Masterarbeit

Aufbau von volumetrischen Karten mit multiplen Auflösungsstufen durch einen mit Tiefensensoren ausgestatteten mobilen Roboter

Christopher Buschor

DLR

Institut für Robotik und Mechatronik	BJ.: 2015
	IB.Nr.: 572-2015/18

MASTERARBEIT

AUFBAU VON VOLUMETRISCHEN KARTEN MIT MULTIPLEN AUFLÖSUNGSSTUFEN DURCH EINEN MIT TIEFENSENSOREN AUSGESTATTETEN MOBILEN ROBOTER

Freigabe:	Der Bearbeiter:	Unterschriften
	Christopher Buschor	C. Buschor
	Die Betreuer:	
	Dipl.-Math. Christian Rink	
	Dipl.-Math. Daniel Seth	
	Der Abteilungsleiter	
	Prof. Dr.-Ing. Alin Albu-Schäffer	
	Der Institutsdirektor	
	Prof. Dr.-Ing. Alin Albu-Schäffer	

Dieser Bericht enthält 96 Seiten, 34 Abbildungen und 5 Tabellen.

Ort: Oberpfaffenhofen	Datum: 07.05.2015	Bearbeiter: Christopher Buschor	Zeichen:

Masterarbeit

Aufbau von volumetrischen Karten mit multiplen Auflösungsstufen durch einen mit Tiefensensoren ausgestatteten mobilen Roboter

Christopher Peter Ulrich Buschor

29. April 2015

Lehrstuhl für Datenverarbeitung
Technische Universität München

Christopher Peter Ulrich Buschor. *Aufbau von volumetrischen Karten mit multiplen Auflösungsstufen durch einen mit Tiefensensoren ausgestatteten mobilen Roboter.* Masterarbeit, Technische Universität München, München, 2015.

Betreut von Prof. Dr.-Ing. K. Diepold , Dipl.-Math Daniel Seth (DLR) und Dipl.-Math Christian Rink (DLR); eingereicht am 29. April 2015 bei der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München.

Lehrstuhl für Datenverarbeitung, Technische Universität München, 80290 München, `http://www.ldv.ei.tum.de`.

Zusammenfassung

Am Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt (DLR) wurden Algorithmen entwickelt, die während der Bewegung des „omniRob“ von Kuka, mithilfe der durch die Tiefensensoren akquirierten Informationen, eine 3D-Karte erstellen. Auf Basis dieser Karte wird auch die Navigation des Roboters durchgeführt, sodass für Explorationsanwendungen unbekanntes Gebiet fortlaufend erkundet werden kann.
Als Kartentyp wird ein Voxelgitter in einem aus Octrees aufgebauten Raum benutzt, welches auf dem probabilistischen Modell von Occupancy Maps [60] basiert. Das Voxelgitter stellt eine Diskretisierung des Raumes in ein regelmäßiges Gitter[1] dar. Die Messdaten der Sensoren werden auf Voxelebene aktualisiert und benachbarte Voxel mit gleichem Zustand ggf. entsprechend der Octree-Datenstruktur zusammengefasst.
Es wurde in dieser Arbeit eine Übersicht über verschiedene aktuelle Mapping-Verfahren gegeben und deren Charakteristika herausgearbeitet. Zur Weiterentwicklung des momentan am Institut verwendeten Verfahrens wurde die Idee aufgenommen, Daten in der Karte in verschiedenen Auflösungsstufen zu speichern, welche von der Qualität der zu integrierenden Daten abhängig sind.
Jene Funktionalität wurde in dieser Arbeit entwickelt, auf das bisherige Verfahren angewendet und erfolgreich getestet. Die Auflösung von Karten kann jetzt, während ihrer Erstellung, dynamisch angepasst werden, indem die Größe der einzufügenden Volumenelemente entsprechend gewählt wird. Je geringer die Messunsicherheit, desto kleinere Elemente werden in die Karte integriert, was in höher aufgelösten Bereichen resultiert.
Es konnte gezeigt werden, dass bei Verwendung des hier entwickelten Verfahrens die benötigte Zeit zum Aufbau einer Karte sowie der benötigte Speicherbedarf deutlich reduziert werden konnten. In einem repräsentativen Testfall zur Integration eines Tiefenbildes in eine Karte sowie deren Speicherung als .obj-Datei wurde die Rechenzeit um 69 % und der Speicherbedarf dieser Datei um 96 % reduziert.
Die maximale Auflösung der Karte kann jetzt sehr hoch gewählt werden, da die Karte nicht mehr konstant in einer gewählten Auflösung erstellt wird, sondern während des Mappings dynamisch angepasst wird. Es können interessante Bereiche von einem Roboter nah angefahren und auf hoher Auflösung vermessen werden und andere, weiter entfernte Bereiche in geringerer Auflösung gespeichert werden. Die gesamte Karte kann somit bezüglich des Speicherbedarfs klein gehalten werden, obwohl sie hoch aufgelöste Bereiche enthält.

[1] Ein regelmäßiges Gitter unterteilt den Raum vollständig in achsenparallele rechtwinklige Bereiche, wobei Kanten entlang einer Achse stets die gleiche Länge besitzen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

1.1 Motivation

Am Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt (DLR) wurden Algorithmen entwickelt, die während der Bewegung der mobilen Plattform des „Rollin' Justin“ oder dem „omniRob“ von Kuka mithilfe der durch die entsprechenden Sensoren akquirierten Tiefeninformationen eine 3D-Karte erstellen. Auf Basis dieser Karte wird auch die nötige Navigation der Roboter bewerkstelligt, sodass unbekanntes Gebiet fortlaufend ohne Gefahr von Kollisionen erkundet werden kann und nicht nur der als sicher vordefinierte Bereich um den initialen Standort des Roboters als Observationsposition verwendet werden kann.

Dem Problem, aus gegebenen Sensordaten mit den dazugehörigen Sensorposen[1], die entsprechenden Karten zu erstellen, widmen sich Mappingverfahren. Als Sensordaten stehen hier Tiefeninformationen durch die Tiefensensoren des Roboters, wie Time-of-Flight-Kameras (ToF-Kameras) oder Laserscanner, zur Verfügung.

Um den Messdaten der Sensoren, während der Bewegung des Roboters, die jeweilige Sensorpose zuordnen zu können, existieren diverse SLAM-Verfahren (Simultaneous Localization And Mapping). SLAM-Verfahren stellen eine Technik dar, um mit autonomen Robotern eine Karte in unbekannter Umgebung zu erstellen oder aktualisieren während fortlaufender Lagebestimmung des Roboters. Eine Möglichkeit der Lagebestimmung für das SLAM-Verfahren, die momentan am Institut verwendet wird, ist die Nutzung von Sensoren zur Odometrie[2].

Als Kartentyp wird ein Voxelgitter in einem aus Octrees aufgebauten Raum benutzt, welches auf dem probabilistischen Modell von Occupancy Maps [60] basiert. Das Voxelgitter stellt eine Diskretisierung des Raumes in ein regelmäßiges Gitter[3] aus Würfeln, den Voxeln, dar. Die Daten der Sensoren werden auf Voxelebene aktualisiert und benachbarte Voxel mit gleichem Zustand ggf. entsprechend der Octree-Datenstruktur zusammengefasst.

[1] Die Position des Sensors im Raum, sowie dessen Orientierung

[2] Positionsbestimmung eines mobilen Systems anhand der Daten seines Antriebssystems

[3] Ein regelmäßiges Gitter unterteilt den Raum vollständig in achsenparallele rechtwinklige Bereiche, wobei Kanten entlang einer Achse stets die gleiche Länge besitzen.

Auf dem Gebiet des 3D-Mappings wird viel geforscht und neue Verfahren vorgestellt. Es ist davon auszugehen, dass es (neue) Ideen gibt, mit denen das derzeitige Verfahren verbessert werden kann. Eine Verbesserung wäre es beispielsweise, wenn das Mapping weniger Rechenzeit oder Rechenleistung beansprucht, die Karten eine höhere Qualität aufweisen oder der Speicherbedarf der Karten sinkt. Es sollen daher aktuelle Verfahren untersucht werden und die darin verwendeten Techniken dahingehend analysiert werden, ob Teile davon sinnvoll in das momentan verwendete Mapping-Verfahren integriert werden können.

1.2 Aufgabenstellung und Ziele

Momentan wird das Mapping auf einer Voxelkarte in einem probabilistischen Raum nach [10] unter Verwendung des Dynamic-Octree-Raumes nach [80] durchgeführt. Es wurden nach Implementierung dieses Mappingverfahrens, welches stets weiterentwickelt wird, diverse neue Verfahren zum 3D-Mapping vorgestellt. Es soll in dieser Arbeit eine Übersicht über aktuelle Mapping-Verfahren gegeben werden und untersucht werden, wie mithilfe der untersuchten Methoden die aktuell am Institut verwendete Methode bezüglich diverser Kriterien verbessert werden kann. Kriterien sind der Speicherbedarf der entstehenden Karte, die benötigte Zeit zum Aufbau der Karte, sowie die Qualität der Karte. Optimierungsideen, basierend auf der Inspektion diverser Verfahren sollen adaptiert, implementiert und mit dem bisherigen Verfahren verglichen werden.

1.3 Aufbau der Arbeit

Nachdem in Kapitel 1 die Aufgabenstellung dargestellt und für das Thema motiviert wurde, wird in Kapitel 2 das Problem des Mappings erläutert und es werden verschiedene generelle Lösungsansätze vorgestellt. Konkrete aktuelle Verfahren werden in Kapitel 3 genauer analysiert und nach verwendbaren Ideen, zur Anwendung auf das derzeit am Institut existierende Verfahren, durchsucht. In Kapitel 4 wird ein Überblick über Techniken der Tiefensensorik gegeben und es werden die verschiedenen Sensorarten auf Verwendbarkeit für das zu entwickelnde Mapping-Verfahren bewertet. Die Umsetzung der Idee der Karten mit multiplen Auflösungsstufen wird in Kapitel 5 beschrieben. Tests der Implementierung und Vergleiche mit dem bisherigen Verfahren sind in Kapitel 6 dargestellt. In Kapitel 7 werden die Ergebnisse zusammengefasst und es wird ein Ausblick gegeben.

2 Stand der Technik

SLAM: Mobile Roboter müssen ihre Lage im Raum kennen, um kollisionsfrei navigieren und dabei fortlaufend eine Karte aufbauen zu können. Zu Beginn einer Exploration ist meist keine Karte vorhanden. Der Roboter vermisst somit die Umgebung initial und erstellt eine Karte, deren Mittelpunkt für gewöhnlich die aktuelle Roboterposition definiert. Falls eine Karte des Raumes, in dem sich der Roboter befindet, bereits verfügbar ist, so kann sich der Roboter mithilfe seiner Sensoren darin lokalisieren. Dazu vermisst der Roboter die relative Lage von in der Karte bereits vorhandenen Objekten zu ihm und kann mit den bekannten Positionen der Objekte seine absolute Lage schätzen. Während der ausreichend langsamen Bewegung des Roboters wird durch neue Messungen ein Teil der Objekte der bereits erstellten Karte erneut gemessen sowie ein bis dato noch unbekannter Teil. Aus der Überlappung des bereits bekannten Teils der neuen Aufnahmen kann die Bewegung des Roboters berechnet werden. Da die Berechnung der Bewegung des Roboters zwischen zwei Aufnahmen jedoch nie exakt ist, wird die berechnete Position des Roboters von der tatsächlichen immer weiter abweichen. Durch die immer stärker abweichende berechnete Position des Roboters wird der Versatz der neu zu integrierenden Tiefenbilder größer, was zu einer Abnahme der Qualität der Karte führt.

SLAM ist somit ein Henne-Ei-Problem, da weder die Karte noch die Position bekannt ist, sondern diese gleichzeitig geschätzt werden sollen. Erste Lösungsvorschläge des Problems wurden von [75] und [23] vorgestellt. Frühe SLAM-Verfahren hatten sich hauptsächlich mit der Akquirierung von 2D-Karten beschäftigt. Dabei wurden vorzugsweise Laserscanner oder Sonar-Sensoren verwendet.

Inzwischen wurden viele Verfahren, wie u.A. in [63, 54, 73] vorgestellt, welche von einer Robotertrajektorie von sechs Freiheitsgraden ausgehen und bei denen eine 3D-Scan-Registrierung stattfindet. Scans werden dabei häufig von Iterative Closest Points (ICP) Algorithmen [7] registriert. ICP-Algorithmen arbeiten auf Rohpunkten. Der Aufwand der Berechnung ist bei hochauflösenden Bildern, mit entsprechend vielen Rohpunkten, sehr hoch. Durch die erhebliche Rechenzeit pro Bild sinkt dadurch die Bildwiederholrate. Um den Aufwand zu reduzieren und damit höhere Bildwiederholraten zu erreichen, wird deshalb bei vielen Verfahren ein Subsampling der Rohpunkte durchgeführt. Für diese Verarbeitung eigenen sich GPUs durch ihre ausgeprägt parallelisierte Verarbeitung sehr gut. Ein moderner, sehr effizienter ICP-Algorithmus, welcher auf GPUs rechnet, ist in [62] vorgestellt.

Mapping: Das prinzipielle Problem mit welchem sich Mappingverfahren beschäftigen, ist es, aus gegebenen Sensordaten z_n mit den dazugehörigen Sensorposen u_n, die entsprechende Karte

$$m^* = \arg\max_m P(m|d) \quad \text{mit} \quad d = \{u_1, z_1, u_1, z_2, ..., u_n, z_n\} \tag{2.1}$$

zu erstellen.

Generell lassen sich Mapping-Methoden in zwei verschiedene Ansätze unterteilen.

Zum einen existieren *visuell Feature-basierte Methoden*, wie in [1, 17, 37, 27]. Bestimmte Strukturen werden dabei erfasst und gespeichert, ohne, dass die räumliche Struktur aller gemessenen Objekte in der Karte gespeichert wird. Dadurch entsteht das Problem, dass diese Karten zur Navigation von Robotern nicht ohne Einschränkungen genutzt werden können, da einerseits Regionen ohne ausgeprägte Struktur oder Merkmale vorhanden sind und es andererseits in Regionen mit solcher Struktur schwer ist, zwischen freien und besetzten Raum zu unterscheiden. Die Kollisionsvermeidung bei der Bewegung mobiler Roboter ist darauf demnach kaum zu gewährleisten.

Zum anderen gibt es *volumetrische Ansätze*, bei denen der Raum in Volumenelemente aufgeteilt wird und diesen Besetzungswahrscheinlichkeiten zugewiesen werden. Objekten werden festen Koordinaten in diesem Raum zugewiesen. Jeder vermessene Raumpunkt wird mit der jeweiligen Besetzungswahrscheinlichkeit in der Karte gespeichert. Existiert durch vorherige Messungen an einer Stelle der Karte, an der ein Punkt eingefügt werden soll, bereits ein Punkt, so wird die bisherige Besetzungswahrscheinlichkeit mit der neuen entsprechend einer Vorschrift kombiniert. Die dadurch entstehenden Karten sind sehr gut zur Navigation geeignet, da freier und besetzter Raum klar definiert abgespeichert ist und einfach ausgelesen werden kann. Diese Form des Mappings findet u.A. in [80, 44, 76, 62] Anwendung.

Bei den bisher genannten Verfahren wird stets von statischen Karten ausgegangen. Bei diesen gibt es per Annahme keine sich verändernden Objekte. In der realen Welt sind jedoch dynamische Objekte vorhanden, welche je nach Anwendung von bedeutender Rolle sind. In der Service-Robotik beispielsweise, in der Roboter mit diversen Objekten interagieren, muss diese Dynamik in Karten beachtet werden. In [36, 92, 94] wird eine Dynamik erfasst, jedoch nur als Störung interpretiert. Dabei wird versucht, diese Störung herauszufiltern und nicht nutzbar in Karten zu speichern. In [20, 49, 53, 69] wird die Bewegung von Objekten geschätzt, um eine dynamische Karte zu erhalten. Bei diesen Verfahren werden die Zellen stets unabhängig voneinander angenommen. Veränderungen in einer Zelle beeinflussen demnach Nachbarzellen nicht. Dieses Verhalten entspricht zwar nicht der Realität, da Objekte die Umgebung sehr wohl beeinflussen können, durch diese Annahme sind die Verfahren jedoch schnell und weniger aufwendig, als wenn der Einfluss auf Nachbarzellen beachtet werden würde. Dieser Einfluss auf Nachbarzellen wird in [50, 93] modelliert.

Nachfolgend sollen die verschiedenen grundlegenden Ansätze zur Bewältigung des Mapping-Problems überblicksartig dargestellt werden.

- **Occupancy Grid Maps**

  Occupancy Grid Maps wurden von Moravec und Elfes in 1985 [60] eingeführt. Sie repräsentieren die Welt als Gitter von Zellen, in denen Besetzungswahrscheinlichkeiten gespeichert werden. Unter der Voraussetzung, dass die Pose des Roboters bekannt ist, können damit die verrauschten und unsicheren Messdaten von Sensoren abbildet werden.

  Occupancy Grid Maps sind nie vollständig korrekt, da die Daten in nicht beliebig kleine Würfel diskretisiert werden können. Durch die Wahl ausreichend kleiner Würfel kann jedoch ein ausreichend genaues Modell gespeichert werden.

  Um eine Occupancy Grid Map zu erstellen, ist es notwendig die Besetzungswahrscheinlichkeit jeder Zelle zu bestimmen. Um dies effizient durchführen zu können, wird nach [59] oft die Annahme getätigt, dass die Besetzungswahrscheinlichkeiten der individuellen Zellen stochastisch unabhängig sind. Somit kann die Wahrscheinlichkeit einer partikulären Karte m in das Produkt der individuellen Wahrscheinlichkeiten der enthaltenen Zellen zerlegt werden. Die Karte hängt dabei von den vorherigen Positionen des Roboters x^t und den zugehörigen Sensordaten an diesen Positionen z^t ab:

$$p(m|x^t, z^t) = \prod_n p(m_n|x^t, z^t) \; . \tag{2.2}$$

- **Temporal Occupancy Grid Maps**

  In [19] wird eine Erweiterung zu den Occupancy Grid Maps vorgeschlagen. Die Temporal Occupancy Grid Map ist eine Occupancy Grid Map, die aus mehreren Schichten besteht. Jede Schicht beinhaltet die Besetzungswahrscheinlichkeiten zu bestimmten Zeitpunkten. Ein Zeitabschnitt wird dazu in mehrere Zeitpunkte diskretisiert und für jeden Zeitpunkt eine Schicht erstellt. Eine Zelle speichert also in den Schichten die Besetzungswahrscheinlichkeiten für mehrere Zeitpunkte. Durch diese Funktionalität können dynamische Umgebungen in dieser Karte entsprechend gespeichert werden.

- **Dymodda**

  Dymodda, der „DYnamic Multiple-Octree Disctretionary Data spAce“, basiert auf einer Voxelkarte in dem Probabilistic-Dynamic-Octree-Raum.

Der Dynamic-Octree-Raum, wie in [10] beschrieben, besteht aus Voxeln, welche bei entsprechender geometrischer Lage und gleichem Inhalt effizient in einer Octree-Datenstruktur zusammengefasst werden. Ein Octree ist ein gewurzelter Baum, dessen Knoten jeweils entweder acht Kinder oder gar keine Kinder haben. Sie werden hier benutzt um dreidimensionale Datensätze hierarchisch unterteilt zu speichern [72]. Somit sinkt der Speicherbedarf, insbesondere bei Karten mit vielen homogenen Objekten, sehr stark. Die Karte kann dynamisch erweitert werden, indem weitere Octrees an den entsprechenden Stellen hinzugefügt werden.

Durch die weitere Verwendung des Probabilistic-Raumes, welcher in [80] eingeführt wurde, wird für jedes Voxel eine Besetzungswahrscheinlichkeit gespeichert. Dadurch kann die Unsicherheit der Sensoren bei Messungen implementiert werden. Als Update-Verfahren stehen drei verschiedene probabilistische Algorithmen zur Verfügung: Ein Bayes-basiertes, ein Fuzzy-basiertes und ein Dempster-Shafer basiertes Verfahren. Zusätzlich kann ein naiver Ansatz gewählt werden.

