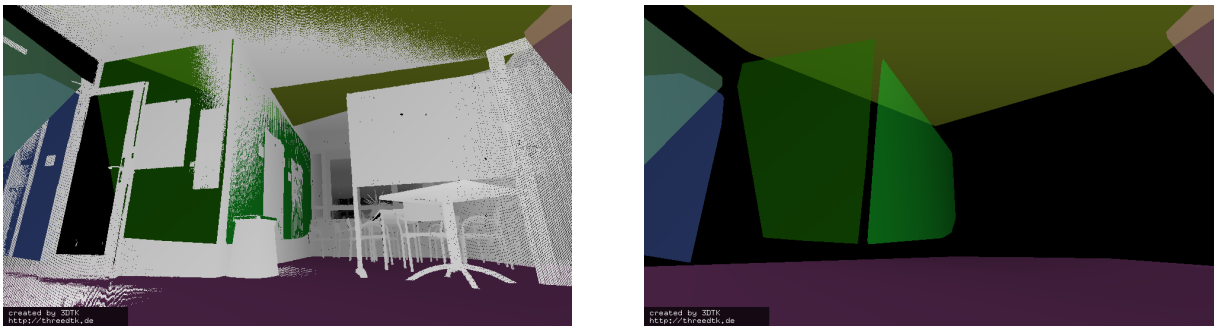


Abbildung 4.1: Gutes Ergebnis mithilfe der Standard Hough-Transformation

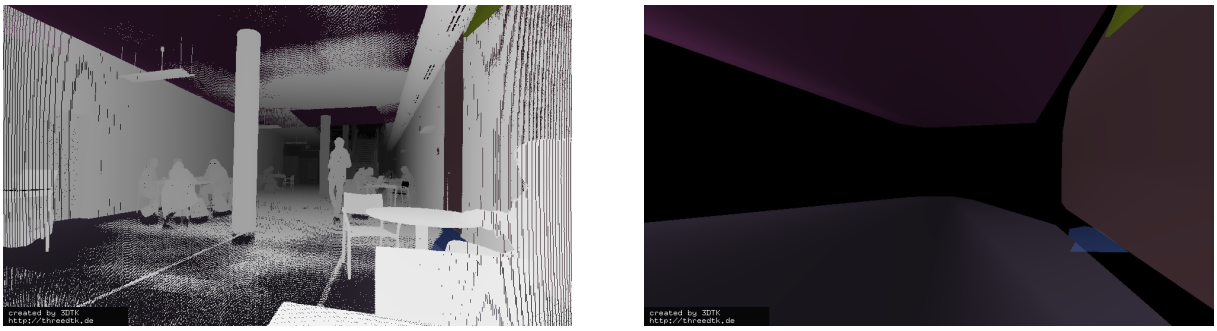


Abbildung 4.2: Standard Hough-Transformation hat vierte große Ebene nicht erkannt

Aufgrund von Störanfälligkeiten wird bei der Standard Hough-Transformation im Hough-Raum nicht nach der Zelle mit dem größten Score, sondern nach einem Bereich mit hohem additiven Score gesucht. Die Wahl für die größe dieses Bereiches ist ausschlaggebend für die Qualität des Ergebnisses und kann demnach leicht zu Problemen führen. Ist der entsprechende Parameter zu groß, können Ebenen ignoriert werden, ist er zu klein werden unter Umständen mehrere nahezu identische Ebenen entdeckt (siehe Abbildung 4.3).

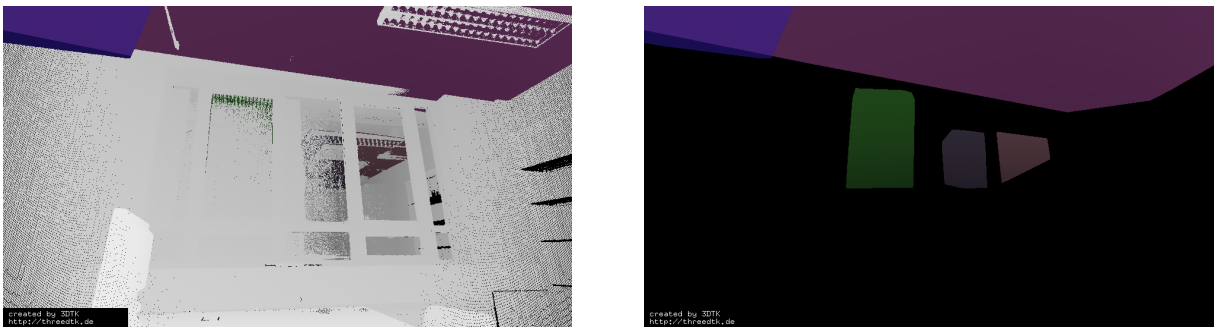


Abbildung 4.3: Viele ähnliche Ebenen an der Decke enteckt

4.2 Probabilistische Hough-Transformation

Die Verwendung von nur $m < |P|$ zufälligen Punkten bringt wie erwartet eine Laufzeitverbesserung mit sich. Bei der Implementierung im 3D-Toolkit treten dabei keine großen Qualitätsverluste der Ergebnisse auf, anderweitige Verbesserungen neben der schnelleren Berechnungsdauer natürlich aber auch nicht. Dementsprechend handelt es sich hierbei um eine schnellere Version der Standard Hough-Transformation. Wie in Abbildung 4.4 zu sehen treten deshalb auch noch die gleichen Probleme auf. Dadurch dass sich die Maxima jetzt eventuell weniger stark vom Rest abzeichnen scheint der Effekt sogar noch etwas verstärkt zu werden.