Durch die templatisierte Implementierung können weitere Eigenschaften leicht hinzugefügt werden.

- **OctoMap**

  Eine Alternative zu Dymodda stellt die in [38] vorgestellte OctoMap dar.

  Dieses Verfahren ist Dymodda sehr ähnlich. Als zugrunde liegende Datenstruktur werden Octrees verwendet, eine Expansion des Raumes durch dynamisches Hinzufügen von Octrees ist möglich und weitere Eigenschaften können durch die templatisierte Implementierung leicht hinzugefügt werden.

  Ein Unterschied liegt darin, dass die OctoMap als Updateverfahren lediglich den probabilistischen Algorithmus von [60] verwendet, wohingegen bei Dymodda mehrere Algorithmen zur Verfügung stehen.

- **Surfel Maps**

  Ziel der Surfel Maps ist es, eine Struktur zur Verfügung zu stellen, mit welcher neue Rohdaten von dreidimensionale Objekten schnell gerendert werden können. Im Gegensatz zu konventionellen Oberflächendiskretisierungen (z.B. Dreiecksnetzen) passen sich Surfels der Auflösung des Framebuffers an.

  Ein Surfel kann als Ansammlung von Punkten gesehen werden, welche eine Oberfläche repräsentieren. Zusätzlich zu den Punkten selbst, werden Attribute, wie die Statistiken zur räumlichen Verbindung und die Farbverteilung der Punkte, für jedes Surfel gespeichert. Beim Render-Prozess wird dann jedes Surfel mit Farbinformation gefüllt und auf die entsprechenden Pixel rasterisiert.

  Surfel Objekte bieten die Möglichkeit der Darstellung komplexer Objekte bei niedrigen Renderkosten und zugleich hoher Bildqualität [65]. Dadurch sind sie besonders

für niedrigpreisige Echtzeitgrafikanwendungen, wie Computer- und Konsolenspiele, geeignet. In [56] wird untersucht, unter welchen Bedingungen die Surfel-Darstellung konventionellen Techniken überlegen ist.

- **Conditional Transition Maps**

  Conditional Transition Maps werden in [50] vorgestellt. Sie sind eine Gitter-basierte-Repräsentation, welche die Bewegung von dynamischen Objekten modelliert. Die Modellierung erfolgt durch eine Wahrscheinlichkeitsverteilung für Objekte welche eine Zelle verlassen. Die Eintrittsrichtung der Objekte muss dazu jeweils bekannt sein. Die Bewegungsparameter werden dabei aus einem temporären Signal der Besetzungswahrscheinlichkeit von Zellen durch die Verwendung der Local-Neighborhood-Kreuzkorrelation-Methode bestimmt.

- **Spatial Affordance Map**

  Die Spatial Affordance Map wird in [53] vorgestellt. Sie wird benutzt, um Personen zu erfassen und zu verfolgen. Diese Karten sind nützlich in Situationen, in denen Roboter und Menschen zusammen arbeiten. Die Karte stellt eine Langzeit-Repräsentation von menschlichen Aktivitäten dar. Das Model ist Gitter-basiert und geht davon aus, dass eine menschliche Bewegung in einer Umgebung ein unabhängiges Event ist. Events werden hierbei als inhomogene Poisson-Prozesse modelliert.

- **Reflection Maps / Enviroment Maps**

  Dieses Mapping Verfahren wurde erstmals von Blinn und Newell 1976 in [9] vorgestellt. Das Environment Mapping, auch als Reflection Mapping bezeichnet, ist eine effiziente Methode zur Simulation von spiegelnden Oberflächen. Die gespiegelte Umgebung wird dabei als Textur gespeichert und auf das entsprechende Objekt abgebildet.

  Zur Texturspeicherung werden im wesentlichen folgende zwei Methoden verwendet. Beim *sphärischen Mapping* wird die Textur auf das Innere einer Kugel abgebildet, wohingegen beim *kubischen Mapping* die Textur auf die sechs Seiten eines Würfels abgebildet wird. Bei beiden Verfahren wird zunächst für jedes Pixel auf dem reflektierenden Objekt die Normale sowie der Reflexionsvektor, basierend auf der Observationsposition des Sensors und der Oberflächennormalen, berechnet.

  Ein Nachteil dieser Technik liegt darin, dass die spiegelnde Umgebung bereits vor dem Rendern des Objekts bekannt sein muss. Selbstreflexionen und sich ändernde Umgebungen sind somit nicht direkt erfassbar.

- **Topologische Karten**

  Eine topologische Repräsentation der Umwelt, bestehend aus Knoten, Kanten und Gewichten. Ein Knoten stellt jeweils einen diskreten Ort dar, Kanten beschreiben die

Wege zwischen den Knoten und die Gewichte sind ein Maß für die Länge der Wege [78]. Knoten werden in der Regel für wichtige Positionen, wie z.B. einem Eingang, einem Ausgang, einer Gefahrenquelle, etc., erstellt.

- **Selbst-organisierende Karten**

  Selbst-organisierende Karten wurden von Teuvo Kohonen entwickelt und in [45, 46] vorgestellt. Es handelt sich dabei um ein spezielles künstliches neuronales Netz, das sich, nach dem Vorbild des menschlichen Gehirns, selbst organisiert.

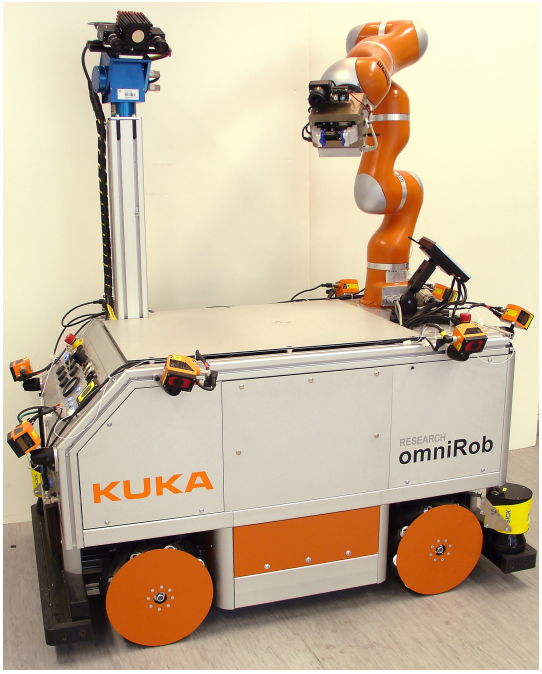
Diverse **2D-Mapping Methoden** wurden ausführlich in [83] beschrieben und teilweise in [80] adaptiert und verwendet.

3 Analyse der existierenden Verfahren

3.1 Annahmen und Voraussetzungen

Es werden folgende Voraussetzung an das Verfahren gestellt und Gegebenheiten dargestellt bzw. angenommen:

- Als Datenstruktur werden derzeit Octrees verwendet und sollen auch weiterhin verwendet werden.
- Das Verfahren muss insbesondere für Mapping von Innenräumen geeignet sein.
- Der Algorithmus muss effizient auf einer CPU (nicht GPU) lauffähig sein.
- Bei dem Roboter, auf dem das Verfahren später implementiert werden soll, handelt es sich um den „omniRob“ von Kuka, dargestellt in Abbildung 3.1(a):
  - Länge, Breite, Höhe: 1,20 m, 0,71 m, 0,65 m
  - Gewicht: ca. 270 kg zzgl. Bleigel-Akku (48 V, 80 Ah, ca. 100 kg)
  - Antrieb: Mecanum-Räder für omnidirektionale Fahrmanöver
  - Sensorik: 8 ToF-Kameras vom Typ O3D100 mit einer Auflösung von 50 x 64 Pixel aus Abbildung 3.1(b)
  - Anforderungen an den Untergrund: Eben, da der Roboter keine Federung hat.
- Es stehen ausschließlich Tiefendaten von Tiefensensoren als Information zum Kartenbau zur Verfügung - keine Textur oder Farbinformationen.
- Es existiert eine Simulationsumgebung, in der das Verfahren mit dem vorgestellten Roboter evaluiert werden kann.



(a) „omniRob“ von Kuka in vom DLR umgebauter Version, Bild: Simon Kriegel, DLR

(b) ToF-Kamera IFM O3D100, Bild: Stefan Fuchs, DLR

Abbildung 3.1: „omniRob“ von Kuka in vom DLR umgebauter Version sowie eine der darauf montierten ToF-Kameras

3.2 Analyse ausgewählter aktueller Verfahren

In einigen Veröffentlichungen der letzten zwei Jahre werden Verfahren für große Umgebungen [77, 70] vorgestellt. Manche Verfahren beschäftigen sich außerdem mit dem Mapping von dynamischen Objekten [71, 77], wobei [71] diese Dynamik außerdem für große Umgebungen betrachtet. Für Kartentypen, wie kognitive Karten zur Navigation [84] oder topologische 2D-Karten [21] werden ebenfalls Verfahren vorgestellt.

Diese Verfahren sind für diese Arbeit ungeeignet. Fünf aktuelle Verfahren die für die Anwendung in dieser Arbeit infrage kommen, werden im Folgenden näher betrachtet.

3.2.1 Continuous-Occupancy-Maps mit Benutzung überlappender lokaler Gaußprozesse

Konventionelle Occupancy Grid Maps bestehen aus unabhängigen Zellen. Jede Zelle wird bezüglich ihrer Besetzungswahrscheinlichkeit individuell aktualisiert. Der Algorithmus ar-

beitet unter dieser Voraussetzung sehr schnell und ist leicht zu implementieren. Er ist jedoch ungenau, da nur Zellen, welche von einem Sensorstrahl getroffen werden, aktualisiert werden und keine umliegenden. Ein realistisches Modell bezieht auch umliegende Zellen mit ein, was bei konventionellen Karten nicht erfolgt und somit zu einer geringeren Qualität der entstehenden Karte führt.

In [44] wird ein realistisches Modell durch die Verwendung von Gaußprozessen erreicht. Der Nachteil bei der Verwendung von Gaußprozessen ist ihr hoher Rechenaufwand von O(N^3) mit N = Anzahl an Trainingsdaten. Das Vorgehen ist somit nicht für große Umgebungen geeignet, da diese eine hohe Anzahl an Trainingsdaten erfordern würden. Das Hauptaugenmerk in [44] liegt auf der Reduzierung des Rechenaufwandes bei der Verwendung solcher Gaußprozesse, während der Erhaltung der Genauigkeit der Karte.

Anstatt die Gauß-Prozess-Klassifikation direkt anzuwenden, wird in einem ersten Schritt die Gauß-Prozess-Regression angewandt und anschließend die probabilistische Klassifikation der kleinsten Quadrate durchgeführt. Es werden dadurch zeitaufwendige Approximationen wie die Laplace-Methode vermieden. Der Verlust an Genauigkeit hält sich dabei in akzeptablem Rahmen.

Als Kovarianz-Funktion wird die M'atern-Kovarianz-Funktion verwendet, da eine quadratische, exponentielle Kovarianz-Funktion aus Methoden wie [64] zu eben wäre, um scharfe Änderungen der Besetzungswahrscheinlichkeiten zu modellieren. Eine Kovarianz-Funktion beschreibt die Ähnlichkeit zweier Punkte. Kontinuierliche Strahlen einer Entfernungsmessung müssen also zunächst in mehrere Punkte diskretisiert werden. Dies würde jedoch zu einer noch größeren Menge an Trainingsdaten führen. Liniensegmente der Strahlen können mithilfe von Integral-Kernels nach [64] als Einzelpunkte behandelt werden. Anschließend wird die Klassifikation der kleinsten Quadrate [66] angewendet.

Um in einem zweiten Schritt den Rechenaufwand weiter zu reduzieren, werden Testpositionen und Trainingsdaten partitioniert und lokale Gaußprozesse auf jedes daraus entstehende Cluster angewandt:

Entfernungsmessungen zur Akquirierung von Trainingsdaten werden durchgeführt, wobei sich die Messungen jeweils aus einem Liniensegment (frei) und einem Auftreffpunkt (besetzt) zusammensetzen. Die Linienelemente beeinflussen die Testposition abhängig der Distanz zwischen ihnen. Es müssten daher, wie in [74] beschrieben, alle Liniensegmente nah genug an jeder Testposition gefunden werden. Dies ist jedoch sehr zeitaufwendig. Anstatt dessen werden hier die Auftreffpunkte mit dem k-Means-Clustering aus [40] aufgeteilt und dann die Liniensegmente in Cluster, mit ihren entsprechenden Auftreffpunkten, zugewiesen. Dadurch werden scharfe Kanten wiederhergestellt, da die Beeinflussung von Liniensegmenten und Auftreffpunkten in den Clustern ausgeglichen wird.

Die Unabhängigkeitsannahme von Zellen wird nahe den Grenzen von Clustern verletzt, da diese disjunkt sind und somit die Korrelationen von nahen Trainingsdaten ignoriert werden. Dies führt zu einem Diskontinuitäts-Problem, da Vorhersagen nicht den Lokalschätzern

entsprechen. Hierzu werden die Trainingsdaten mit überlappenden Boxen partitioniert und die Entfernungen der Cluster ausgeweitet.

3.2.2 Multi-Resolution-Surfel-Maps für effiziente 3D-Modellierung und Verfolgung

Bei der Methode basierend auf Multi-Resolution-Surfel-Maps, welche in [79] vorgestellt wird, handelt es sich um ein sehr rechen-effizientes Mappingverfahren, welches somit niedrige Anforderungen an die Hardware stellt. Viele Methoden der 3D-Registrierung rechnen auf GPUs, um eine ausreichende Bildwiederholrate zu erzielen. Im Vergleich zu CPUs welche für diese Methode ausreichend sind, sind GPUs teurer und das Systemgewicht höher. Für Anwendungen welche leicht oder günstig sein sollen eignet sich dieses Verfahren besonders. Als ein Beispiel sind Flugroboter zu nennen, bei denen das Gewicht eine große Rolle spielt.

Als Datenstruktur werden bei diesem Verfahren Octrees gewählt. Die Daten werden in dieser Baumstruktur integriert, wobei jede Ebene eine Auflösung repräsentiert. Die Wurzel enthält den gesamten Raum als ein Objekt (minimale Auflösung), die Blätter nur je ein Voxel (maximale Auflösung). Als Daten stehen von einem Sensor aufgenommene RGB-D-Bilder[1] zur Verfügung. Aus diesen Bildern werden beim Mapping die Farb-, Textur- und Tiefeninformationen speziell gespeichert. In jedem Knoten des Baumes werden Statistiken zur räumlichen Verbindung und Farbverteilung der Punkte innerhalb des Volumens gespeichert, welches als Surfel bezeichnet wird (Surfels wurden in Kapitel 2 bei Vorstellung der Surfel Maps genauer beschrieben). Diese Informationen sind redundant für jede Auflösung enthalten. Dies resultiert in einem erhöhten Speicheraufwand, jedoch auch in einer schnellen Verarbeitung auf jeder Auflösungsebene.

Die Farbverteilung sowie räumliche Verbindung der Punkte innerhalb eines Surfels (beinhaltet alle Kind-Knoten) wird als normalverteilt modelliert. Dabei wird hier ein Single-Pass-Update-Schema mit hoher numerischer Präzision [14] verwendet. Im Vergleich zu Multi-Pass-Update-Schemata arbeitet dieses sehr schnell und dennoch mit ausreichender Präzision.

Um ein neu aufgenommenes Bild in die bisherige Karte zu integrieren, wird es mit dem zuvor aufgenommenen verglichen. Dazu werden die Surfels der beiden Bilder gegenüber gestellt. Um diesen Vergleich effizient durchführen zu können, wird der „Shape Texture Descriptor“, ähnlich den FPFH-Features aus [68] eingeführt, welcher lokale Merkmale der Surfels repräsentiert. Für jedes Surfel wird ein solcher Deskriptor erstellt, welcher die Form und Textur in der Umgebung des Surfels (bis zu 26 Nachbarn) beschreibt. Die Zeiger zu Nachbarn werden in jedem Knoten gespeichert. In Abbildung 3.2 ist ein Deskriptor in 2D

[1] RGB-D Bilder sind Farbbilder, wobei die Pixel neben den RGB-Farbinformationen ebenfalls Tiefeninformationen enthalten

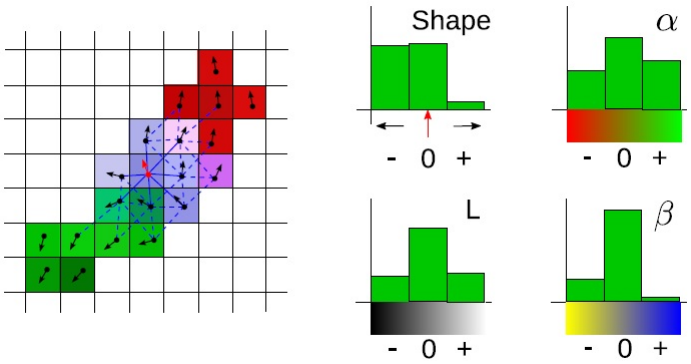


Abbildung 3.2: Aus [79]. 2D-Darstellung des Shape Texture Descriptors. Form (Shape), Chrominanz (α, β) und Luminanz (L) werden beschrieben, um damit Surfels besser assoziieren zu können. Jeder Knoten beinhaltet einen solchen Deskriptor, welcher die Daten mit bis zu 26 Nachbarn beschreibt.

visualisiert. Durch diese dann bereits vorhandenen Deskriptoren kann der spätere Vergleich von Surfels verschiedener Scans schneller ablaufen. Aus allen Surfels, für welche eine Korrespondenz im vorherigen Bild gefunden wurde und sie damit in die Karte integriert werden konnten, wird daraus die Pose des Sensors berechnet. Surfels für die keine Übereinstimmung gefunden werden konnte, werden mit der aus der Pose berechneten Transformation in die Karte integriert.

Bei dem Einfügen von neuen Punkten in die bestehende Karte ist folgendes anzumerken: Eine einfache Implementierung für das Update der Karte wäre das Hinzufügen der ausreichenden Statistiken eines Punktes in den Baum, beginnend am Wurzelknoten (inkrementelles Update). Dies ist jedoch entsprechend rechenaufwendig. Es können physikalische Eigenschaften der Sensoren ausgenutzt werden, um ohne Qualitätsverlust weniger Knoten des Baumes aktualisieren zu müssen. Das Rauschen nimmt quadratisch mit der Entfernung gemessener Punkte zum Sensor zu. Die Sensorauflösung in der Entfernung ist dementsprechend geringer. Wird die Octree-Knoten-Größe auf das Produkt der Pixelgröße und der quadratischen Entfernung zum Sensor reduziert, werden nur Auflösungsebenen der Karte aktualisiert, welche auch ausreichend vom Sensor erfasst werden konnten. Dadurch kann der Rechenaufwand deutlich reduziert werden.

3.2.3 KinectFusion

Das in [62] vorgestellte Verfahren ermöglicht ein genaues Echtzeit-Mapping von beliebigen und komplexen Indoor-Szenarien mit dem Microsoft-Kinect-Sensor der Version 1. Das Verfahren ist zwar speziell für den Kinect-Sensor entwickelt worden, kann jedoch prinzipiell auch auf die Rohdaten anderer Tiefensensoren angewandt werden.

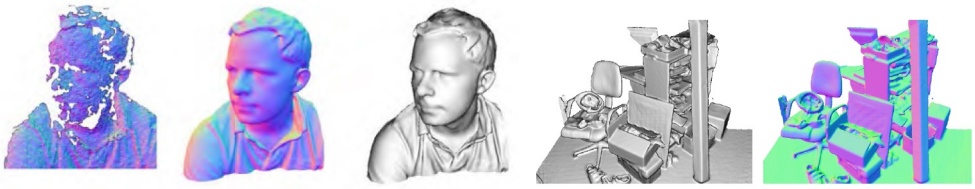


Abbildung 3.3: Aus [62]. Links zu sehen ist ein unbearbeitetes Tiefenbild der Kinect, welches als Input in das vorgestellte System dient. Nach Bearbeitung mit der hier vorgestellten Methode entstehen Normalenkarten (2. v.l., 1. v.r.) sowie Phang-Shaded-Darstellungen (3. v.l., 2. v.r.).

3D-Modelle können bei diesem Verfahren zum einen dadurch generiert werden, dass der Sensor, z.B. auf einem Roboter, durch den zu vermessenden Raum bewegt wird. Zum anderen, dies ist insbesondere für kleine zu vermessende Objekte möglich, dadurch, dass Objekte vor dem Sensor so bewegt werden, dass dieser das gesamte Objekt erfassen kann.

In Abbildung 3.3 sind die aus dem Verfahren resultierenden Karten zweier Beispielszenarien dargestellt. Das linke Bild der Abbildung stellt das Tiefenbild einer Kinect-Kamera dar, welches mit dem hier im Folgenden vorgestellten Verfahren verarbeitet wird. Es entsteht dabei eine Normalenkarte (2. v.l.) sowie eine Phang-Shaded-Darstellung (3. v.l.). Die beiden rechten Bilder zeigen die entstehenden Darstellungen für eine andere Szenerie, deren Tiefenbild nicht dargestellt ist.

Der Ablauf des Verfahrens lässt sich in folgende Schritte unterteilen, deren Abhängigkeiten in Abbildung 3.4 dargestellt sind.

- Oberflächenmessung

  Bei diesem Schritt werden aus den von dem Kinect-Sensor generierten Rohdaten eine Punktdichte- sowie eine Normalenkarte erzeugt.

  Dazu wird zunächst ein bilateraler Filter [86] auf die Rohtiefenkarte angewandt, um eine Discontinuity-Preserved-Tiefenkarte mit reduziertem Rauschen zu erhalten. Durch eine Rückprojektion der gefilterten Tiefenwerte in das Sensorframe entsteht die Punktdichtekarte. Da jedes Bild des Tiefensensors eine Oberflächenmessung auf einem gleichmäßigem Gitter[2] ist, können die Normalenvektoren über das Kreuzprodukt zwischen benachbarten Kartenpunkten berechnet werden.

- Oberflächenvorhersage

  Die Schleife zwischen Mapping und Lokalisierung wird geschlossen. Dies wird durch Raycasting der TSDF in das vorhergesagte Bild erreicht. Bei dem Raycasting wird

[2] Ein regelmäßiges Gitter unterteilt den Raum vollständig in achsenparallele rechtwinklige Bereiche, wobei Kanten entlang einer Achse stets die gleiche Länge besitzen.

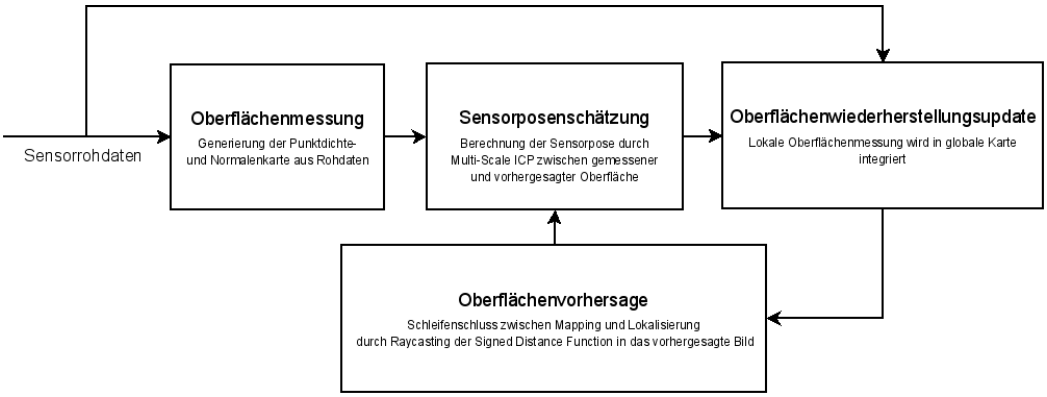


Abbildung 3.4: Schematischer Ablauf des Echtzeit-Mappings.

für jedes Pixel des zu berechnenden Bildes ein Strahl erstellt, welcher durch das Volumen verläuft. In regelmäßigen Abständen werden die Farb- und Opazitätswerte erfasst. Anschließend werden diese Werte kombiniert und das aus dem Strahl resultierende Pixel in der Bildebene errechnet. Durch diese Oberflächenvorhersage kann die aktuelle Tiefenkarte entsprechend ausgerichtet werden.

- Sensorposenschätzung

  Die Live-Sensor-Verfolgung wird durch eine Multi-Scale-ICP-Ausrichtung zwischen der vorhergesagten Oberfläche und der aktuell durch den Sensor gemessenen bewerkstelligt, wobei dazu nicht alle Daten des Tiefenbildes genutzt werden. Durch eine hohe Bildwiederholrate kann eine nur geringe Bewegung von einem Bild zum nächsten angenommen werden. Somit kann der schnelle Projective-Data-Association-Algorithmus nach [8] verwendet werden, um die Korrespondenz sowie die Metrik der Punktebene aus [16] zu erhalten. Durch die voll parallelisierte Verarbeitung moderner GPUs kann die Datenassoziierung und Punkt-Ebene-Optimierung alle verfügbaren Oberflächenmessungen benutzen, ohne dabei den Prozess durch lange Rechenzeiten merklich zu verzögern.

- Oberflächenwiederherstellungsupdate

  Die Pose des Roboters muss dazu bekannt sein. Hier werden die Daten der Oberflächenmessung in das bestehende globale Modell integriert. Dies geschieht mithilfe der Truncated Signed Distance Function (TSDF) [18]. Diese Distanzfunktionen geben für jeden Punkt die Distanz zur nächsten Oberfläche an. Positive Werte beschreiben Punkte im freien Bereich, negative Werte hingegen Punkte auf der nicht sichtbaren Seite der Oberfläche. Abbildung 3.5 zeigt ein Beispiel.

Falls der Sensor nicht lokalisiert werden kann, gibt es ein Verfahren zu dessen Wiederlokalisierung. Dazu wird die zuletzt bekannte Pose benutzt, um eine Oberflächenschätzung

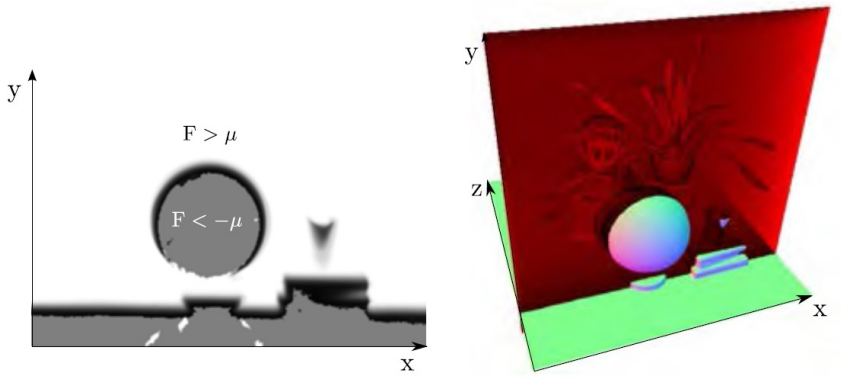


Abbildung 3.5: Visualisierung der Truncated-Signed-Distance-Funktion aus [62]. Hier ist ein Schnitt durch ein Truncated-Signed-Distance-Volumen mit der Tuncated-Funktion F dargestellt. Punkte innerhalb des Volumens ($F > \mu$) sind grau dargestellt, Punkte an der Oberfläche (F = 0) in schwarz und Punkte im freien Bereich außerhalb des Volumens ($F < \mu$) weiß.

darzustellen. Der Benutzer wird dann aufgefordert, das aufgenommene Tiefenbild manuell mit dem angezeigten auszurichten. Sobald sich der Sensor wieder erfolgreich lokalisieren kann, wird das Mapping automatisch fortgeführt.

3.2.4 Volumetrisches 3D-Mapping in Echtzeit auf einer CPU

In [76] wird ein neuer Algorithmus vorgestellt, welcher sich mit den Abhängigkeiten zwischen Nachbarzellen unterschiedlicher Auflösung beschäftigt. Das Ziel ist ein effizientes Update dieser entsprechenden Bereiche des Dreiecksnetzes. Der Algorithmus ist dabei so gewählt, dass der Rechenaufwand gering gehalten wird und somit auf einer handelsüblichen CPU ausreichend schnell ausgeführt werden kann.

Da durch das Scannen verschiedene Objekte aus unterschiedlicher Entfernung vermessen worden sind, werden die Geometrie-Informationen in verschiedenen Auflösungen gespeichert, um die Sensorauflösung direkt abbilden zu können. Zur Repräsentation der Geometrie wird dabei die Truncated-Signed-Distance-Funktion verwendet, wie sie in Abschnitt 3.2.3 bereits beschrieben wurde. Weiter entfernte Objekte werden in geringerer Auflösung in der Datenstruktur (Octree) integriert, als nähere. Dies begründet sich dadurch, dass aufgrund der physikalischen Eigenschaften der Sensoren die Auflösung der erfassten Pixel quadratisch mit der Entfernung abnimmt. Gespeichert wird die Geometrie-Information in „Bricks“, welche aus je 8^3 Voxeln bestehen. Ein Voxel wiederum speichert den Wert der SDF, ein Gewicht sowie die Farbe. Die Größe der Bricks und der darin enthaltenen Voxel ist variabel. Bei höherer Auflösung sind die Bricks und Voxel entsprechend kleiner, wobei ein Brick stets 8^3 Voxel beinhaltet. Ein Brick enthält bis zu acht weitere

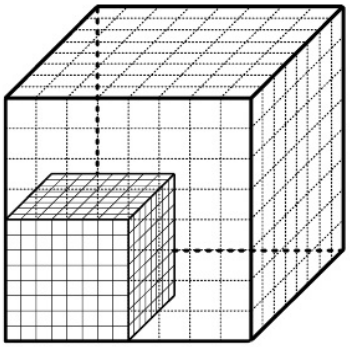


Abbildung 3.6: Aus [76]. Brick, welcher einen weiteren Brick beinhaltet. Jeder Brick besteht aus 8^3 Voxeln.

Bricks, bis die maximale Auflösung erreicht ist und keine weitere Auflösungsebene, repräsentiert durch eine weitere Brickunterteilung, vorhanden ist. In Abbildung 3.6 ist ein Brick, welcher einen weiteren Brick beinhaltet, visualisiert. Der Raum ist dabei innerhalb des äußeren Bricks nur an der Stelle des inneren Bricks genauer aufgelöst.

Als Datenstruktur werden Octrees verwendet, welche die Daten in einer Baumstruktur speichern. Jeder Baum beinhaltet die Position in Bezug auf den globalen Koordinatenursprung, die Skalierung sowie bis zu acht Zweige. Jeder Zweig beinhaltet einen Brick und bis zu acht weitere Zweige. Jeder Brick ist also in der Baumstruktur einem Zweig zugeordnet, sodass die räumlichen Abhängigkeiten der Bricks effizient abgefragt werden können.

Um die Geometrie zusammenzufügen, wird der Ansatz der Truncated Signed Distance Function von [18] benutzt, welcher auf die hier benutzte Datenstruktur angepasst wurde. Zuerst wird für jedes Pixel im Bild die dreidimensionale Position im Raum bestimmt und der entsprechende Brick gesucht, in dem der Punkt dann gespeichert wird. Alle Bricks, die einen Punkt der aktuellen Tiefenkarte beinhalten, werden anschließend verarbeitet. Dabei werden die Distanz-Werte in jedem Brick neu berechnet.

Für das anschließende Meshing wird der Marching-Cubes-Algoritmus für Nulllevel-Extraktion [52] angewandt. Dieser Algorithmus ist einfach auf ein regelmäßiges Gitter anzuwenden. Für die Verwendung mit der hier verwendete Struktur mit Bricks sind jedoch einige Anpassungen nötig, welche in [76] näher beschrieben sind.

3.2.5 Fotorealistisches 3D-Mapping von Innenräumen mit RGB-D-Scanning-Prozessen

Bei der in [87] vorgestellten Methode entsteht ein fotorealistisches Modell der Umgebung.

Zunächst muss ein RGB-D Video von dem Raum, welcher modelliert werden soll, aufgenommen werden. Dies kann durch einen mobilen Roboter, als auch durch eine natürliche Person geschehen. Nachdem das Video erstellt ist, werden mit dem Video als Input alle folgenden Schritte automatisch durch das hier beschriebene Verfahren durchgeführt.

Zuerst wird die 3D-Trajektorie, wie in [88] beschrieben, durch ein RGB-D Tracking generiert. Es können durch Kenntnis der Trajektorie den Bildern des Videos die jeweiligen Sensorposen zugeordnet werden. Es werden dabei „Keyframes“ ausgewählt, wenn ein zuvor festgelegter Winkel oder eine translatorische Distanz zu dem existierenden Modell besteht. Keyframes sind aus dem Video extrahierte RGB-Bilder zu bestimmten Zeitpunkten. Eine mögliche Keyframe-Auswahl ist in Abbildung 3.7 dargestellt. Diese Keyframes repräsentieren nicht alle vorhandenen Informationen, da als Keyframes nur einige, wenige Bilder des Videos ausgewählt werden, um speichereffizient zu arbeiten. Es werden Punktwolken, zu denen nicht bereits das zugehörige RGB-Bild als Keyframes gewählt wurde, auf das nächste Keyframe gewarped.

Um die Präzision der entstandenen Karte anschließend zu erhöhen, wird ein Filter angewandt, welcher noch existierende Löcher schließt. Anschließend wird die Punktwolke, mithilfe der Poisson-Methode aus [41], in ein Polygonnetz konvertiert, wobei Rauschen sowie fehlende Daten, speziell berücksichtigt werden. Ein Polygonnetz hat gegenüber Punktwolken den Vorteil, weniger Speicher zu benötigen und besser von Standard-3D-Modellierungsprogrammen unterstützt zu werden. Die Polygone werden auf alle Keyframe-Ansichten projiziert und Texturkoordinaten generiert, wie in Abbildung 3.8 zu sehen ist. Abschließend werden die Keyframes verbunden. Dabei können Farbstreifenbildungseffekte auftreten, falls die Helligkeit über das Bild variiert. Ein geeigneter Filter, um diese Effekte zu reduzieren ist der Laplacian-Pyramid-Filter aus [81].

3.3 Verwendbarkeit von Methoden der analysierten Verfahren

In dem vorgestellten Verfahren aus Abschnitt 3.2.1 wird ein Prozess erläutert, mit dem Zellen in Occupancy Maps, welche nicht direkt von einem Sensorstrahl getroffen wurden (benachbarte Zellen), basierend auf Gauss-Prozessen aktualisiert werden. Durch die hier vorgestellte Methode wird der prinzipiell dabei sehr hohe Rechenaufwand bei der Anwendung der Gaus-Prozesse reduziert. Diese Aktualisierung von Nachbarzellen, welche im dem dieser Arbeit zu Grunde liegenden Verfahren nicht implementiert ist, wäre eine Möglichkeit die Qualität der Karte zu verbessern.

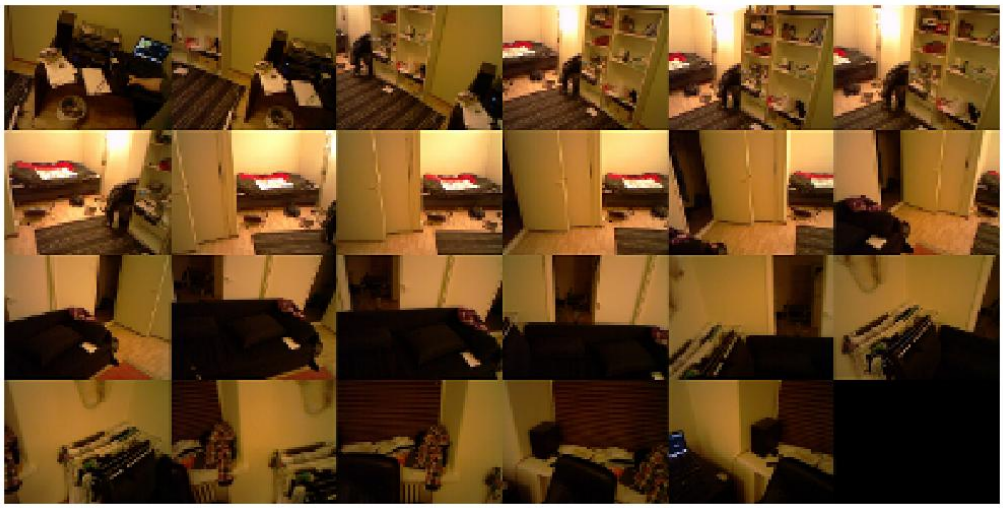


Abbildung 3.7: Aus [87]. Extrahierte Keyframe-Bilder des Videos, gespeichert in einer großen Textur.

Das Verfahren aus Abschnitt 3.2.2 verwendet RGB-D Bilder als Eingabedaten. Kern des Algorithmus ist das Arbeiten mit Deskriptoren, welche Informationen zur Farbverteilung und Textur beinhalten. Diese Informationen lassen sich mit den in dieser Arbeit von Tiefensensoren erzeugten Tiefenbildern nicht ermitteln. Durch die fehlenden RGB-Daten ist dieses Verfahren daher nicht sinnvoll zu adaptieren.

In Abschnitt 3.2.3 wird ein Verfahren vorgestellt, welches ein Echtzeit-Mapping unter Verwendung des Kinect-Sensors beschreibt. Dabei liegt der Hauptaspekt der Arbeit auf der Sensorposenschätzung mittels eines Multi-Scale-ICP. Es entstehen dabei Normalenkarten sowie Phang-Shaded-Darstellungen. In dieser Arbeit wird die Pose durch einen bereits implementierten SLAM-Algorithmus bestimmt. Die Daten sollen außerdem in einer Tiefenkarte basierend auf Octrees gespeichert werden. Deshalb ist das Verfahren für diesen Zweck nicht geeignet.

Das Verfahren in Abschnitt 3.2.4 beschäftigt sich mit der Abhängigkeit von Nachbarzellen unterschiedlicher Auflösung, zum Zweck eines effizienten Updates der entsprechenden Bereiche in einem Dreiecksnetz. Das bisher verwendete Verfahren verwendet kein Dreiecksnetz. Dieses müsste mit implementiert werden, um anschließend das effiziente Update-Verfahren testen zu können. Die hier vorgestellte Methode ist somit nicht direkt geeignet.

Bei dem in Abschnitt 3.2.5 erläuterten Verfahren wird aus einem zuvor aufgenommenen RGB-D Video eines Raumes, dieser entsprechend modelliert. Dabei entsteht ein fotorealistisches Modell dadurch, dass ausgewählte RGB-Bilder auf das zuvor berechnete Netz

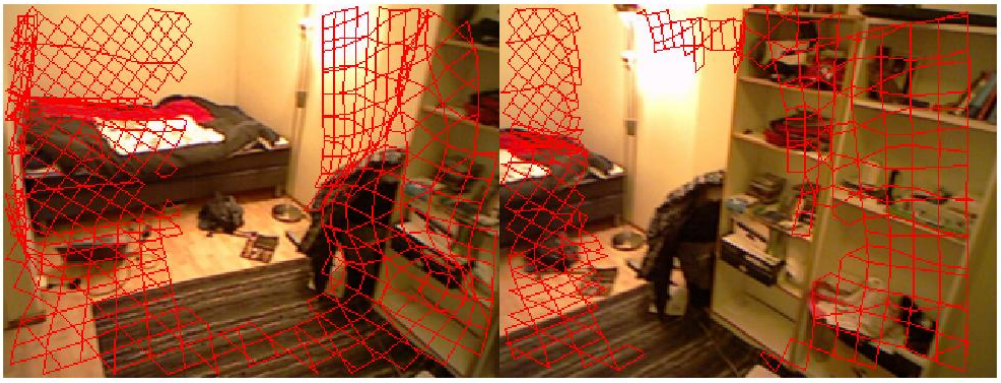


Abbildung 3.8: Aus [87]. Auf Keyframe-Bilder projizierte Poisson-Polygone.

projiziert werden. In dieser Arbeit kann jedoch nicht zuvor ein erforderliches RGB-D Video aufgenommen werden, da ein Roboter einen unbekannten Raum explorieren können muss und sich dabei nur in bereits als kollisionsfrei bestimmten Raum bewegen darf. Des weiteren sind keine RGB-Daten verfügbar.