Über die Wahl von m kann die Qualität und Geschwindigkeit des Verfahrens angepasst werden. So ist es außerdem möglich größere Zeitersparnisse zu erreichen, wenn schlechtere Ergebnisse in Kauf genommen werden.

4.3 Adaptive Probabilistische Hough-Transformation

Dadurch dass eine gewisse Konstanz der Maxima im Hough-Raum für das Erkennen einer Ebene vorausgesetzt wird sind die gefundenen Ebenen bisher stets korrekt. Die Schwierigkeit liegt deshalb eher darin viele Ebenen zu finden, wofür ein gewisses Feingefühl für die verschiedenen Parameter notwendig ist. Gleichermaßen hängt auch die Laufzeit von den eingestellten Werten ab und kann stark variieren. Wenn mehrere Ebenen gesucht werden ist sie mit der der Probabilistischen Hough-Transformation vergleichbar, andernfalls schneller oder langsamer, je nachdem ob nur wenige oder sehr viele Ebenen zu erkennen sind. Die Ergebnisse sind, wie im Bild 4.5 zu sehen, gut sofern die Parameter richtig gewählt sind und demnach ausreichend viele Ebenen gefunden wurden. Ansonsten kann es auch passieren dass nur eine (siehe Abb 4.6) oder gar keine Ebene erkannt wird.

Aufgrund der wachsenden Laufzeit und der notwendigen Erfahrung, falls mehr Ebenen erkannt werden sollen, sollte die Adaptive Probabilistische Hough-Transformation daher mehr für das Erkennen einer oder weniger Ebenen eingesetzt werden.

4.4 Progressive Probabilistische Hough-Transformation

Sowohl die Laufzeit, als auch die Qualität ist im Vergleich zu den Vorgängern deutlich gestiegen. Die Progressive Probabilistische Hough-Transformation liefert konsequent verhältnismäßig schnell gute Ergebnisse (siehe Abbildung 4.7). Auch der durch das viele Glas schwieriger zu händelnde Datensatz des kleinen Aufenthaltraums macht, wie im Bild 4.8 zu sehen, keine Probleme. Allerdings werden nicht immer die größten Ebenen zuerst gefunden, sodass bei einer Suche mit begrenzter Anzahl von Ebenen eventuell eine wichtige Ebene ignoriert wird. In Abbildung 4.9 wird deshalb, wie bereits bei der Standard Hough-Transformation, die vierte große Ebene nicht entdeckt. Aus diesem Grund ist das Verfahren ungeeignet, wenn nur die größte Ebene zu ermitteln ist.