Bei den Verfahren in den Abschnitten 3.2.2 und 3.2.4 werden Multi-Resolution-Maps verwendet. Informationen werden in verschiedenen Auflösungsebenen gespeichert. Da aufgrund der physikalischen Eigenschaften der Sensoren die Auflösung der erfassten Pixel, laut den jeweiligen Verfahrensbeschreibungen, quadratisch mit der Entfernung abnimmt, werden auch die Daten in der Karte in entsprechender Auflösung gespeichert. Somit wird Speicherplatz gespart und ein Upsampling, von für die Datenstruktur zu gering aufgelösten Sensordaten, muss nicht durchgeführt werden.

Derzeit werden, bei dem in dieser Arbeit zu erweiternden Verfahren, alle Messdaten auf Voxelebene integriert. Es wird also ausschließlich auf höchster Auflösungsebene, bzgl. der Integration neuer Messdaten, gearbeitet. Durch die Verwendung einer einer Karte mit multiplen Auflösungsstufen könnte das aktuelle Verfahren hinsichtlich Geschwindigkeit und Speicherbedarf verbessert werden, da dadurch Daten auch teilweise in höherer, als der maximalen, Auflösung gespeichert würden. Diese Idee wird im weiteren verfolgt und entsprechend der speziellen Einsatzzwecke am Institut ausgearbeitet. Einsatzzweck ist dabei vor allem die Exploration von unbekannten Innenräumen auf einem mobilen Roboter.

3.4 Entwicklungsidee und deren damit angedachte Verbesserungen

Bei der Integration von Messdaten kann die Qualität der Daten durch verschiedene Verfahren berücksichtigt werden. Hier soll die Qualität der Daten über die Entfernung der Mess-

punkte zum Sensor approximiert werden. Je weiter die Messpunkte vom aufnehmenden Sensor entfernt sind, desto geringer wird die Qualität angenommen. Der Verlauf der Abnahme der Qualität mit der Entfernung, mit dem in dieser Arbeit verwendeten Sensor, soll dabei näher untersucht werden.

Die Idee ist, ein dynamisches Wachstum der Auflösung der Karte zu erreichen. Qualitativ höherwertige Daten, welche mit geringerer Distanz zum Sensor aufgenommen wurden, werden in höherer Auflösung gespeichert, als weiter entfernte Daten. Bei der Exploration mit einem mobilen Roboters werden weit entfernte Objekte zunächst in geringer Auflösung gemessen und in die Karte integriert. Im späteren Verlauf können diese dann, wenn der Roboter diese von näherem erneut vermisst, in höherer Auflösung erfasst werden. Diese qualitativ höherwertigen Daten derselben Gebiete werden dann entsprechend integriert. So ist es möglich hoch aufgelöste Gebiete für solche, die aus qualitativ hochwertigen Messdaten erstellt wurden, in der Karte zu erhalten, ohne, dass die gesamte Karte in dieser hohen Auflösung erstellt werden muss.

Ein Beispielszenario ist die Aufgabe einen Roboter zu einem Regal fahren und einen bestimmten Gegenstand daraus greifen zu lassen. Für die Navigation des Roboters zu dem Regal ist keine allzu hohe Auflösung notwendig. Insbesondere weit entfernte Gebiete, welche keine Kollisionsgefahr für den Roboter darstellen, sind dabei prinzipiell uninteressant. Für das Identifizieren und vor allem das Greifen des Gegenstandes mithilfe am Greifarm angebrachter Tiefensensoren ist wiederum eine sehr hohe Auflösung von Nöten. Durch ein dynamisches Wachstum der Auflösung um den Sensor am Greifarm des Roboters kann die Karte im Nahbereich dieses Sensors sehr hoch aufgelöst werden und dabei die gesamte Karte in überschaubarer Größe gehalten werden. Bisher müsste bei dem verwendeten Verfahren die gesamte Karte auf der maximal benötigten Auflösung erstellt werden, was zu extrem hohen Rechen- und Speicheraufwand führen würde.

Auch ganz allgemein, bei der während einer Exploration nötigen Kollisionsvermeidung, ist dieses Prinzip des Kartenbaus sinnvoll. Zur groben Orientierung, z.B. um Durchgänge zu finden, werden diese weit entfernten Objekte in geringer Auflösung vermessen. Um dann durch einen (engen) Durchgang zu navigieren oder nah an Hindernisse heranzufahren, ist eine höhere Auflösung nötig, welche dann automatisch durch den geringeren Abstand der Sensoren zu dem Hindernis, gegeben ist.

4 Geeignete Tiefensensoren

4.1 Arten von Tiefensensoren

Durch spezielle Techniken mit verschiedenen Sensoren können Entfernungen zu Raumpunkten bestimmt und daraus Tiefenkarten erstellt werden. Ein Überblick über verschiedene Techniken der Tiefenmessung zur 3D-Kartenerstellung ist in Abbildung 4.1 gegeben, welche im Folgenden näher erläutert werden.

Tiefensensoren sind prinzipiell, im Gegensatz zu RGB-Sensoren, welche im wesentlichen nur aus einem Objektiv und Sensorchip bestehen, ein komplexes System. Die Fehleranfälligkeit durch die Verwendung diverser Komponenten, welche miteinander kommunizieren, mit ihren jeweiligen durch den technischen Fertigungsprozess bedingten Toleranzen, ist ungleich höher. Eine genaue Analyse sowie Modellierung und Kalibrierung von 3D-Tiefensensoren wird in [28] durchgeführt.

4.1.1 Time-of-Flight-Kameras

Time-of-Flight-Kameras, welche in [58] ausführlich beschrieben werden, sind Kamerasysteme, welche Distanzen über die Signallaufzeiten messen. Die Szenerie wird dazu mit modulierten Lichtpulsen, welche meist im nahen Infrarotbereich liegen [51], ausgeleuchtet und die Reflexion von der Sensorik erfasst. Da die Laufzeit proportional zur Distanz ist, können die Entfernungen zu den Reflexionspunkten bestimmt werden. Die Laufzeiten können entweder direkt durch Zeitmessung eines Pulses, oder indirekt über die Phasenverschiebung bestimmt werden.

Aufgrund der Ausbreitung der Pulse mit Lichtgeschwindigkeit und der damit verbundenen kurzen Signallaufzeit, ist die direkte Messung, dargestellt in Abbildung 4.2(a), für kurze Distanzen, wie sie beispielsweise bei Indoor-Szenarien auftreten, sehr aufwendig zu realisieren. Bei der direkten Messung der Laufzeiten von Pulsen, muss der Puls so kurz sein, dass die Reflexion erst empfangen wird, wenn der Puls nicht mehr gesendet wird um eine Überlappung auszuschließen.

Technisch einfacher, wie in Abbildung 4.2(b) dargestellt, ist es, zu einem festen Zeitpunkt die Phasenverschiebung zu messen. Deshalb werden bei der Countinous-Wave-Methode die Signale moduliert (meist als Rechteck-Signal, da diese mit digitalen Schaltungen leicht

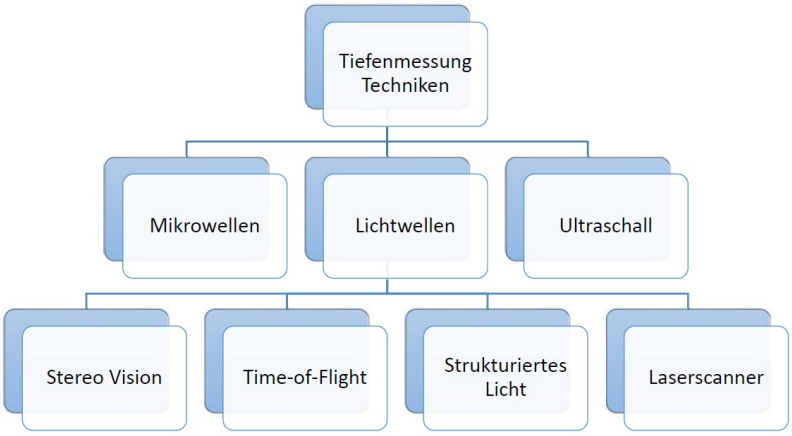


Abbildung 4.1: Übersicht über verschiedene Techniken der Tiefensensorik.

realisiert werden können [58]) und die Phasenverschiebung zwischen dem gesendeten und empfangenen Signal bestimmt.

Nach Gleichung 4.1 - 4.4 kann über die Kreuzkorrelation c des empfangenen Signals $r(t)$ und dem gesendeten Signal $s(t)$, mit konstantem Bias b, Phasenverschiebung φ, Dämpfung a, Modulationsfrequenz ω und Offset τ die Entfernung d berechnet werden.

$$r(t) = b + a \cdot \cos(\omega t + \phi) \tag{4.1}$$

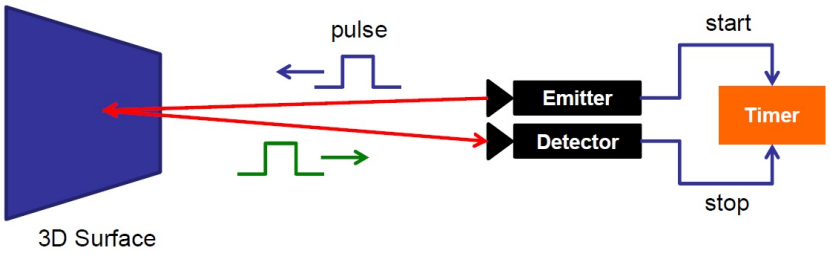
$$s(t) = \cos(\omega t) \tag{4.2}$$

$$c(\tau) = \int_{-\infty}^{\infty} s(t) \cdot g(t + \tau) dt = \frac{a}{2} \cos(\omega\tau + \phi) + b \tag{4.3}$$

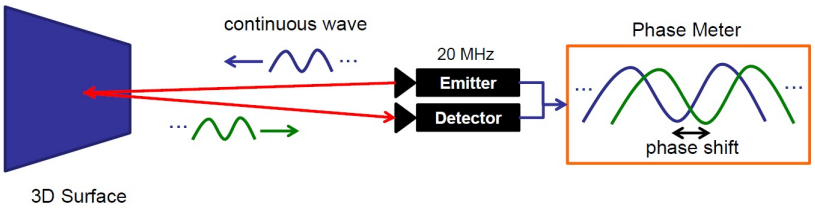
$$d = \frac{c}{4\varpi\omega}\phi \tag{4.4}$$

Je größer die Entfernung, desto länger ist ein gesendetes Signal unterwegs und die Phasenverschiebung des aktuell gesendeten Signals fortgeschritten. Die berechneten Tiefendaten werden über die Auflösung der Sensorik in Pixel quantisiert.

ToF-Kameras bieten u.A. den Vorteil, dass keine manuelle Tiefenberechnung erforderlich ist und durch die hoch energetischen Lichtpulse keine hohen Ansprüche an die Beleuchtung der Szenerie bzgl. eventueller Störeinflüsse bestehen. Die Systeme sind jedoch komplex aufgebaut und die Fehleranfälligkeit, bedingt durch technische Toleranzen, relativ hoch. Bei dem Continous-Wave-Verfahren kommt hinzu, dass durch die benötigte Integrationszeit bei der Bestimmung der Phasenverschiebung die Bildwiederholrate eingeschränkt wird.



(a) Puls Verfahren



(b) Coutinous Wave Verfahren

Abbildung 4.2: Aus [91]. Entfernungsmessungsverfahren von ToF-Kameras.

Eine sehr bekannte und häufig verwendete ToF-Kamera ist der Kinect-Sensor in der Version 2. Die Spezifikationen des Sensors sind in Tabelle 4.1.1 dargestellt. Zum Vergleich sind die Spezifikationen der ersten Version, welche in wissenschaftlichen Anwendung häufig verwendet wird, aufgeführt. Bei der ersten Version handelt es sich um einen Sensor, der mit strukturiertem Licht arbeitet, nicht um eine ToF-Kamera.

4.1.2 Stereo Vision

Stereo Vision benutzt zwei Kameras, welche in einem fest definierten Abstand zueinander stehen. Objekte mit ausreichend differenzierter Fläche, sodass Korrespondenzen zwischen Bildern der beiden Kameras gefunden werden können, besitzen eine messbare Disparität. Diese Disparität kann dahingehend weiter verarbeitet werden, dass die Entfernung von Objekten zum Kamerasystem bestimmt werden kann. Ein großer Vorteil von Stereo Vision ist es, dass dabei sehr günstige RGB-Kameras oder sogar Graustufen-Kameras verwendet werden können. Nachteile sind, dass die Szenerie ausreichend beleuchtet sein muss und die Objekte eine ausreichend differenzierbare Textur besitzen müssen. Außerdem ist die Verarbeitung sehr rechenaufwendig.

In Abbildung 4.1.2 ist beispielhaft eine Kameraanordnung dargestellt. Die Kameras stehen in einer Ebene mit gleicher Orientierung in einem festen Abstand x zueinander. Aus

Tabelle 4.1: Microsoft-Kinect-Sensor-Spezifikationen [5, 4]

Kinect-Baureihe	**Version 1**	**Version 2**
Max. Auflösung der RGB-Kamera	1280x960	1920x1080
Bildwiederholrate (bei max. Auflösung)	30	30
Max. Auflösung des Tiefensensors	640 x 480	512 x 424
Tiefenbereich	0,40 m - 4,0 m	0,40 m - 4,5 m
Sichtfeld	43° x 57°	70° x 60°

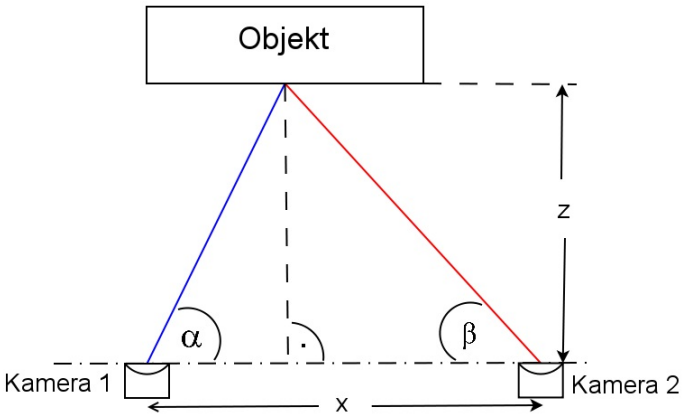
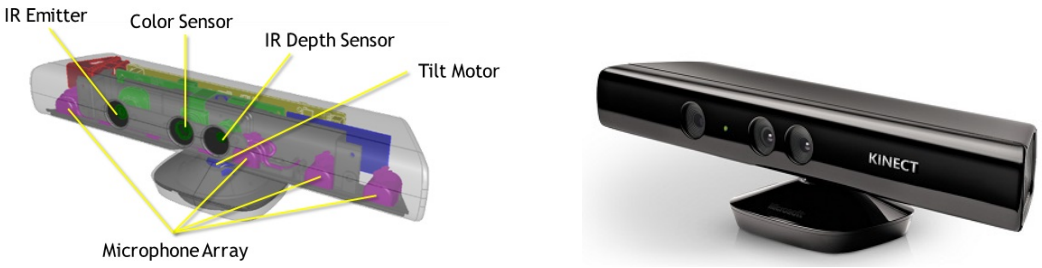


Abbildung 4.3: Stereokamera-System, welches über die Disparität zwischen den Bildern der beiden Kameras die Winkel α und β bestimmt, woraus die Entfernung des Objekts z berechnet werden kann.

der Disparität der Bilder der beiden Kameras können die Winkel α und β bestimmt werden. Nach Gleichung 4.5 kann aus diesen Daten die kürzeste Entfernung zur Ebene, auf der sich die Kameras befinden, berechnet werden. Die Entfernung des Objekts zu den jeweiligen Kameras kann über die geometrischen Zusammenhänge berechnet werden.

Verwendung bei der Kartenerstellung in der mobilen Robotik findet die Technik u.A. in [61] oder zur Lokalisierung und Kartenerstellung für autonome Exploration in Außenbereichen in [2, 48].

$$z = \frac{x}{\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}} \tag{4.5}$$



(a) Schematische Ansicht (b) Reale Ansicht

Abbildung 4.4: Microsoft-Kinect Version-1, Bildquelle: Microsoft

4.1.3 Strukturiertes Licht

Bei diesem Verfahren wird ein bekanntes Muster von Pixeln auf eine Szenerie projiziert. Durch die Deformation des Musters bei Auftreffen auf Objekte können Tiefen- und Oberflächeninformationen gewonnen werden. Um ein Tiefenbild zu erhalten, müssen, während einer Einstellung, Projektionen kodiert oder phasenverschoben werden. Die Bildwiederholrate ist deshalb gering und der Sensor, sowie die zu vermessenden Objekte dürfen sich in der Zeit nicht bewegen. In [34] wird das Verfahren vorgestellt. Verschiedene Techniken von strukturiertem Licht werden in [29] verglichen.

Das wohl bekannteste und in der Wissenschaft meist verwendete System, welches nach diesem Prinzip arbeitet, ist die Kinect-Kamera in der Version 1, dargestellt in Abbildung 4.4. Die Spezifikationen wurden beim Vergleich der beiden Versionen 1 und 2 der Kinect-Sensoren, in Tabelle 4.1.1 angegeben.

4.1.4 Laserscanner

Laserscanner sind Sensoren, die einen Laserstrahl so ablenken, dass dieser Oberflächen rasterartig überläuft und über die Reflexionen ein entsprechendes Tiefenbild erstellt. Abbildende Laserscanner erfassen, neben der Geometrie, auch die Intensität des reflektierten Signals, wodurch Rückschlüsse auf die jeweiligen Materialeigenschaften möglich sind und im entstehenden Bild verschiedene Materialien unterschiedlicher Reflektivität visuell leicht zu unterscheiden sind.

Ein Laserscanner besteht aus einem Scankopf sowie einer Treiber- und Ansteuerelektronik. Es existieren dabei verschiedene Arten von Scanköpfen die im Folgenden nach [31, 30] kurz beschrieben werden.

- Spiegelscanner

  Die einfachste Methode ist die Ablenkung des Strahls durch rotierende Spiegel. Zur zweidimensionalen Ablenkung wird entweder ein Spiegel in zwei Richtungen bewegt, oder es werden zwei orthogonal drehbare Spiegel genutzt, die beide vom Laserstrahl durchlaufen werden.

- Prismenscanner

  Durch zwei axial drehbare Prismen, sogenannten Risley-Prismen, kann der Laserstrahl zweidimensional abgelenkt werden. Prismenscanner werden derzeit nur für wenige spezielle Anwendungen im militärischen Bereich eingesetzt.

- Akustooptische Deflektoren

  Bei dieser Methode wird der Laserstrahl durch einen akustooptischen Modulator mit Ultraschallwellen gebeugt.
  Ein akustooptischer Modulator ist ein optisches Bauelement, welches einfallendes Licht in Frequenz und Ausbreitungsrichtung beeinflusst. Hierzu wird in einem transparenten Festkörper mit Schallwellen ein optisches Gitter erzeugt, an welchem der Strahl gebeugt und in seiner Frequenz verschoben wird. An diesem Gitter wird der Lichtstrahl gebeugt und gleichzeitig in seiner Frequenz verschoben. Akustooptische Modulatoren, die zur Ablenkung des Lichts eingesetzt werden, werden auch Braggzellen genannt.

Es existieren außerdem Scanköpfe die neben der einstellbaren Ablenkung in zwei Ebenen noch eine einstellbare Optik für die Tiefenfokusierung besitzen.

4.1.5 Ultraschall-Tiefensensoren

Aktive Sonare (**So**und **N**avigation **A**nd **R**anging) senden Ultraschallwellen und empfangen deren Echo. Über die Laufzeiten, welche direkt proportional zur Entfernung sind, können Tiefenbilder berechnet werden. Die Bestimmung der Richtung der Echos erfolgt in der Regel dadurch, dass mehrere Empfänger an bekannten Positionen das Echo empfangen und über die Laufzeitunterschiede zwischen den Empfängern die Richtung berechnet werden kann. Nach Kenntnis von Richtung und Entfernung, können dann die Daten entsprechend angezeigt oder in eine Karte integriert werden.

Um nach dem Senden eines Signals nicht auf alle möglichen Reflexionen warten zu müssen, bevor das nächste Signal gesendet werden kann, gibt es Sender, die kontinuierlich frequenzmoduliert senden. Über die Frequenz der empfangenen Signale können diese

dem Zeitpunkt des Sendens zugeordnet werden. Es gibt auch weitere Methoden der Signalkodierung, welche nicht mit einer einfachen Frequenzmodulation arbeiten und die Signale speziell kodieren. Dies ist insbesondere von Vorteil, wenn die Beeinflussung durch (gezielte) Störsignale ausgeschlossen oder minimiert werden soll.

Sonar-basierte Mapping-Verfahren für mobile Roboter werden u.A. in [60, 24, 25] vorgestellt. Mit dem Problem verrauschte Daten mehrerer Sonar-Sensoren in eine Zellenbasierte Karte zu integrieren, beschäftigt sich [57]. Diese Verfahren stammen aus den Jahren 1985 - 1988. 2005 wurde in [90] ein Verfahren vorgestellt, welches sich mit der Lokalisierung kleiner Roboter in unbekanntem Gebiet und der Kartenerstellung aus Sonardaten beschäftigt. Aktuell wird in dieser Richtung für mobile Landroboter nicht viel geforscht, da es inzwischen dafür geeignetere Sensoren und Verfahren gibt.

Das Verfahren wird insbesondere im Wasser, oder allgemein in Flüssigkeiten angewandt, da sich Schall dort mit höherer Geschwindigkeit als in Luft ausbreitet und alle Methoden, die mit elektromagnetischen Wellen arbeiten, dort nicht funktionieren. Außerdem ist das Verfahren im Wasser deutlich verlustärmer als in Luft, insbesondere bei hohen Frequenzen [89]. Umfassende Informationen zu Sonaren in Unterwasser-Anwendungen sind in [3] zu finden.

Es sei angemerkt, dass es neben den aktiven Sonaren auch passive Sonare gibt, welche Signale nur empfangen, jedoch nicht senden. Sie können demnach keine Objekte orten und stellen keine Tiefensensorik dar.

4.1.6 Mikrowellen-Tiefensensoren

Es werden Mikrowellenimpluse ausgesendet und die Reflexionen gemessen. Die Impulse werden je nach Medium, durch welches sie sich bewegen, entsprechend stark gedämpft. Ist das Medium bekannt, so lässt sich dadurch die Dicke berechnen. So kann beispielsweise die innere Struktur eines reflektierenden Behältnisses, gefüllt mit einem homogenen Material bekannter spezifischer Dämpfung, vermessen werden.

Zur Kartenerstellung mithilfe eines mobilen Roboters finden diese Sensoren praktisch keine Anwendung. Diese Sensoren finden vor Allem in Bereichen der Geographie wissenschaftliche Anwendung, unter anderem in der Bestimmung der Schneedicke durch an Satelliten befindlicher Sensorik [55, 15]. Es muss dabei beachtet werden, dass nach [39] auch die Struktur des Schnees zu unterschiedlichen Dämpfungen führt und die Tiefeninformationen verfälscht. Diese Verfälschung durch die Struktur eines Materials ist auch auf andere Materialien übertragbar und muss stets beachtet werden.

4.2 Bewertung der Sensoren für die Verwendung mit dem Mapping-Verfahren

Von den vorgestellten Techniken eigenen sich, zum hier durchgeführten Mapping, manche besser als andere. Im Folgenden wird kurz auf die jeweiligen Verfahren eingegangen, aus welchen Gründen sie, für die Verwendung in dieser Arbeit, als geeignet bzw. ungeeignet erscheinen.

Sensoren, die mit *strukturiertem Licht* arbeiten, sind prinzipiell zu Mapping-Zwecken gut geeignet, da durch dieses Verfahren sehr hochwertige Modelle akquiriert werden können, obgleich die Kosten solcher Sensoren deutlich höher als beispielsweise bei ToF-Kameras ausfallen. Bei Verwendung dieser Technik ist der Störeinfluss von Beleuchtung jedoch von größter Bedeutung [35]. Durch Innenraumbeleuchtungen können für, zu diesem Zweck, ungünstig beleuchtete, zu vermessende Gebiete, teilweise keine verwertbaren Daten erfasst werden. Aufgrund dieser Störanfälligkeit werden diese Sensoren, für dieses allgemeine Mapping-Verfahren, nicht empfohlen.

Mikrowellen-Sensoren sind für Kartenerstellungszwecke der mobilen Robotik, wie zuvor beschrieben, gänzlich ungeeignet.

Bei *Ultraschall-Tiefensensoren* nimmt die Genauigkeit sehr stark mit der Entfernung ab und ist außerdem vom Messwinkel, der Temperatur und den Druckverhältnissen abhängig. Werden mehrere Sensoren verwendet, müssen alle auf anderen Frequenzen arbeiten, um eine gegenseitige Beeinflussung zu vermeiden. Die Störung durch Reflexionen des Schalls und unterschiedliche Dämpfungen verschiedener zu vermessener Objekte stellt ein großes Problem dar. Die maximale Auflösung aktueller Sensoren ist außerdem deutlich geringer als die von Laserscannern oder ToF-Kameras [82]. Aufgrund dieser Einschränkungen scheinen solche Sensoren weniger geeignet.

ToF-Kameras stellen ein kostengünstiges, kompaktes System zur 3D-Tiefenmessung dar. Die Distanzinformationen können, ohne großen Rechenaufwand, direkt aus dem vom Sensor aufgenommenen Tiefenbild extrahiert werden. Vor- und Nachteile von ToF-Sensoren sind in [6] ausführlich dargestellt.

Laserscanner arbeiten nach ähnlichem Prinzip, wie ToF-Kameras, allerdings sind dazu mechanische, bewegliche Teile nötig, um den Laserstrahl abzulenken. Dies macht sie teurer und langsamer, da die Bildpunkte seriell statt parallel erfasst werden. Sie bieten im Gegensatz zu ToF-Kameras den Vorteil, dass dadurch, dass immer nur ein Punkt beleuchtet wird, eine höhere Messgenauigkeit erreicht wird. Bei ToF-Kameras, welche einen größeren Bereich gleichzeitig beleuchten, erreicht das Licht, unter Umständen, durch multiple Reflexionen, Reflexionsobjekte auf unterschiedlichen Pfaden. Dadurch kann die gemessene Distanz größer als die tatsächliche sein. Verschiedene, oft in der Forschung verwendete,

Laser-Tiefensensoren werden bzgl. ihres Rauschens für Oberflächenmessungen in [67] untersucht.

Aus historischen Gründen ist der am Institut vorhandene „omniRob“ bereits acht ToF-Kameras vom Typ IFM O3D100 ausgestattet, welche zum Mapping verwendet werden. Dieses Setup soll für die hier durchgeführten Versuche auch verwendet werden. Es ist jedoch anzudenken, zukünftig die vorgestellten Kinect-Sensoren der Version 2 anstatt dessen zu verwenden. Es handelt sich dabei ebenfalls um ToF-Kameras. Sie bieten eine deutlich höhere Auflösung (512 x 424 im Vergleich zu 64 x 48 Pixel) und sind kostengünstig zu beziehen. Die Microsoft-Kinect-Sensoren wurden bzgl. ihrer Genauigkeit und Auflösung für Mapping von Innenräumen in [42] untersucht.

Stereokameras sind für Mapping, mit gewissen Einschränkungen, ebenfalls gut geeignet. Ein Vorteil ist es, dass günstige RGB-Kameras verwendet werden können und zusätzlich zu den errechneten Tiefendaten auch RGB-Bilder vorhanden sind. Als Einschränkung ist zu nennen, dass die Szenerie ausreichend beleuchtet sein muss und zu vermessende Objekte eine ausreichend differenzierbare Textur besitzen müssen, um die Tiefe bestimmen zu können. Die Verarbeitung ist außerdem sehr rechenaufwendig. In Abschnitt 6.4 wird das hier entwickelte Verfahren auch mit dieser Technik getestet.

5 Erweiterung des bestehenden Mapping-Verfahrens durch multiple Auflösungsstufen

5.1 Datenstruktur

Die Karte wird aus Octrees aufgebaut. Ein Octree ist ein gewurzelter Baum, dessen Knoten jeweils entweder acht Kinder oder gar keine Kinder haben. Sie werden hier benutzt um dreidimensionale Datensätze hierarchisch unterteilt zu speichern [72]. Für Teile der Karte, die Information enthalten sollen, werden Octrees erzeugt. Sie besitzen zur Repräsentation der Lage in der Karte einen Ankerpunkt[1] sowie eine zuvor definierte Größe, welche für alle Octrees der Karte gleich ist. Octrees beinhalten Octree-Elemente in welchen Daten gespeichert werden. Octree-Elemente sind würfelförmige, geometrische Strukturen zur Abbildung des Raumes, welche jeweils eine 1-Byte-Variable zur Speicherung der Besetzungswahrscheinlichkeit des Volumens des Octree-Elements, beinhalten. Die Größe der Octree-Elemente ist variabel.

Initial besteht ein Octree aus nur einem Octree-Element gleicher Größe, welches in kleinere Elemente zerlegt werden kann. Bei der Zerlegung eines Octree-Elements wird dieses stets in acht gleich große Unterelemente zerlegt. Dies ist solange möglich, bis ein Octree-Element einem Voxel entspricht, welches nicht weiter unterteilt werden kann. Die geometrische Anordnung von Octree-Elementen ist so vorgegeben, dass kleinere Octree-Elemente stets vollständig in größeren liegen. Es kann demnach kein Octree-Element geben, welches in mehreren Octree-Elementen einer höheren Größe liegt. Um die korrekte Zerlegung in jedem Fall zu gewährleisten, muss die Kantenlänge eines Octrees (entspricht dem initialem Octree-Element) dem 2^x-fachen, mit $x \in \mathbb{N}_0$, der Kantenlänge eines Voxels entsprechen. Die Größe eines zusammenhängenden Raumes, der Karte, ergibt sich damit in jeder Raumrichtung zu einem vielfachem einer Octree Kantenlänge. Bei dynamischer Erweiterung des Raumes werden, an den entsprechenden Stellen, weitere Octrees erzeugt.

[1] Die Koordinate eines Octrees oder Octree-Elements mit dem jeweils zahlenmäßig geringsten x-, y- und z-Wert

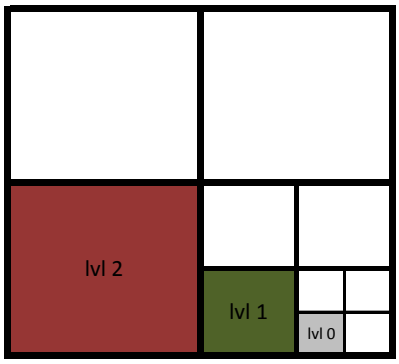


Abbildung 5.1: Octree-Element des Levels 3, welches Elemente niedrigeren Levels beinhaltet. Darstellung in 2D.

Den Octree-Elementen wird zur Bezeichnung je ein Level nach Ihrer Größe zugewiesen. Das geringste Level eines Octree-Elements wird als Level-0 bezeichnet und entspricht einem Voxel. Ein Octree-Element des Levels 1 besteht aus acht würfelförmig angeordneten Elementen des Levels 0, in diesem Fall Voxeln, ein Octree-Element des Levels 2 aus acht Elementen des Levels 1. Die Bezeichnung setzt sich, entsprechend dieser Vorschrift, für alle weiteren Level fort. Abbildung 5.1 zeigt dies für den 2D-Fall.

Die Unterteilung eines Octrees mit initialem Octree-Element, nach Änderung der Daten in einem Bereich, ist exemplarisch in Abbildung 5.2 für den 2D-Fall dargestellt. Das initiale Octree-Element wird, wenn sich die ursprünglich homogenen Daten verändern, so weit unterteilt, dass jedes Octree-Element Bereiche mit gleichen Daten zusammenfasst und dabei möglichst große Octree-Elemente bestehen bleiben. Ein Octree-Element wird also solange in acht gleich große kleinere Octree-Elemente unterteilt, bis die nötige Auflösung erreicht ist. Das kleinst mögliche Octree-Element entspricht der Größe eines Voxels.

Zunächst besteht der Octree aus einem Octree-Element identischer Größe mit dem Wert „127“. Dies entspricht dem Fall, dass alle Voxel den Wert „127“ besitzen. Nach einer Messung wird einem Octree-Element des Levels 1 sowie einem Voxel der Wert „4“ zugewiesen (in der Abbildung grau hinterlegt), was nun effizient in der Datenstruktur gespeichert werden soll. Das initiale Octree-Element (mit einer doppelten Linie umgeben) wird dazu solange in den entsprechenden Bereichen unterteilt – jeweils in acht gleich große Teilelemente – bis wieder jedes Element aus homogenen Daten, hier dem Wert „4“ oder dem Wert „127“, besteht. Die Linien in der Abbildung grenzen die entstehenden Octree-Elemente ein.

Diese Datenstruktur ist Voraussetzung für die sinnvolle Implementierung der im Folgenden vorgestellten Methode für Karten mit multiplen Auflösungsstufen.

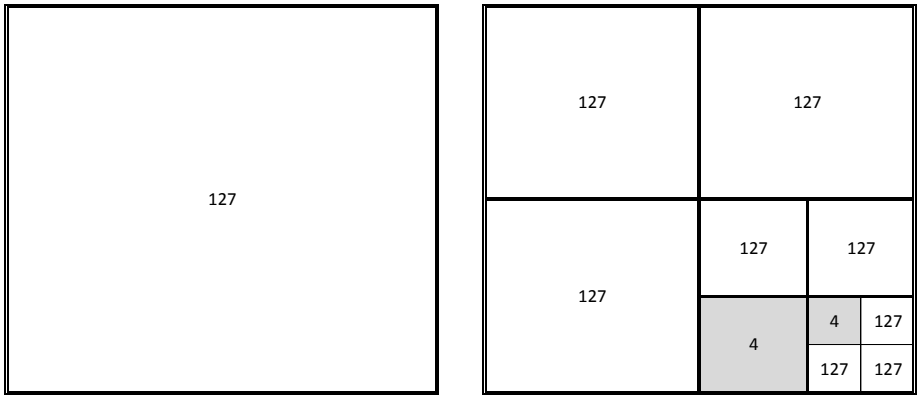


Abbildung 5.2: Unterteilung eines Octrees in 2 Dimensionen mit initial homogenen Daten (127), nach Hinzufügen von neuen Daten (4). Das initiale Octree-Element (doppelt umrandet) wird dazu so lange in je 4 gleich große Unterelemente unterteilt, bis jedes Element wieder aus homogenen Daten besteht.

5.2 Verarbeitung aufgenommener Tiefenbilder

Für jedes Pixel im aufgenommenen Tiefenbild wird ein virtueller Strahl erstellt, welcher anschließend iterativ durchlaufen wird. Als Ursprung des Strahls wird die Position des Sensors, welcher das Tiefenbild aufgenommen hat, gewählt. Dieser Strahl wird später iterativ durchlaufen und es werden dabei vom Strahl getroffene Voxel detektiert, deren Besetzungswahrscheinlichkeiten dann aktualisiert werden. Als initiale Iterationsposition wird ein Punkt auf dem Strahl gewählt, der einen zuvor definierten Mindeststand zum Ursprung aufweist. Dieser Mindestabstand wird entsprechend dem verwendeten Sensor gewählt, welcher erst ab einem diesem Abstand zuverlässige Werte liefert. Während der Iteration werden nicht, wie im bisherigen Verfahren, nur Voxel, welche von dem Strahl geschnitten werden, gesucht und aktualisiert, sondern abhängig von der Distanz zum Ursprung des Strahls und damit dem Sensor, auch größere Octree-Elemente.

Zur recheneffizienten Iteration über die Punkte des Strahls wird dieser nicht in vielen kleinen Schritten direkt entlang des Strahls durchlaufen, sondern die Iteration anderweitig in möglichst wenig Schritten durchgeführt, wobei sicher alle getroffenen Octree-Elemente erfasst werden. Dies hat neben der Recheneffizienz den Vorteil, dass wirklich alle durchdrungenen Voxel, auch wenn sie nur über kurze Distanz am Eck vom Strahl durchdrungen sind, sicher erfasst werden. Bei einem direkten Durchlaufen des Strahls, wobei die Iterationsposition auf dem Strahl liegt und in Strahlrichtung in kleinen Schritten erhöht wird um an jeder Position das zugehörige Voxel erfassen, ist dies nicht sicher gestellt. Um sicher alle Voxel zu erfassen, müssten die Iterationsabstände beliebig klein gewählt werden, da ansonsten zwischen zwei Schritten ein getroffenes Voxel übersehen werden könnte. Es

ist jedoch anzumerken, dass es nicht immer einen Informationsgewinn bringt, jedes Voxel, sei es noch so knapp vom Strahl getroffen, zu erfassen. Durch Toleranzen und Messfehler ist ein solcher Schnitt des Strahls mit Voxeln oft zufälliger Natur, aufgrund dieser Ungenauigkeiten.

Vor Beginn der Iteration entlang des Strahls wird die Gesamtzahl an nötigen Iterationsschritten innerhalb des Octrees, der den Ursprung des Strahls beinhaltet, berechnet. Diese wird aus der Kenntnis über den Startpunkt des Strahls in dem Octree sowie der Länge und Richtung berechnet. Die hier bestimmte Gesamtzahl der Schritte wird für den Fall bestimmt, wenn jedes Voxel, welches der Strahl trifft, einzeln aktualisiert würde – keine Octree-Elemente höheren Levels. Der Schritt von einem zum nächsten Voxel findet dabei nur seitlich des vorherigen statt, nicht diagonal.

Diese berechnete Gesamtzahl der Schritte muss während der Iteration laufend angepasst werden, da sie sich abhängig der Größe der erfassten Octree-Elemente verändert. Dadurch, dass im neuen Verfahren auch größere Octree-Elemente erfasst werden, verringert sich diese Schrittzahl mit zunehmender Strahllänge. Außerdem sind diagonale Schritte möglich, welche die Schrittzahl ebenfalls verringern. Für die korrekte Anpassung gilt es, folgende Fälle zu unterscheiden, wobei L das Level des Octree-Elements bezeichnet:

- Schritt ausschließlich in x-, y- oder z-Richtung:

  Schrittzahl = 2^L $\Rightarrow$ Dekrementierung um $2^L - 1$

- Schritt zu gleichen Teilen ausschließlich in x- und y-, y- und z- oder x- und z-Richtung:

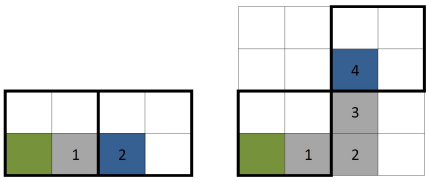
  Schrittzahl = $2^L \cdot 2$ $\Rightarrow$ Dekrementierung um $2^L \cdot 2 - 1$

- Schritt zu gleichen Teilen in x-, y- und z-Richtung:

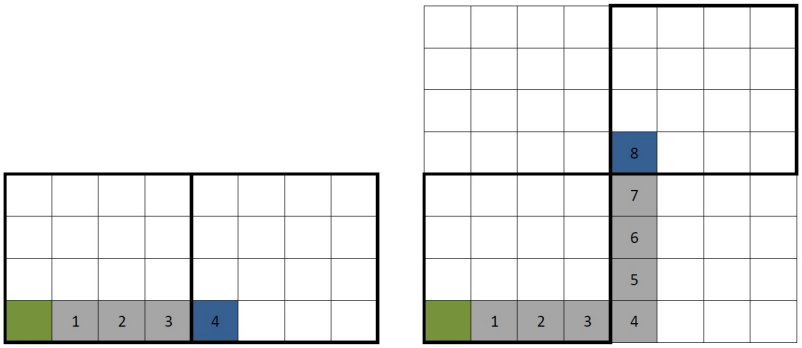
  Schrittzahl = $2^L \cdot 3$ $\Rightarrow$ Dekrementierung um $2^L \cdot 3 - 1$

Abbildung 5.3 veranschaulicht die Anpassung der Schrittzahlen für das Level 1 und 2 für die beiden erste Fälle, der dritte Fall begründet sich analog. Das grün gekennzeichnete Voxel ist der Startpunkt, das blaue der Zielpunkt und der Weg dahin ist durch graue Voxel gekennzeichnet. Für den ersten Fall ist offensichtlich, dass die nötige Schrittzahl exponentiell mit dem Level steigt. Für den zweiten Fall steigt die Schrittzahl ebenfalls exponentiell mit dem Level, jedoch für zwei Raumrichtungen, weshalb sich die Schrittzahl aus Fall-1 verdoppelt. Bei Fall-3, welcher hier nicht dargestellt ist, sind drei Raumrichtungen zu iterieren und der Aufwand verdreifacht sich.