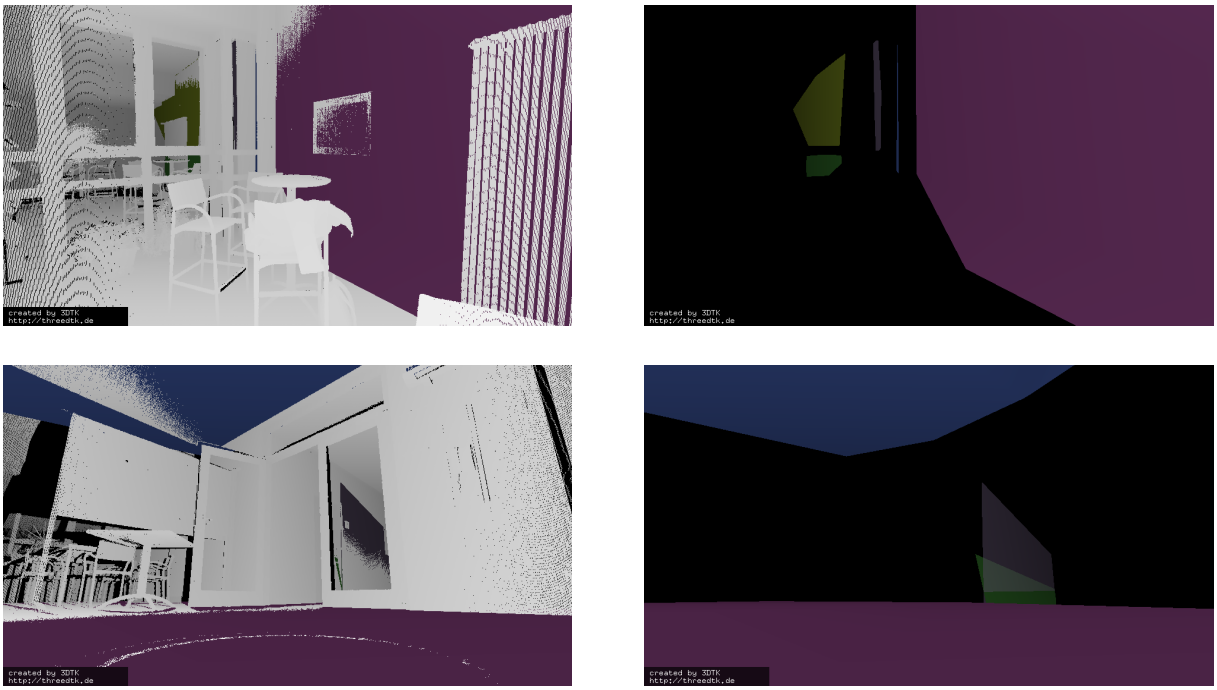


Abbildung 4.4: Ergebnisse der Probabilistischen Hough-Transformation

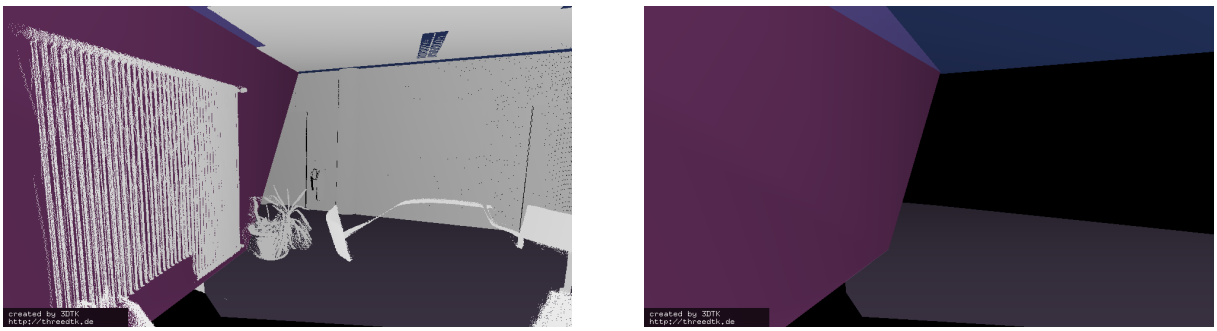


Abbildung 4.5: Mehrere richtige Erbenen erkannt

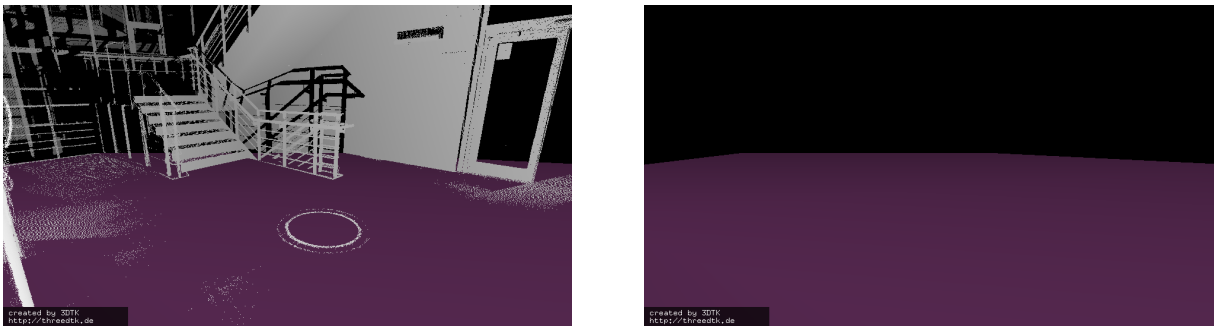


Abbildung 4.6: Obwohl mehr erwünscht und möglich waren wurde nur eine Ebene gefunden

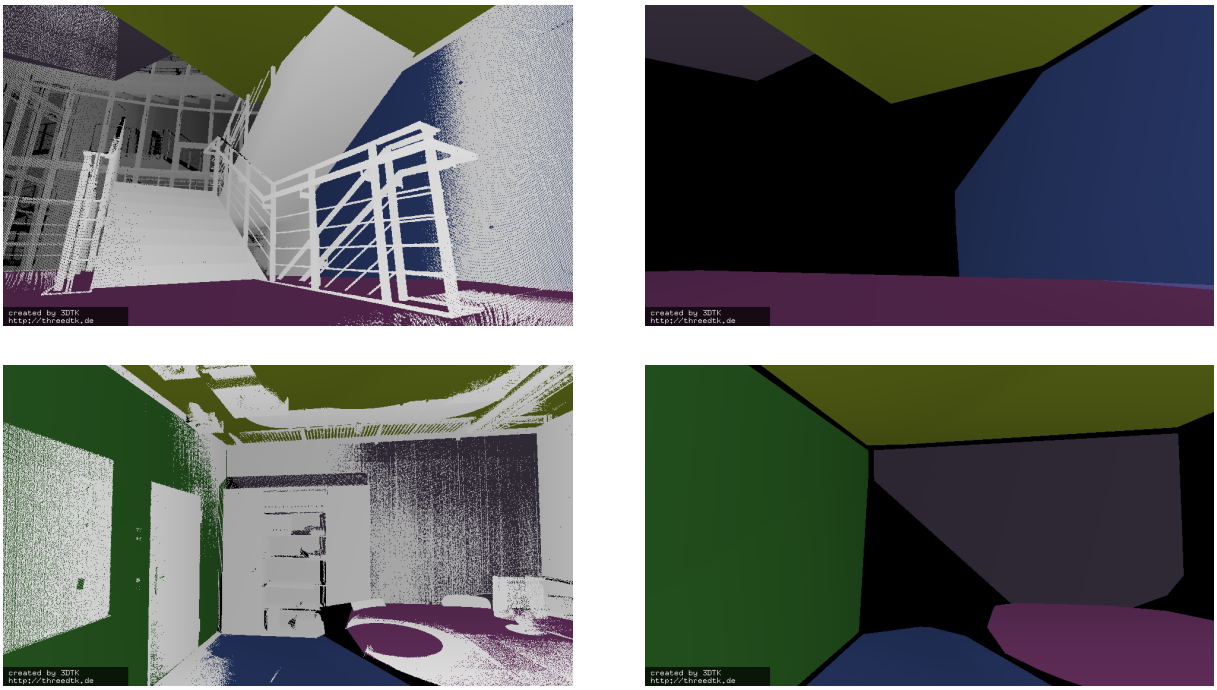


Abbildung 4.7: Gute Ergebnisse mithilfe der Progressiven Probabilistischen Hough-Transformation

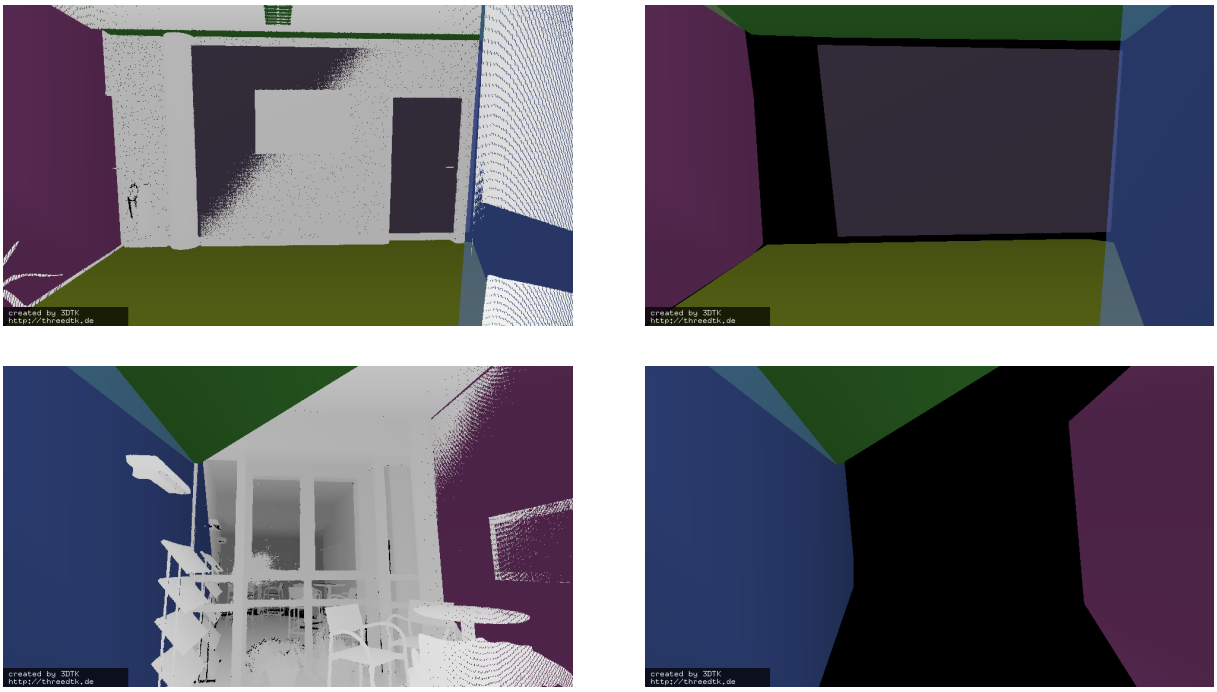


Abbildung 4.8: Ergebnis auf dem Datensatz des kleinen Aufenthaltsraums

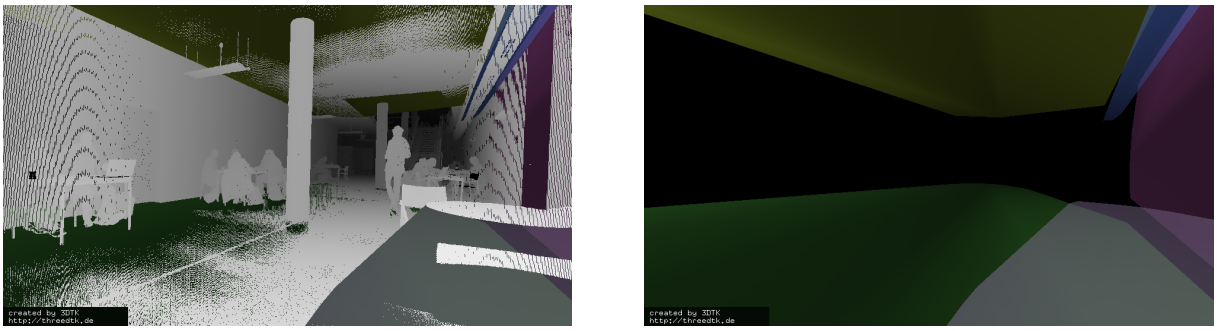


Abbildung 4.9: Erneut die vierte Wand im Aufenthaltsbereich der Studenten nicht erkannt

4.5 Randomisierte Hough-Transformation

Obwohl sich die Randomisierte Hough-Transformation als das mit Abstand schnellste herausstellt sind die Resultate mit denen des bisher besten Verfahrens, der Progressiven Probabilischen Hough-Transformation, vergleichbar. Im Bild 4.10 sind die Ergebnisse zu sehen, die sehr stark an die des letzten Kapitels erinnern. Im Scan des Aufenthaltsbereiches der Studenten fällt jedoch auf, dass jetzt endlich die linke Wand frühzeitig als Ebene erkannt wird (siehe Abbildung 4.11), weshalb das Verfahren nicht nur das bisher schnellste, sondern auch das mit den Resultaten ist.