Die benötigte Schrittzahl beim Durchlauf von Voxeln in je nur eine Richtung, wonach die Gesamtschrittzahl initial berechnet wurde, beträgt stets 1 pro Schritt. Von dem jeweiligen Resultat der oben dargestellten Fälle wird demnach zur Anpassung der Schrittzahl 1 abgezogen, da nur die Veränderung gegenüber der initialen Schrittzahl eine Anpassung erfordert.



(a) Level-1-Octree-Elemente



(b) Level-2-Octree-Elemente

Abbildung 5.3: Schritte beim voxelweisen Durchlauf von Octree-Elementen (dick schwarz umrahmt). Startvoxel in Grün, Zielvoxel in Blau, durchlaufene Voxel am Weg vom Start- zum Zielvoxel in Grau. Die Nummern geben an, um den wievielten Iterationsschritt es sich zum Erreichen des jeweiligen Voxels handelt.

Die Berechnung der gesamten Schrittzahl, unter fortlaufender Aktualisierung, für das Einfügen des Strahls findet deshalb statt, da es theoretisch recheneffizienter ist, als den Durchlauf durch Abfrage eines Abbruchkriteriums zu beenden. Nach der einmaligen Bestimmung der initialen Schrittzahl, welche als Ganzzahl gespeichert wird, erfordert die Anpassung dieser Schrittzahl im ungünstigsten Fall je eine Potenz zur Basis 2, welche effizient durch einen Bitshift berechnet werden kann, eine Multiplikation sowie zwei Additionen. Bei der Abfrage eines Abbruchkriteriums müsste jedes Mal überprüft werden, ob der Strahl den aktuellen Octree verlässt, oder zuvor in diesem endet. Bei dieser Überprüfung treten Divisionen auf Gleitkommazahlen auf, welche nach [12] im Gegensatz zu den im anderen Fall durchgeführten Rechenoperationen auf Ganzzahlen, deutlich mehr Rechenzeit beanspruchen.

5.3 Auswahl des jeweils nächsten Octree-Elements

Nachdem die Gesamtzahl der nötigen Iterationen zum Einfügen der Octree-Elemente in dem Octree berechnet wurde, muss für jeden Iterationsschritt des Einfügens des nächsten

Elements in die Datenstruktur dieses ausgewählt werden. Zur *Bestimmung des jeweils nächsten Octree-Elements* wird dazu wie folgt verfahren:

1. Berechnung der kleinsten Entfernung vom Schnittpunkt(ein)[2] zur, bzgl. der Entfernung, nächsten Kante des Octree-Elements in x-, y- und z-Richtung unter Berücksichtigung der erfassten Iterationsrichtung. Die Iterationsrichtung wird hierbei so erfasst, dass für jede Raumrichtung lediglich festgestellt wird, ob die Iteration die Koordinaten inkrementiert (positive Iterationsrichtung), dekrementiert (negative Iterationsrichtung) oder nicht verändert. Daraus entsteht eine jeweils positive, negative oder keine Iterationsrichtung für jede Raumrichtung.

2. Für jede Raumrichtung, in die eine Iteration festgestellt wurde, wird die Länge einer Strecke in Strahlrichtung mit Startpunkt am Schnittpunkt(ein) bestimmt, um die im vorherigen Schritt berechnete nächste Entfernung in die dazugehörige Raumrichtung zu erreichen. Die jeweilige Länge der betrachteten Raumrichtungen wird in einem Vektor v gespeichert. v[0] speichert die x-Komponente, v[1] die y-Komponente und v[2] die z-Komponente.

   Für Richtungen, in die keine Iteration festgestellt wurde, also keine Erhöhung der Koordinate mit Fortschreiten des Strahls erfolgt, wird die entsprechende Komponente des Vektors v auf ∞ gesetzt.

3. Unterscheidung der möglichen verschiedenen Fälle, abgeleitet aus v zur Bestimmung der Schrittrichtung, wobei t die Toleranz bezeichnet, dass Längen des Vektors v als gleich erachtet werden (hier: 10^{-14}). Dargestellt ist der Ablauf in Abbildung 5.4. Dabei wird die kleinste Komponente des Vektors gesucht. Existiert eine solche eindeutig, so erfolgt der Schritt in diese Richtung. Werden mehrere Komponenten nach Abzug der Toleranz als kleinste bestimmt, so erfolgt ein diagonaler Schritt in diese Richtung. Werden beispielsweise die x- und y-Komponente als kleinste bestimmt, so erfolgt ein Schritt in die Richtung $(x, y, z) = (1, 1, 0)$

4. Bestimmung des Schnittpunktes(aus)[3] und setzen des Schnittpunktes(in) des nächsten Octree-Elementes mit dem hier berechneten Schnittpunktes(aus) des noch aktuellen Octree-Elements.

In Abbildung 5.5 ist das Vorgehen exemplarisch für den 2D-Fall dargestellt. Vom Strahl (violett gepunktet) in dem aktuellen Octree-Element (grau), ist der Schnittpunkt(in) (violetter Punkt) sowie die Richtung bekannt. Die nächsten Entfernungen vom Schnittpunkt(in) zur den Kanten des Octree-Elements in Iterationsrichtung ist durch die geschweiften Klammern (rot in y-Richtung, blau in x-Richtung) gekennzeichnet. Um die 2,8 cm in x-Richtung zu erreichen, ist eine Strecke der Länge 4,1 cm in Strahlrichtung nötig. Analog ergibt sich

[2] Der Schnittpunkt des Strahls mit der Kante eines Octrees oder Octree-Elements, welcher beim Eintreffen des Strahls in das Element entsteht.

[3] Der Schnittpunkt des Strahls mit der Kante eines Octrees oder Octree-Elements, welcher beim Verlassen des Strahls aus dem Element entsteht.

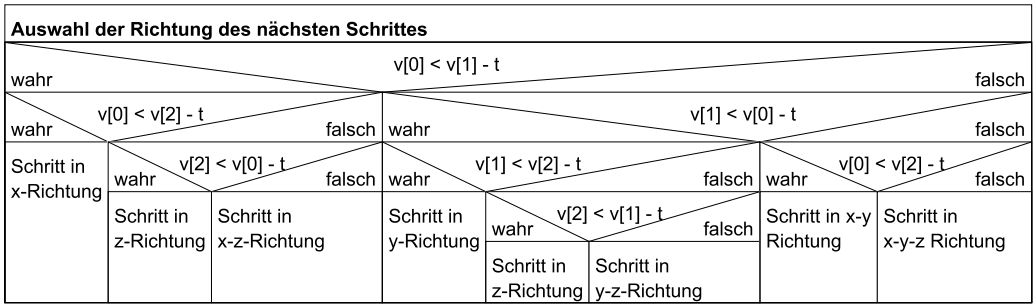


Abbildung 5.4: Nassi-Shneiderman-Diagramm zur Bestimmung der Schrittrichtung durch Analyse des Vektors v. t bezeichnet die Toleranz, unter der zwei Werte als gleich erachtet werden.

eine nötige Länge in y-Richtung zu 5,5 cm. Die x-Komponente (grün) und y-Komponente (orange) des Vektors v sind rechts abgebildet sowie dieser Vektor zahlenmäßig angegeben. In diesem Fall ist die x-Komponente v[0] des Vektors v die kleinste, weshalb korrekterweise ein Schritt in x-Richtung erfolgen muss.

5.4 Einfügen der Octree-Elemente eines Strahls in die Karte

Die Abbildungen 5.6 und 5.7 zeigen den Raum in zwei Dimensionen. Die Sensorposition befindet sich im blauen Voxel. Von dieser Position aus beginnt der Strahl, welcher am aus dem Tiefenbild gelesenen Reflexionspunkt endet. Die Auflösung entspricht initial Level-0, bei den Quadratmarkierungen am Strahl wird die Auflösung für das nächste Octree-Element gröber und das Level steigt somit um 1. Bei jeder Erhöhung des Levels wird der Strahl in dunklerem Blau dargestellt. Die Level und die jeweils entsprechende Farbkennzeichnung für die ausgewählten zu aktualisierenden Elemente sind in Abbildung 5.1 dargestellt.

Zuerst wird am Startpunkt, welcher sich am Strahl mit dem Mindestabstand zum Sensor befindet, das Voxel gewählt, welches diesen beinhaltet und gespeichert. Anschließend wird das Voxel gewählt, welches als nächstes von dem Strahl durchdrungen wird. Dies geschieht so lange, bis die Distanz des Schnittpunktes(ein) des nächsten Voxels zur Sensorposition eine zuvor definierte Länge überschreitet. Diese Längen, ab welchen sich das Level erhöht, sind in den Abbildungen durch ein Quadrat gekennzeichnet. Ab diesem Punkt werden keine Voxel, sondern Octree-Elemente höheren Levels erfasst. Ab welcher Entfernung jeweils das Level erhöht wird, ist in Abschnitt 5.5 erläutert. Zum weiteren Vorgehen werden hier zwei verschiedene Ansätze vorgestellt.

In Abbildung 5.6 wird eine Möglichkeit gezeigt, bei dem in der Karte nicht alle Octree-Elemente, nach dem Einfügen mehrerer Strahlen, deckungsgleich liegen müssen. Es wird

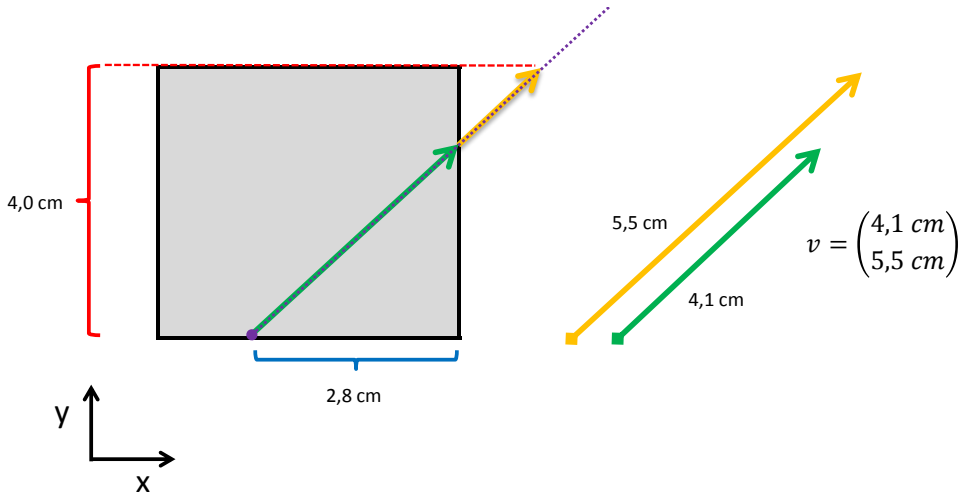


Abbildung 5.5: Verfahren zur Bestimmung des nächsten Octree-Elements anhand der Daten von Position und Richtung des Sensorstrahls (violett gepunktete Linie). Die nächste Octree-Element-Kante in Iterationsrichtung ist vom Schnittpunkt(ein) (violetter Punkt) in x-Richtung 2,8 cm und in y-Richtung 4,0 cm entfernt. Um diese Kanten mit einer Strecke entlang des Sensorstrahls, beginnend am Schnittpunkt(ein), zu erreichen, sind 4,1 cm bzw. 5,5 cm nötig. Diese Werte werden in v gespeichert.

dabei jeweils das als nächstes vom Strahl geschnittene Voxel detektiert, wobei die Schnittkante in der Abbildung rot eingefärbt ist. Dieses wird nach der Detektion auf die aktuell geforderte Octree-Element-Größe expandiert ohne dabei das durch den Octree vorgegebene Raster zu berücksichtigen. Das geschnittene Voxel ist dabei in dunklerer der jeweiligen Farbe des Octree-Elements nach Abbildung 5.1 dargestellt und die Expansion mit orangen Pfeilen gekennzeichnet. Die Richtung der Expansion erfolgt, ausgehend vom erfassten Voxel, im hier gezeigten 2D-Fall in positive x- und y-Richtung, da die x- sowie y-Werte der Punkte auf dem Strahl, mit fortschreitender Distanz zum Ursprung des Strahls, ansteigen. Im 3D-Fall wird nach dem gleichen Vorgehen ebenfalls die z-Komponente miteinbezogen.

Bei der anderen Möglichkeit, welche in Abbildung 5.7 dargestellt ist, wird darauf geachtet, dass alle Octree-Elemente gleichen Levels sich räumlich nicht überschneiden. Dadurch ist ein effizientes Update von Octree-Elementen möglich, da keine Überlappungen mit eingerechnet werden müssen. Aus diesem Grund wird diese Möglichkeit bevorzugt und implementiert. Die verwendeten Octrees mit ihren Octree-Elementen teilen sich bereits, wie in Abschnitt 5.1 beschrieben, in fester geometrischer Anordnung. Eine unpassende geometrische Anordnung kann demnach nicht auftreten, da die Octree-Elemente solange vorschriftsgemäß unterteilt werden, bis an der entsprechenden Stelle das Octree-Element des passenden Levels erzeugt ist und die Daten darin gespeichert werden können.

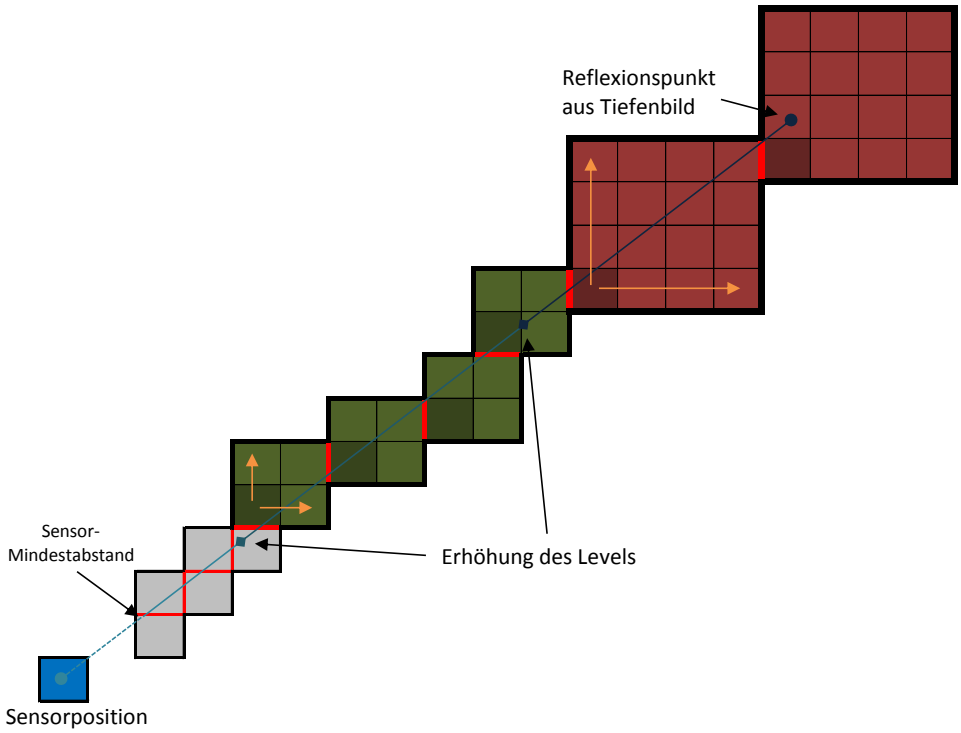


Abbildung 5.6: Einfügen eines Strahls. Es werden alle geschnittenen Voxel (Kanten, welche vom Strahl geschnitten werden, sind in rot eingefärbt) erfasst und mit zunehmender Distanz zur Sensorposition die Voxel zu größeren Octree-Elementen expandiert. Die Anordnung von Octree-Elementen gleichen Levels zwischen dem Einfügen verschiedener Strahlen muss nicht deckungsgleich sein.

Es kann jedoch dabei passieren, dass nach einer Levelerhöhung das neue Octree-Element zuvor erstellte Octree-Elemente des gleichen Strahls geringeren Levels beinhaltet. In Abbildung 5.7 würde bei den mit X gekennzeichneten Voxeln, wenn an der Stelle nach der Levelerhöhung Octree-Elemente des Levels 1 eingefügt würden, dieser Fall eintreten. Ein solches Octree-Element, alloziert an der Stelle der mit X gekennzeichneten Voxel, würde das vorherige Voxel, in dem das Quadrat zur Kennzeichnung der Levelerhöhung liegt, beinhalten. Für diesen Fall wird ein Element des gleichen Levels gewählt und das Level erst dann erhöht, wenn dieser Fall nicht mehr eintritt. In dem Beispiel ist dies nach zwei Schritten, bei denen noch ein Level-0-Element eingefügt wird, der Fall. Bei der Erhöhung des Levels 1 auf 2, tritt der Fall in dem Beispiel nicht ein. Um das geometrisch korrekte Einfügen von Octree-Elementen zu gewährleisten werden nach jeder Levelerhöhung folgende zwei Fälle überprüft:

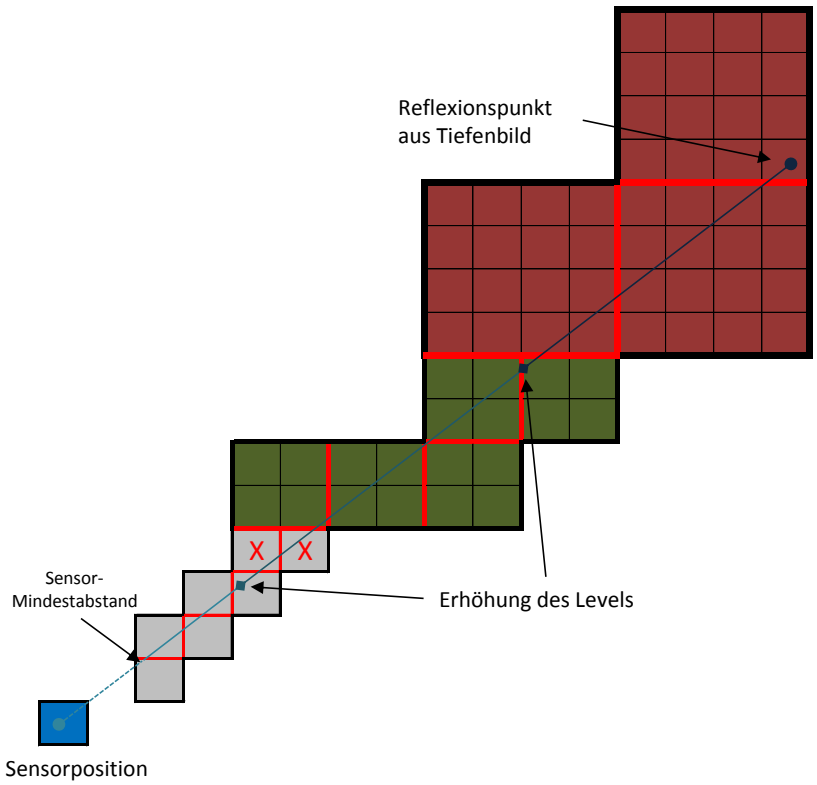


Abbildung 5.7: Einfügen eines Strahls. Es werden mit zunehmender Distanz zur Sensorposition größere Octree-Elemente, welche vom Strahl geschnitten werden (Kanten, welche vom Strahl geschnitten werden, sind in rot eingefärbt), erfasst. Diese sind geometrisch fest gemäß der in 5.1 beschriebenen Datenstruktur angeordnet. Eine Levelerhöhung wird verweigert, wenn dadurch das neue Octree-Element bisherige des aktuellen Einfügevorgangs des Strahls beinhalten würde. Dies tritt an den Stellen der mit einem roten X gekennzeichneten Voxel auf.