4.6 RANSAC

Das zweite randomisierte Methode ist, wie auch die erste, deutlich schneller als die anderen Hough-Verfahren. Die Ergebnisse auf den verkleinerten Datensätzen in Tabelle 4.1 lassen darauf schließen dass sie im Durchschnitt etwa genauso schnell ist wie die Randomisierte Hough-Transformation. Werden die Verfahren allerdings auf größeren Scans angewendetn wird klar das RANSAC nicht ganz mit der Geschwindigkeit der RHT mithalten kann (siehe Abbildung 4.12).

Die Ergebnisse sind hingegangen auf gleichem Niveau. Die größten Ebenen werden, mit der größten beginnend, konsequent und fehlerfrei erkannt (siehe Abbildung 4.13). Bei ähnlich großen Ebenen ist aufgrund der Randomisierung nicht eindeutig, welche zuerst erkannt wird. Selbiges gilt aber auch für RHT und PPHT. Wird eine festgelegte Anzahl an Ebenen gesucht, kann die unterschiedliche Reihenfolge dann zu unterschiedlichen Resultaten führen. Im Bild 4.14 wird dieses Verhalten veranschaulicht. Beim ersten Durchlauf auf dem Datensatz des kleinen Büros unterscheidet sich das Ergebnis von dem der Randomisierten Hough-Transformation. Statt der Decke wird jetzt die hintere Wand erkannt. Im nächsten Durchlauf sind die Ergebnisse der beiden Algorithmen wieder identisch.

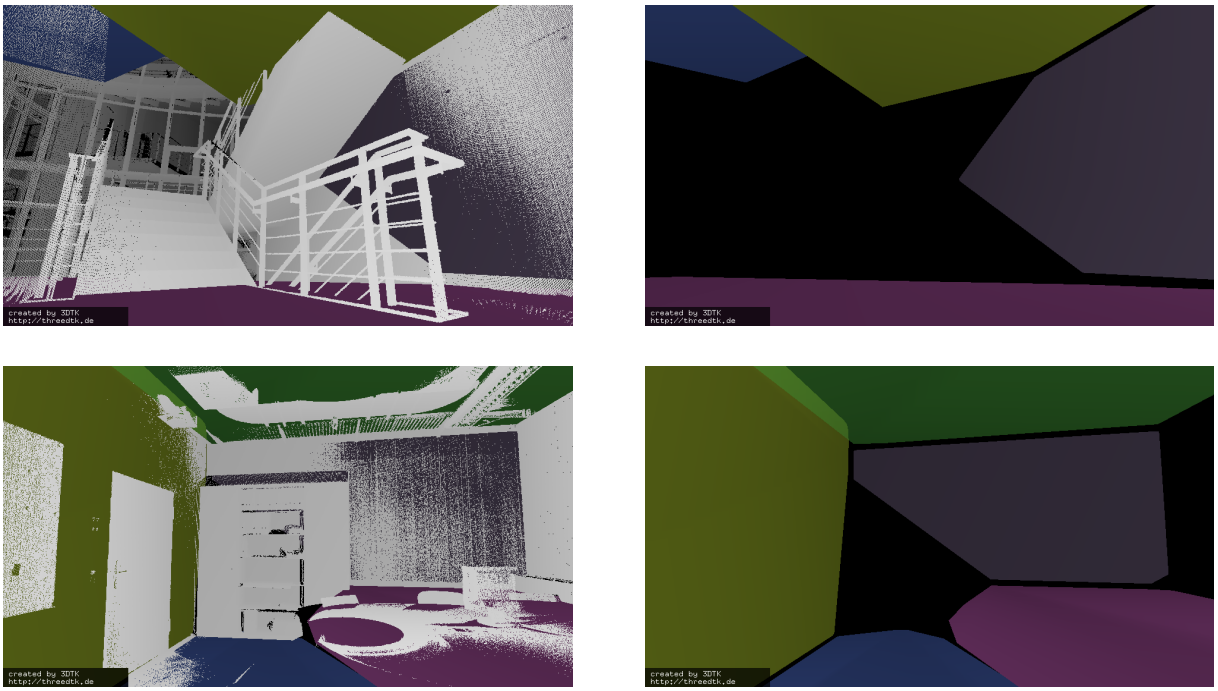


Abbildung 4.10: Gute Ergebnisse der Randomisierten Hough-Transformation

4.7 Praktische Erfahrungen mit den Verfahren

Die verschiedenen besprochenen Verfahren besitzen alle unterschiedliche Kombinationen von Parametern, die eventuell angepasst werden müssen. In diesem Abschnitt soll demnach beschrieben werden, wie umgänglich die einzelnen Methoden sind und was eventuell für Probleme sorgen könnte.

Die meisten Schwierigkeiten bereitete die Adaptive Probabilistische Hough-Transformation. Hier kann es bei falscher Wahl der Parameter durchaus mal dazu kommen, dass keine Ebene gefunden wird. Das Erkennen von vielen Ebenen ist daher teilweise schon eine Herausforderung. Bei der