- Für positive Iterationsrichtung: Der Anker des neuen Octree-Elements ist derselbe, wie der des bisherigen.
- Für negative Iterationsrichtung: Die Koordinate welche dem Anker des neuen Octree-Elements in diesem Element gegenüber liegt ist dieselbe, wie die Koordinate welche dem Anker des bisherigen Octree-Elements in diesem Element gegenüber liegt.

Falls die Richtung des Iterationsschritts ausschließlich in x-, y- oder z-Richtung bestimmt wurde, ist es hinreichend, nur die jeweilige Komponente des Ankers zu überprüfen. Bei einem Schritt in x- und y-, y- und z- oder x- und z-Richtung nur die beiden jeweiligen Komponenten. Dadurch kann der Rechenaufwand reduziert werden.

Falls einer dieser Fälle eintritt, bedeutet dies, dass das neue Octree-Element, Teile des zuvor erstellten beinhaltet. Diese Abfrage wird nur bei Levelerhöhung durchgeführt, da bei gleichbleibendem Level, aufgrund der beschriebenen Iterationsvorschrift, dieses Problem nicht auftreten kann. Tritt einer der beiden oben genannten Fälle nach einer geplanten Levelerhöhung ein, wird diese Levelerhöhung verweigert. Das neue Element wird auf dem bisherigen Level eingefügt und die Levelerhöhungsanfrage bei dem nächsten einzufügenden Element erneut geprüft, bis kein Konflikt mehr bzgl. dieser beiden Fälle auftritt.

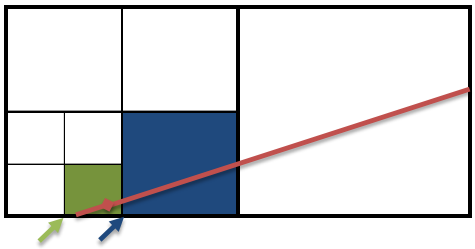
In Abbildung 5.8 ist das Vorgehen für den 2D-Fall veranschaulicht. Der Strahl ist in rot dargestellt, die Entfernung, ab der das Level erhöht werden soll, mit einer Raute markiert. Level-0-Elemente sind in grün, Level-1-Elemente in blau dargestellt. Die Ankerpunkte der jeweiligen Octree-Elemente sind mit Pfeilen gekennzeichnet. In den beiden linken Unterabbildungen 5.8(a) und 5.8(c) ist der Fall dargestellt, dass kein Problem beim Einfügen des Octree-Elements nach Levelerhöhung auftritt. Das Level kann im ersten Versuch erhöht und das Element eingefügt werden. In den beiden rechten Unterabbildungen 5.8(b) und 5.8(d) tritt der Fall auf, dass der Anker des neuen Octree-Elements gleich dem des alten wäre (Abbildung 5.8(b)). Demnach wird das Level zunächst nicht erhöht und das Octree-Element auf bisherigem Level eingefügt (rot umrandet und Anker in rot). Anschließend tritt der Fall nicht mehr auf und das Level wird ordnungsgemäß erhöht.

Verlässt der Strahl einen Octree, wird das Verfahren in dem Octree wiederholt, in dem sich der Strahl fortsetzt. Als Startpunkt wird dabei zunächst das Voxel gewählt, in dem der Strahl die Octree-Kante des neuen Octrees durchdringt und dieses anschließend auf die Größe des Octree-Elements entsprechenden Level gesetzt. Jedem neuen Octree wird als zusätzliche Information die Entfernung vom Ursprung des Strahls zum Schnittpunkt(ein) des Octrees mitgegeben, um eine durchgehende Levelberechnung, welche auf der Entfernung zum Ursprung basiert, zu gewährleisten.

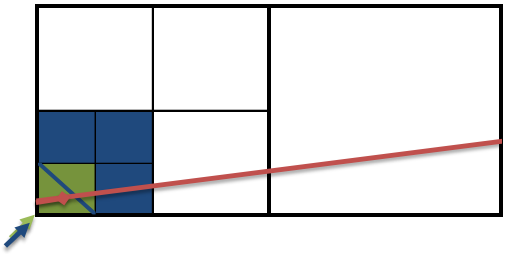
5.5 Bestimmung der Distanzen zur Levelerhöhung der Octree-Elemente

Zur Bestimmung, ab welcher Distanz zum Sensor das Level der Octree-Elemente inkrementiert werden soll, werden die Charakteristika der verwendeten Sensoren betrachtet. Im Verfahren aus Abschnitt 3.2.2 wurde davon ausgegangen, dass die Qualität der Daten quadratisch mit der Entfernung abnimmt. Die Beziehung zwischen Qualität und Entfernung soll hier näher betrachtet werden.

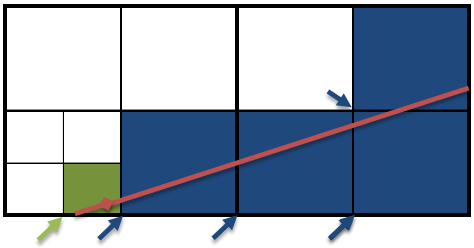
Die hier verwendeten Sensoren vom Typ IFM O3D100 wurden in [32] untersucht. Zur Kalibrierung der Kamera wurde dabei ein schwarz-weißes Schachbrettmuster vermessen. Die dabei bestimmte Standardabweichung der gemessenen Entfernung in Abhängigkeit der tatsächlichen Entfernung ist in Abbildung 5.9 dargestellt. Bei Verwendung der Kamera



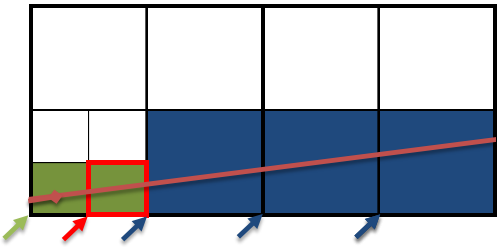
(a) Levelerhöhung kann im ersten Versuch durchgeführt werden. Fortsetzung in (c).



(b) Der Anker des neuen Octree-Elements bei Levelerhöhung wäre gleich dem des alten. Fortsetzung in (d).



(c) Weitere Octree-Elemente gleichen Levels können ohne spezielle Abfrage stets konfliktfrei eingefügt werden.



(d) Das Level wird zunächst nicht erhöht und das Octree-Element auf bisherigem Level eingefügt (rot umrandet und Anker in rot). Anschließend tritt der Fall nicht mehr auf und das Level wird ordnungsgemäß erhöht.

Abbildung 5.8: Einfügen von Octree-Elementen nach Levelerhöhung. Der Strahl ist in rot dargestellt, die Entfernung, ab der das Level erhöht werden soll, mit einem Quadrat markiert. Level-0-Elemente sind in Grün, Level-1-Elemente in Blau dargestellt. Die Ankerpunkte der jeweiligen Octree-Elemente sind mit Pfeilen gekennzeichnet.

im für diese Arbeit typischen Anwendungsgebiet, Räumen, etwa einem Laborraum mit Tischen, Stühlen, Schränken und diversen Geräten, ergibt sich jedoch ein anderer Verlauf. Die Zunahme der Standardabweichung, welche bei dem Fall der Kalibrierung annähernd linear verlief, ist hier exponentiell. Nach [43] kann die Standardabweichung der Messwerte σ solcher Tiefensensoren mit folgender Formel approximiert werden, wobei a einen Skalierungsfaktor, d die gemessene Distanz des Reflexionspunktes zum Sensor und c einen Exponenten darstellt:

$$\sigma = \sqrt{a \cdot d^c} \text{ [mm]} \tag{5.1}$$

Als Werte der Parameter für den verwendeten Sensor in der typischen Umgebung werden in der vorhandenen Simulation des Instituts momentan folgende Werte verwendet, welche auch in dieser Arbeit unverändert benutzt werden sollen:

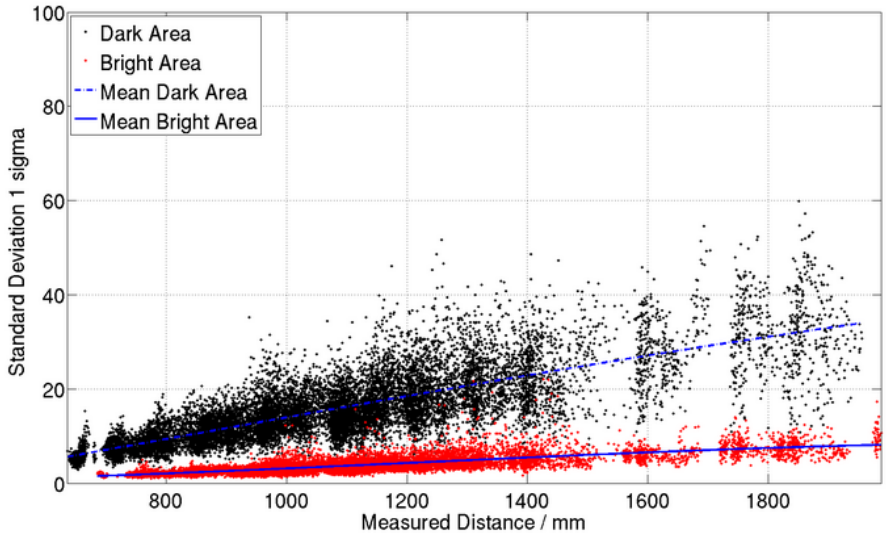


Abbildung 5.9: Standardabweichung der gemessenen Entfernung in Abhängigkeit der tatsächlichen Entfernung der IFM O3D100 ToF-Kamera für schwarz-weißes Schachbrettmuster zur Kalibrierung. Die Messwerte für dunkle Gebiete sind in schwarz, für helle Gebiete in rot dargestellt. Quelle: Stefan Fuchs, DLR

$$a = 5,52 \cdot 10^{-10}$$
$$c = 3,61$$

Es ergibt sich dabei ein Verlauf der Standardabweichung gegenüber der gemessenen Distanz, welcher in Abbildung 5.10 dargestellt ist. Typischerweise wird der Sensor am Institut zur Vermessung für Entfernungen bis maximal 7,00 m verwendet. Größere Messwerte werden aufgrund der hohen Messunsicherheit verworfen.

Das Level beginnt initial bei 0 am Ort des Sensors. Eine Erhöhung des Levels um jeweils 1 findet statt, sobald die Standardabweichung der Messdaten einen Wert von $\frac{1}{3}$ der Kanten-Länge des aktuellen Octree-Elements überschreitet. Die daraus resultierenden Entfernungen ab welchen jeweils eine Erhöhung stattfindet sind für Entfernungen bis 7,00 m und damit Level-8 in Tabelle 5.1 dargestellt. Als maximale Auflösung, welche hierbei der Länge der Kante eines Voxels, also dem Octree-Element des Levels 0, entspricht, wird ein Wert von 4 mm gewählt. Zu den Entfernungen d sind in der Tabelle die dazugehörige Standardabweichung σ, das Level auf welches ab dieser Entfernung erhöht wird L_{neu} sowie die Kantenlänge des Octree-Elements neuen Levels a angegeben.

Diese Berechnungsvorschrift der Erhöhung bei einem Übersteigen der Standardabweichung von $\frac{1}{3}$ der Länge des aktuellen Octree-Element-Kante wurde experimentell für den

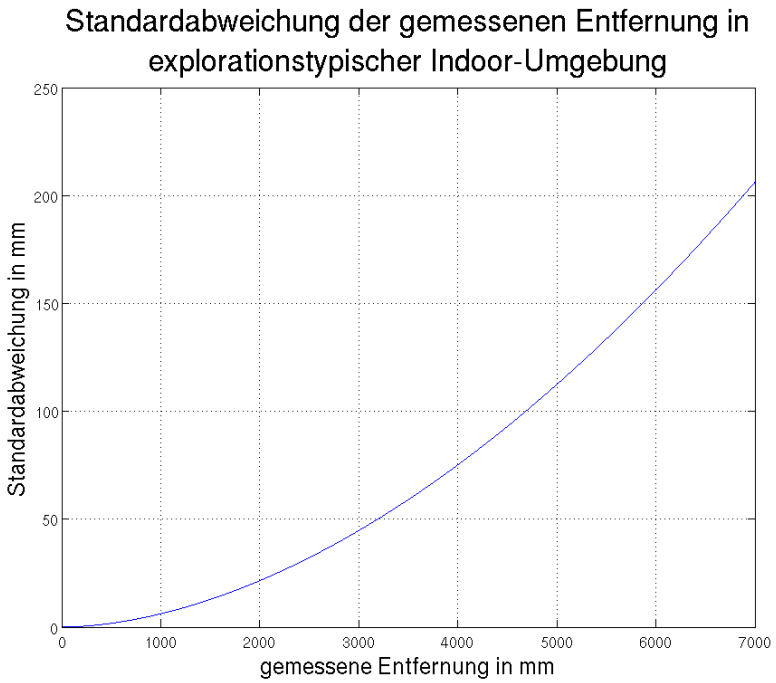


Abbildung 5.10: Standardabweichung der gemessenen Distanz in für die Exploration von Innenräumen typischer Umgebung der IFM O3D100 ToF-Kamera. Darstellung gemäß Formel 5.1.

verwendeten Sensor und dem Zweck der Exploration von Innenräumen bestimmt. Für andere Sensoren oder andere Anwendungsgebiete kann durch Anpassung dieses Wertes der gewünschte Levelanstiegsverlauf festgelegt werden.

Es besteht bei dem Verfahren des Weiteren *die Möglichkeit, ein Level fest zu wählen*, welches unabhängig von der Entfernung für jedes Octree-Element verwendet wird. Während der Exploration kann dieses außerdem angepasst werden und so jeweils einer Reihe von Messungen manuell ein Level, zur Integration der Daten in die Karte, vorgegeben werden.

Ein Anwendungsfall dafür ist die Exploration durch einen ferngesteuerten Roboter. Der Bediener kann gezielt für ihn interessante Gebiete anfahren und mit der für ihn nötigen Auflösung vermessen. Bei bestimmten Anwendungen gibt es außerdem festgeschriebene Vorgaben an die Auflösung für bestimmte Objekte, welche vermessen werden sollen. Durch eine manuelle Festsetzung eines bestimmten Levels für Aufnahmen können die Vorgaben so erfüllt werden. Die einzige Möglichkeit der Einstellung der Auflösung bei dem bisherigen Verfahren, war es, die Auflösung über die Voxelkantenlänge zu definieren. Diese ist dann jedoch für die ganze Karte fix und es müsste demnach für jedes Gebiet, welches eine andere Auflösung erfordert, eine neue Karte angelegt werden. Neben dem erhöh-

Tabelle 5.1: Entfernungen d mit dazugehöriger Standardabweichung σ des verwendeten Sensors, ab welchen das Level auf L_{neu} erhöht wird mit zugehöriger Kantenlänge des Octree-Elements des Levels L_{neu}

d	σ	L_{neu}	a
430,307 mm	1,33333 mm	1	8 mm
631,762 mm	2,66667 mm	2	16 mm
927,533 mm	5,33333 mm	3	32 mm
1361,77 mm	10,6667 mm	4	64 mm
1999,31 mm	21,3333 mm	5	128 mm
2935,32 mm	42,6667 mm	6	256 mm
4309,55 mm	85,3333 mm	7	512 mm
6327,14 mm	170,667 mm	8	1024 mm

ten Speicher- und Rechenaufwand geht auch die Information der relativen Position der Gebiete zueinander verloren.

5.6 Aktualisierung der Besetzungswahrscheinlichkeiten

5.6.1 Generelles Verfahren bisher bei der Aktualisierung auf Voxelebene

Nach Selektion der eines Strahls geschnittenen Voxel, werden diesen ihre jeweilige Besetzungswahrscheinlichkeiten zugewiesen. Hierzu wird eine 1-Byte-Variable mit dem Wertebereich [0, 255] benutzt. Je höher der Wert, desto höher die Besetzungswahrscheinlichkeit, wobei der Wert 127 dazu genutzt wird um unbekanntes (noch nicht vermessenes) Gebiet darzustellen.

Die Besetzungswahrscheinlichkeiten repräsentieren die Sensorunsicherheiten. Zur Berechnung dieser Wahrscheinlichkeiten gibt es mehrere Möglichkeiten. Das am meisten in der Robotik verwendete Verfahren ist das Bayes-Update, welches u.A. in [26, 47] verwendet wird. Andere Verfahren [85, 95] benutzen Belief Ansätze. Das Update mit Fuzzy-Mengen, wie in [33] wird selten verwendet. Eine Übersicht der Ansätze ist in [22] und [83] ausführlich dargestellt.

Bei Dymodda, welches das zugrunde liegende Mapping-Verfahren dieser Weiterentwicklung darstellt, stehen drei verschiedene probabilistische Update-Algorithmen zur Verfügung: Ein Bayes-basiertes, ein Fuzzy-basiertes sowie ein Dempster-Shafer basiertes. Des weiteren steht ein naiver Ansatz zur Verfügung.

Der am häufigsten bei Dymodda verwendete Ansatz ist der Bayes-Ansatz, mit welchem dann die Besetzungswahrscheinlichkeiten berechnet und schritthaltend aktualisiert wer-

den. Die Wahrscheinlichkeit, dass ein Voxel v mit einer bisherigen Besetzungswahrscheinlichkeit von $p_{t-1}(v)$ nach einer neuen Messung z_t unter Verwendung des Sensormodells $p(z|v)$ zum Zeitpunkt t besetzt ist $p_t(v)$ berechnet sich dabei wie folgt [10]:

$$p_t(v) = p(v|z_t)\ \alpha\ p(z_t|v) \cdot p_{t-1}(v) \tag{5.2}$$

Als Sensormodell wird ein inverses Modell nach [47] verwendet. Dieses Modell repräsentiert eine bedingte Dichte oder Wahrscheinlichkeitsfunktion zur Beschreibung des Messvorgangs.

5.6.2 Anpassung des Verfahrens bei Verwendung multipler Auflösungsstufen

Der beschriebene und meist bei Dymodda verwendete Bayes-Ansatz wird hier weiterhin verwendet, jedoch ist bei der Benutzung der hier vorgestellten Funktionalität multipler Auflösungsstufen zur effizienten Verarbeitung neuer Daten, eine Betrachtung verschiedener Fälle bei der Aktualisierung von Besetzungswahrscheinlichkeiten notwendig. Bei der Aktualisierung größerer Octree-Elemente steht durch das dazugehörige Level weitere Information über die Qualität der Daten zur Verfügung, die beim Update auf ausschließlich Voxelebene nicht zu Verfügung standen.

Allgemeines Ziel ist es durch erneute Scans von Bereichen die Qualität der Karte zu erhöhen. Wenn ein Bereich, welcher zuvor mit höherer Auflösung (Sensor näher am Objekt) vermessen wurde, bei erneutem Scan Daten niedrigerer Auflösung vorliegen (Sensor weiter entfernt), wird hierbei davon ausgegangen, dass durch Integration der neuen Daten die Qualität nicht verbessert werden kann. Durch höheres Rauschen ist sogar eine Verschlechterung der Qualität zu erwarten. Außerdem ist Rechenaufwand für eine Integration nötig, der vermieden werden kann. Um diese Funktionalität zu gewährleisten muss von jedem Octree-Element die Auflösung bekannt sein - sie ist nicht implizit. Es kann beispielsweise passieren, dass hoch aufgelöste Daten auf Voxelebene erfasst werden, aufgrund von Homogenität der Daten diese jedoch zu höherwertigen Octree-Elementen zusammengefasst werden. Auch durch die Änderung von Daten während erneuten Scannens können homogene Bereiche entstehen, welche zu Octree-Elementen höheren Levels führen. Das Level eines Octree-Elements gibt also nicht zwingend Auskunft über die Qualität der beinhalteten Daten.

Es bieten sich zur Speicherung der aufgenommenen Auflösung der Daten zwei Methoden an.