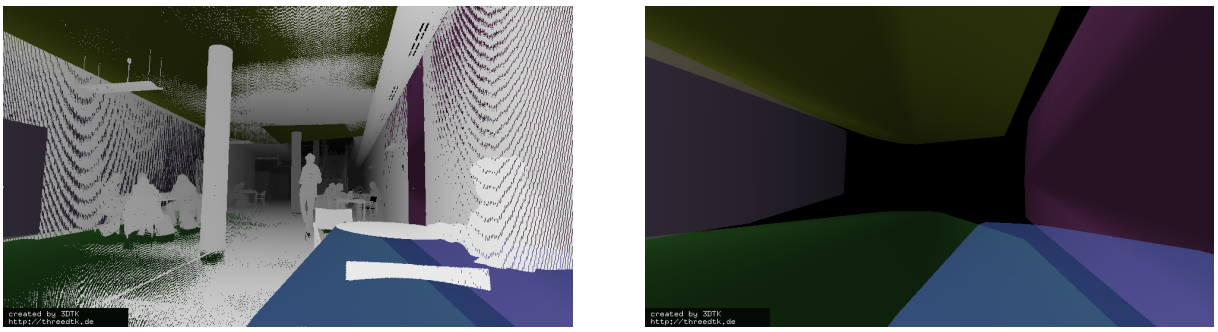


Abbildung 4.11: Linke Wand des Aufenthaltsbereiches endlich erkannt

SHT und PHT treten häufig Probleme beim Finden der Maxima nach dem Transformieren der Punkte. Wie in Kapitel 4.1 bereits erwähnt, spielt die größe des Bereiches in dem die Scores addiert werden, eine große Rolle. Diese ist jedoch ohne Erfahrung schwer einzuschätzen.

Die letzten beiden Hough-Transformationen laufen auf den erstellten Datensätzen ohne große Anpassungen einwandfrei. Natürlich gibt es trotzdem noch einstellbare Parameter, sie sind aber verständlich und dementsprechend leichter zu beurteilen. Der am einfachsten zu händelnde Algorithmus ist allerdings der RANSAC-Algorithmus. Die Funktionsweise des Verfahrens ist sehr gut zu verstehen. Auch die wenigen vorhandenen Parameter machen keine Umstände und lassen sich einfach und intuitiv festlegen.

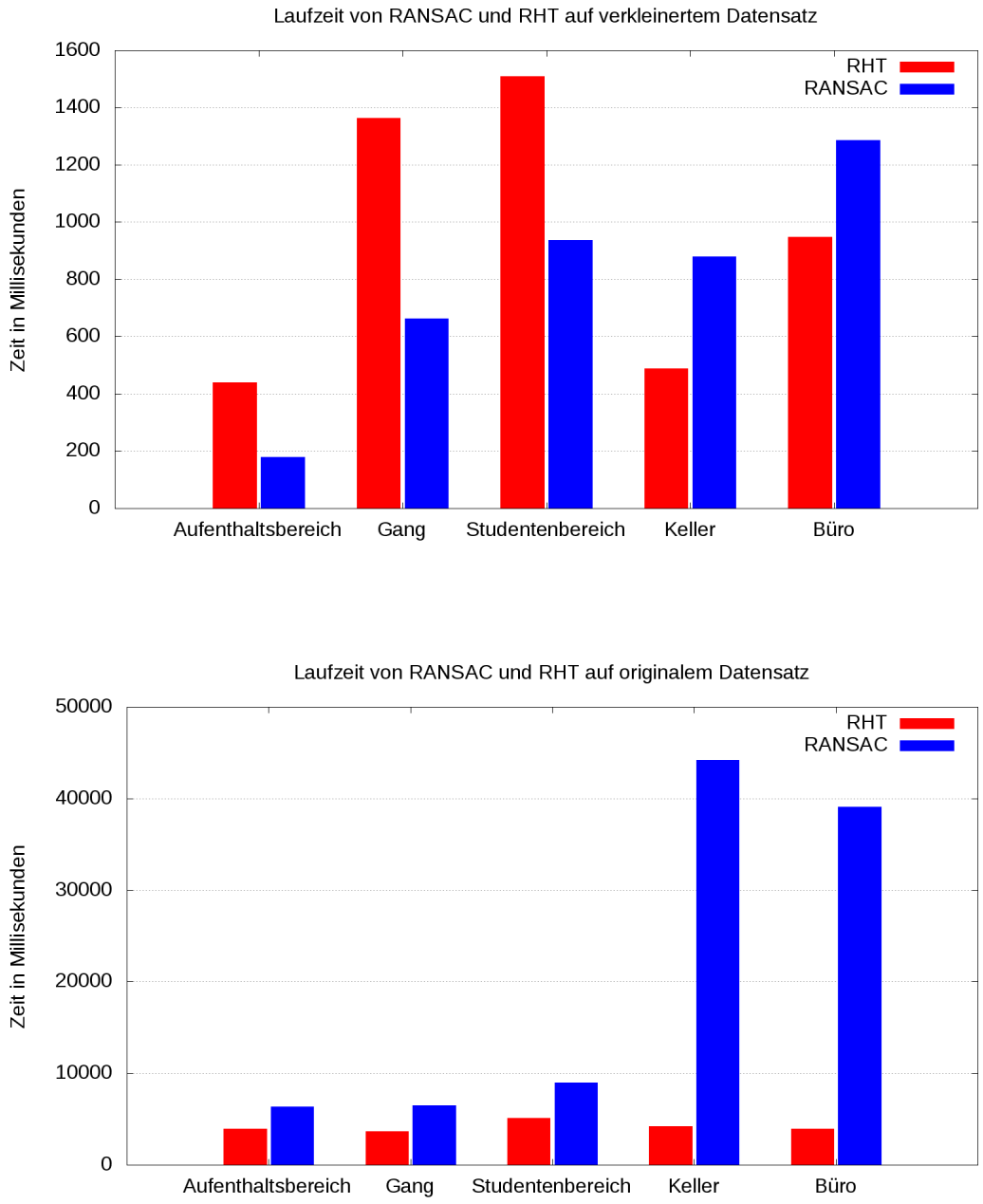


Abbildung 4.12: Laufzeiten von RHT und RANSAC auf verkleinerten und originalen Datensätzen

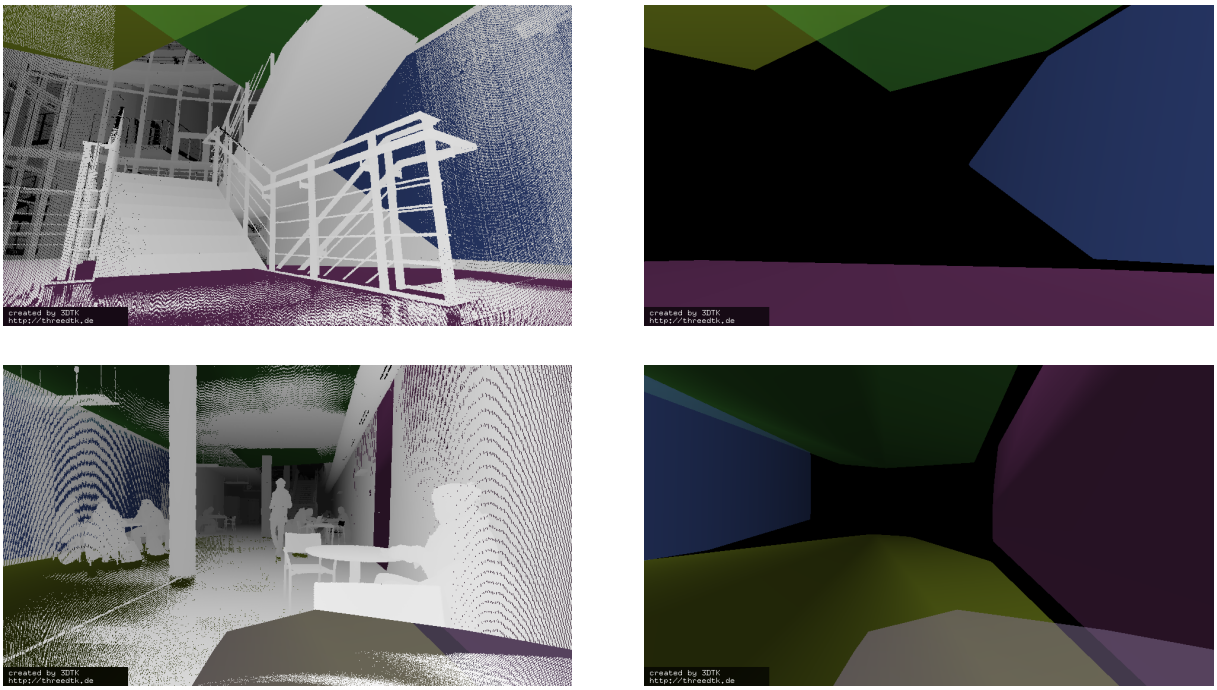


Abbildung 4.13: Gute Ergebnisse des RANSAC-Algorithmus

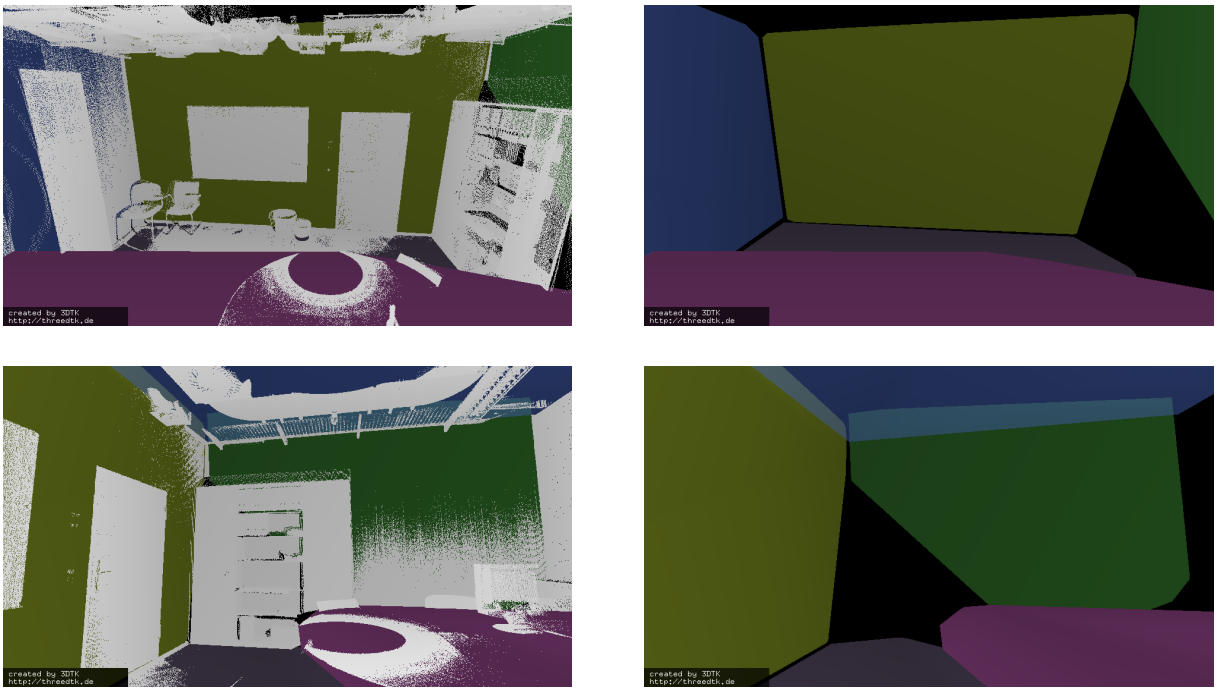


Abbildung 4.14: Unterschiedliche Ergebnisse bei ähnlich großen Ebenen

Kapitel 5

Fazit

Grundsätzlich kann mit jedem der genannten Verfahren gearbeitet werden, Laufzeiten und Bedienbarkeit unterscheiden sich teilweise jedoch gewaltig. Die Berechnungsdauer der Standard Hough-Transformation und die Erfahrung die benötigt wird um die Parameter richtig zu wählen machen das Verfahren unattraktiv. Die Probabilistische und Adaptive Probabilistische Version arbeiten zwar schneller, sind aber beide, insbesondere die APHT, nicht sonderlich umgänglich. Im Verhältnis zu den übrigen Verfahren sind sie aber trotzdem noch relativ langsam. Da auch die Resultate dieser drei Transformationen nicht überzeugen ist von ihnen abzuraten.

Die PPHT stellt zu den bisherigen Verfahren eine deutliche Steigerung auf allen Gebieten dar. Das Anpassen der Parameter stellt keine Herausforderung mehr da und es werden schnell gute Ergebnisse geliefert. Trotzdem kann sie nicht ganz mit der Randomisierten Hough-Transformation und dem RANSAC-Algorithmus mithalten. Die Qualität der Ebenerkennung und die Berechnungsdauer ist bei ihnen am besten. Werden die beiden Methoden direkt miteinander verglichen, so fällt auf, dass die RHT schneller ist. Im Ausgleich dazu ist der RANSAC leichter zu händeln und die Wahl für den passenden Algorithmus hängt dementsprechend von der Person und Datensatz ab. Ist die Punktwolke riesig oder es ist ein sehr schnelles Finden von Ebenen notwendig ist die Randomisierte Hough-Transformation zu empfehlen. Sind die verschiedenen Scans sehr unterschiedlich oder ist ein für den Nutzer einfach zu verstehendes Verfahren gefragt ist der RANSAC-Algorithmus zu bevorzugen.

5.1 Anhang zur Geschwindigkeit von RANSAC

Die Berechnungsdauer des Algorithmus kann wie bereits erwähnt nicht mit der der Randomisierten Hough-Transformation mithalten. Mit einer geschickt gewählten Abbruchbedingung könnte sich das jedoch ändern. Die bisherige Implementierung mit 5000 festen Iterationen ist sehr einfach gewählt. Sie sorgt für ein sicheres Finden von einer der größten Ebene, ist aber bei Weitem nicht auf die Laufzeit optimiert. Wenn viele verschiedene Ebenen in einem Datensatz erkannt werden sollen kann die Anzahl der Iterationen verringert werden, weil es dann nicht

mehr so wichtig ist das eine der größten oder die größte Ebene zuerst gefunden wird. Außerdem kann in diesem Fall noch eine weitere Abbruchbedingung integriert werden. Falls eine Ebene entdeckt wird, auf der mindestens ein gewisser Anteil, beispielsweise 40%, der Gesamtpunkte liegen, kann die Suche vorab abgebrochen werden. Eine ausreichend gute Ebene wurde ja dann schon erkannt. Hinzukommend kann wie beim RHT auch eine Vorbedingung eingebaut werden, welche die Abstände der drei ausgewählten Punkte überprüft. Liegen diese außerhalb von vorgegebenen Grenzwerten werden direkt drei neue Punkte bestimmt ohne Rechenleistung an einer Ebene zu verschwenden die wahrscheinlich sowieso nicht gut ist.

Durch besagte Veränderungen würden allerdings einige neue Parameter hinzukommen , wodurch der RANSAC zwar schneller, aber auch schwieriger zu händeln wird. Dies würde einer seiner größten Stärken, die einfache Benutzung, entgegenwirken. Deshalb, und weil der Algorithmus grundsätzlich auch nicht langsam ist, sind diese Laufzeitoptimierungen eher optional.

Literaturverzeichnis

[1] Borrmann, Dorit; Elseberg, Jan; Lingemann, Kai; Nüchter, Andreas: The 3D Hough Transform fo Plane Detection in Point Clouds: A Review and a new Accumulator Design. In: *3D Research Center and Springer*, 2011

[2] Elseberg, Jan; Borrmann, Dorit; Nüchter, Andreas: One billion points in the cloud – an octree for efficient processing of 3D laser scans. In: *ISPRS Journal of Photogrammetry and Remote Sensing* Bd. 76. Jacobs University Bremen gGmbH, Automation Group, School of Engineering and Science, Campus Ring 1, 28759 Bremen, Germany, 2013

[3] Nüchter, Andreas u. a.: *3DTK – The 3D Toolkit.* `http://slam6d.sourceforge.net/`, 2017

[4] Tayez, Tarsha-Kurdi; Tania, Landes; Pierre, Grussenmeyer: Hough-Transform and Extended RANSAC Algorithms for Automatic Detection of 3D Building Roof Planes from Lidar Data. In: *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007* Bd. XXXVI. Espoo, Finland, September 2007

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Würzburg, April 2017