Eine simple Methode wäre es, das Zusammenfügen von homogenen Bereichen zu Octree-Elementen höheren Levels zu unterbinden. Dann würde die Größe der Octree-Elemente in der Karte die Auflösung zum Zeitpunkt der Messung des entsprechenden Elements repräsentieren. Der Speicheraufwand wäre dabei jedoch sehr hoch.

Eine speicherplatzeffiziente Methode, vor allem falls viele homogene Bereiche auftreten, welche sonst nicht mehr zusammengefasst werden könnten, wäre es, zu jedem Octree-Element die aufgenommene Auflösung in einer 1-Byte-Variable mit abzuspeichern. Bei einer Exploration treten typischerweise viele homogene Bereiche auf. Zu Beginn ist der Großteil „unbekannt“ und der Raum zwischen Objekten wird bei wiederholten Messungen „nahezu sicher frei“. Da das Verfahren primär für Explorationsanwendugen eingesetzt werden soll wird die letztere Methode gewählt und die Auflösung für jedes Voxel mit abgespeichert. Unbekannte Bereiche erhalten stets den maximal möglichen Auflösungswert zugewiesen, welcher diese speziell kennzeichnet. Sie sind in der Regel initial auf diese Besetzungswahrscheinlichkeit gesetzt worden und nicht durch Messung bestimmt worden worden. Falls nach der Aktualisierung dieser Wert erneut auftritt, ist dies auf sehr widersprüchliche oder ungenaue Messungen zurückzuführen. Es wird deshalb davon ausgegangen, dass jede Aktualisierung dieser Elemente mit neuen Messdaten, unabhängig ihrer Auflösung einen Informations- und Qualitätsgewinn mit sich bringt. Durch das Setzen des maximalen Auflösungswertes, wird eine Aktualisierung sichergestellt.

Folgende Möglichkeiten müssen bei der Aktualisierung von Octree-Elementen betrachtet werden:

- Es sollen Octree-Elemente aktualisiert werden und dabei ist die *Auflösung* der neuen Daten *geringer*, als die des in der Karte bereits bestehenden Octree-Elements:

  In dem Fall werden die neuen Daten verworfen und es erfolgt kein Update.

- Es sollen Octree-Elemente aktualisiert werden und dabei ist die *Auflösung* der neuen Daten *höher*, als die des in der Karte bereits bestehenden Octree-Elements:

  In dem Fall werden die bisherigen Daten mit den neuen überschrieben.

- Es sollen Octree-Elemente aktualisiert werden und dabei ist die *Auflösung* der neuen Daten *gleich* denen, des in der Karte bereits bestehenden Octree-Elements:

  In dem Fall wird der arithmetische Mittelwert der neuen und alten Daten berechnet und gespeichert.

6 Test und Vergleich des erweiterten Mapping-Verfahrens

Vorbemerkung: In den Abbildungen dieses Kapitels werden bzgl. der Einfärbung von dargestellten Octree-Elementen nachfolgende Konventionen getroffen:

Die Besetzungswahrscheinlichkeit von Octree-Elementen wird über Graustufen modelliert. Sie wird auf 256 Werte, die ganzen Zahlen von 0 bis 255, quantisiert. Der Wert 0 steht für die Quantisierung der minimalen Besetzungswahrscheinlichkeit und wird in weiß (RGB: (255, 255, 255)) dargestellt, der Wert 255 wird für maximale Besetzungswahrscheinlichkeit benutzt und in schwarz (RGB: (0, 0, 0)) dargestellt. Abstufungen werden durch die entsprechenden Werte dazwischen (RGB: (1, 1, 1) - (254, 254, 254)) repräsentiert.

Dem Wert 127 kommt eine spezielle Bedeutung zu. Er repräsentiert unbekannten Raum (Besetzungswahrscheinlichkeit 50 %). Initial wird der von der Karte abgebildete Raum, falls nicht explizit anders angegeben, mit diesem Wert initialisiert.

Nicht allozierter Raum wird in den Abbildungen blau dargestellt. Dieser ist nicht Bestandteil einer Karte.

6.1 Einfügen von Strahlen

Dieser Versuch soll die prinzipielle Verarbeitung der Pixel eines Tiefenbildes mit dem hier weiterentwickelten Verfahren veranschaulichen.

Die Karte besteht aus einem Octree und wird wie folgt initialisiert:

- Kantenlänge eines Octrees: 6400 mm
- Octree Anker-Position: (x, y, z) = (0 mm, 0 mm, 0 mm)
- Maximale Auflösung (Kantenlänge eines Voxels): 4 mm
- Mindestabstand zum Sensor: 1 mm
- Maximale Länge des Sensorstrahls: 7000 mm

Bei dem hier durchgeführten Versuch sollen drei Pixel eines Tiefenbildes eingefügt werden. Zur guten Visualisierbarkeit werden die Punkte so gewählt, dass die beim Einfügen aus den Punkten erstellten Strahlen auf den Kanten des die Karte repräsentierenden, initialen Octrees und vollständig innerhalb dieses Octrees liegen. Während der Laufzeit des Verfahrens wird die Karte somit nicht um weitere Octrees erweitert. Es werden dazu folgende drei Punkte p_1, p_2 und p_3 in der Form $p = (x, y, z)^T$ definiert:

$$p_1 = \begin{pmatrix} 5002\ mm \\ 2\ mm \\ 2\ mm \end{pmatrix},\ p_2 = \begin{pmatrix} 2\ mm \\ 5002\ mm \\ 2\ mm \end{pmatrix},\ p_3 = \begin{pmatrix} 2\ mm \\ 2\ mm \\ 5002\ mm \end{pmatrix}.$$

Zum Einfügen dieser Punkte werden drei Strahlen der Länge 5000 mm angelegt, welche alle ihren Ursprung mit den Koordinaten $(x, y, z) = (2\ mm, 2\ mm, 2\ mm)$ im Voxel am Anker des Octrees haben. Dieser Ursprung repräsentiert die Sensorposition des Sensors, mit welchem das virtuelle Tiefenbild aufgenommen wurde. Die Richtungsvektoren der Strahlen r_1, r_2, r_3 in der Form $r = (x, y, z)^T$ lauten:

$$r_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},\ r_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},\ r_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Der Mindestabstand zum Sensor konnte mit 1 mm so klein gewählt werden, da die einzufügenden Strahlen, durch Auswahl der Punkte p_1, p_2 und p_3, manuell bestimmt wurden und nicht Resultat einer (simulierten) Messung sind. Durch diesen geringen Abstand ist die Arbeitsweise des Verfahrens gut zu erkennen und die Darstellung der Karte bleibt übersichtlich.

In Abbildung 6.1 ist die Karte nach Einfügen der Punkte dargestellt. Abbildung 6.1(a) zeigt dabei das Resultat nach Einfügen der Punkte mit dem bisherigen Verfahren, wohingegen Abbildung 6.1(b) das Ergebnis des Einfügens unter Verwendung des neuen Verfahrens darstellt. Die in Octree-Elemente quantisierten, abgebildeten Strahlen enden an ihrem virtuellen Reflexionspunkt. In den Abbildungen ist das letzte Octree-Element entsprechend dunkel gefärbt. Vorherige Elemente sind gemäß dem Update-Verfahren nach Bayes mit abnehmender Wahrscheinlichkeit als besetzt und somit zunehmender Wahrscheinlichkeit als frei anzunehmen. Die Besetzungswahrscheinlichkeit nimmt zum Ursprung des Strahls hin ab, die Farbe in der Darstellung wird somit, mit größerer Entfernung zum Reflexionspunkt, heller.

Anmerkung: Die unterschiedlichen Grautöne der Flächen des, die gesamte Karte repräsentierenden, Octree-Würfels, ausschließlich der Stellen, an denen der Strahl eingefügt wurde, entsprechen alle dem Wert 127 mit welchem sie initialisiert wurden. Sie sind lediglich zur dreidimensionalen Visualisierung unterschiedlich hell in der Abbildung dargestellt.

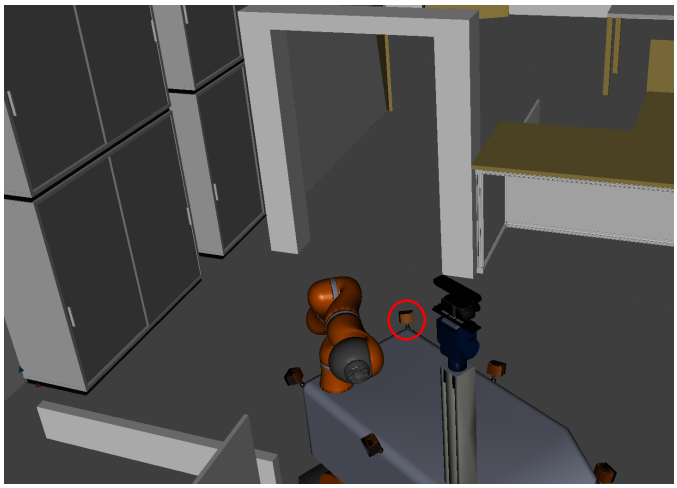
(a) Karte nach Einfügen der Strahlen mit bisherigem Verfahren auf Voxelebene.

(b) Karte nach Einfügen der Strahlen mit neuem Verfahren auf verschiedenen Octree-Element-Level.

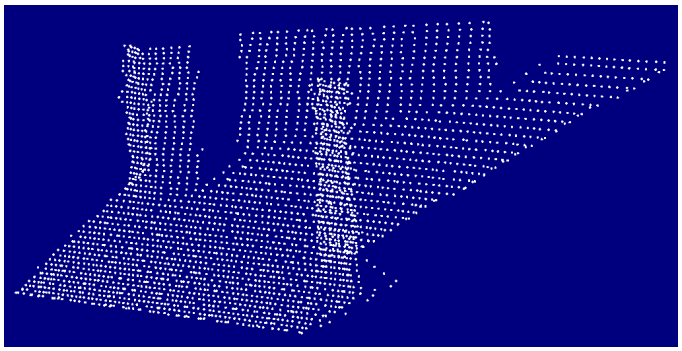
Abbildung 6.1: Karte nach dem Einfügen der Strahlen mit dem bisherigen als auch dem neuen Verfahren. Der Raum wurde initial auf unbekannt gesetzt, hier durch das Grau des großen Würfels dargestellt. Beim Einfügen der Strahlen werden entsprechend dem Modell der Messunsicherheit mit dem Bayes-Update die Besetzungswahrscheinlichkeiten der Elemente angepasst. Je dunkler, desto höher die Besetzungswahrscheinlichkeit. In der linken, oberen Ecke ist jeweils der gesamte Octree dargestellt, aus dem ein Bereich vergrößert abgebildet ist.

6.2 Einlesen und Verarbeitung eines Tiefenbildes

Hier soll ein Tiefenbild, welches vom zuvor beschriebenen Sensor aufgenommen wurde, in die Karte integriert werden. Die Aufnahme des Tiefenbildes wurde dazu in der Simulation mit den entsprechenden Parametern des Sensors durchgeführt. In Abbildung 6.2(a) ist die Pose der ToF-Kamera (rot umkreist) in der Simulationsumgebung dargestellt, mit welcher das Bild aufgenommen wurde. In Abbildung 6.2(b) ist die Punktwolke, welche die Tiefenpunkte des aufgenommenen Tiefenbildes repräsentiert, dargestellt. Für diese Darstellung als Punktwolke wurden die Pixel des Tiefenbildes um ihren Tiefenwert senkrecht zur Bildebene verschoben. Das Tiefenbild besitzt eine Auflösung von 50 x 64 Pixel, die Punktwolke umfasst daher 3200 Punkte.



(a) Pose der Tof-Kamera am Roboter in der Simulationsumgebung.



(b) Punktwolke des Tiefenbildes.

Abbildung 6.2: Pose der Tof-Kamera in der Simulationsumgebung mit aus der Messung resultierender Punktwolke des aufgenommenen Tiefenbildes.

Die zu erstellende Karte besteht initial aus einem Octree. Die Position und Größe dieses Octrees wurde so gewählt, dass alle Daten des Tiefenbildes, nach erfolgter Verarbeitung und Integration, innerhalb dieses Octrees liegen. Die Karte wird dadurch beim Update mit den Messdaten nicht um weitere Octrees erweitert. Sie wird wie folgt initialisiert:

- Kantenlänge eines Octrees: 16384 mm
- Maximale Auflösung (Kantenlänge eines Voxels): 8 mm
- Mindestabstand zum Sensor: 50 mm
- Maximale Länge des Sensorstrahls: 7000 mm

Tabelle 6.1: Auswertung von Laufzeiten und Speicherbedarf der beiden Verfahren bei der Verarbeitung eines Tiefenbildes.

	bisheriges Verfahren		**neues Verfahren**		**Differenz**	
	M	S	M	S	ΔM abs.	ΔM rel.
t_1	0,10005 s	0,02615 s	0,00467 s	0,00039 s	0,09539 s	95,34 %
t_2	0,97153 s	0,21567 s	0,64834 s	0,12879 s	0,3232 s	33,27 %
t_3	81,56311 s	1,69105 s	24,91811 s	5,20884 s	56,65 s	69,45 %
t_{ges}	82,63469 s	1,71489 s	25,57112 s	5,33835 s	57,06 s	69,06 %
$stor$	66.497 kB	0 B	2.578 kB	0 B	63.919 kB	96,12 %

Die resultierende Karte, nach Verarbeitung des Tiefenbildes mit dem weiterentwickelten Verfahren, ist in Abbildung 6.4 dargestellt. Abbildung 6.3 zeigt das Resultat mit dem bisherigen Verfahren, welches ausschließlich auf Voxelebene Daten speichert. Dargestellt sind jeweils alle Octree-Elemente mit einer Besetzungswahrscheinlichkeit höher 50 %. Um die Abbildungen übersichtlich zu halten, werden alle anderen Elemente ausgeblendet. Es ist bei dem neuen Verfahren mit multiplen Auflösungsstufen ersichtlich, dass der Durchgang durch den Türrahmen, trotz geringerer Auflösung an den entsprechenden Stellen, auch hier gut zu erkennen ist. Ein Pfad durch den Durchgang kann also auch mit dem neuen Verfahren vom Standort des Roboters aus geplant werden. Die Wahl der maximalen Auflösung mit den Regeln zur distanzabhängigen Anpassung der Auflösung ist, zum Zweck der Navigation und Pfadplanung, augenscheinlich sinnvoll gewählt.

Die Laufzeiten t sowie der Speicherbedarf $stor$ der beiden Verfahren sind in Tabelle 6.1 gelistet. Die gesamte Laufzeit t_{ges} unterteilt sich, bei dem hier durchgeführten Versuch, in die benötigte Zeit um den Raum zu initialisieren t_1, die Strahlen aus dem Tiefenbild zu erzeugen und einzufügen t_2 sowie abschließend den aktualisierten Raum in eine .obj[1]-Datei zu speichern t_3. Die relative und absolute Zeit- bzw. Speicherersparnis des neuen Verfahrens sind ebenfalls in der Tabelle angegeben.

Da das simulierte Rauschen des Sensors und nicht immer zu beeinflussende Hintergrundprozesse des Simulationsrechners zu Abweichungen der Messergebnisse führen, sind in der Tabelle für jedes Verfahren jeweils der Mittelwert M sowie die empirische Standardabweichung S nach 20 Durchläufen angegeben.

Die Auswertung zeigt, dass für die Verarbeitung und Integration des gewählten Tiefenbildes in eine „leere“ Karte sowie deren Speicherung in eine .obj-Datei, eine Zeitersparnis von 69 % gegenüber dem bisherigen Verfahren erreicht wurde. Der zeitlich größte Anteil daran kommt dem Speichern der Karte als .obj-Datei (t_3) zu. Bei einem Scan-Vorgang

[1] .obj ist ein von Wavefront Technologies entwickeltes offenes Dateiformat zum Speichern von dreidimensionalen geometrischen Formen, welches 1989 erstmals veröffentlicht wurde.

des Roboters werden prinzipiell die Tiefenbilder aller acht am Roboter montierten ToF-Kameras ausgewertet. Diese Auswertung läuft seriell ab, sodass generell, auch bei acht zu integrierenden Bildern, eine relative Zeitersparnis für die Verarbeitung und Integration dieser Bilder (t_2) von 33 % angenommen werden kann. Da die Karte jedoch normalerweise frühestens am Ende der Scans aller Sensoren, oder erst nach weiteren Scann-Vorgängen, in eine Datei gespeichert wird, kann dann nicht mehr von einer Zeitersparnis von 66 % für das gesamte Verfahren ausgegangen werden. Je mehr Messungen durchgeführt und in die Karte integriert werden, bevor diese als Datei gespeichert wird, desto mehr wird sich die Zeitersparnis von t_{ges} zu t_2 verschieben.

Die Werte der relativen Zeitersparnis für t_2 können abweichen, abhängig davon, ob und welche Daten in der Karte an den Stellen, an denen neue Daten integriert werden sollen, bereits vorhanden sind. Stehen aus der neuen Messung beispielsweise nur Daten geringerer Auflösung, als der an entsprechenden Stellen in der Karte vorhandenen, zur Verfügung, werden diese, wie in 5.6.2 beschrieben, verworfen. Dies resultiert in einer höheren Zeitersparnis, da im bisherigen Verfahren, bedingt durch fehlende Distanzinformation, neue Daten immer integriert werden und eine Differenzierung nicht stattfinden kann.

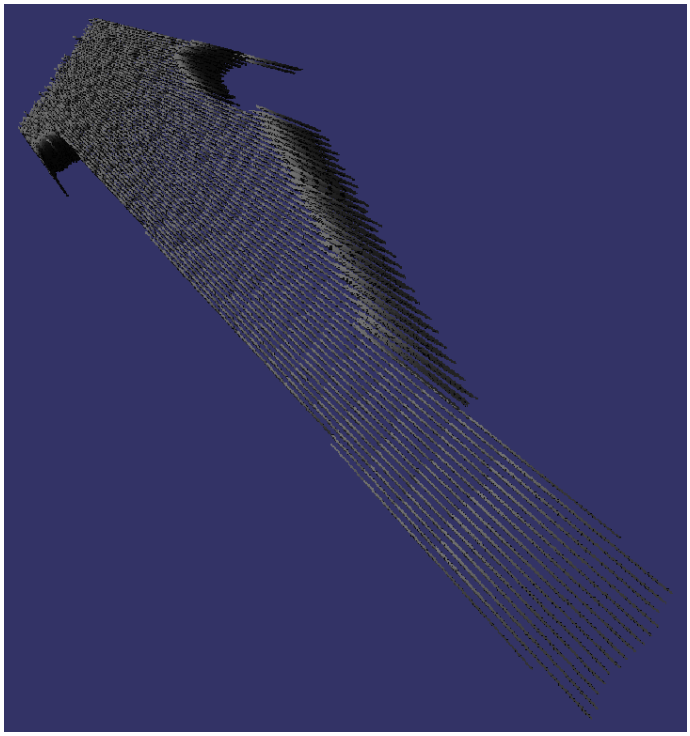
Die Zeit zur Initialisierung der Karte t_1 ist der Vollständigkeit wegen erfasst, macht sich jedoch in der Gesamtzeit des Versuchs kaum bemerkbar. Dieser Vorgang wird, im Gegensatz zur Aktualisierung der Karte (t_2), einmalig durchgeführt, wodurch sich die Zeit t_1 nicht aufsummiert.

Die Differenz beim Speicherbedarf der Karte fällt mit einem relativen Wert von 96 % sehr hoch aus. In diesem Versuch wurde nur ein Tiefenbild in die Karte integriert. Der Speicherbedarf von 65 MB, bei Verwendung des bisherigen Verfahrens, stellt für die hier verwendete Hardware noch kein Problem dar. Da typische Karten jedoch einen deutlich größeren Raum, als den hier durch eine Messung repräsentierten, abbilden, kann die Speicherintensitiät des bisherigen Verfahrens zum Problem werden. Die Auflösung der Karte muss demnach bei größeren Karten entsprechend gering gewählt werden. Das neue Verfahren arbeitet durch die dynamische Auflösung sehr speichereffizient, da es Daten nur in den Bereichen, in denen es die Qualität der Daten rechtfertigt, entsprechend hoch auflöst. Auch größere Karten können mit diesem Verfahren bei hoher Maximalauflösung erstellt werden.

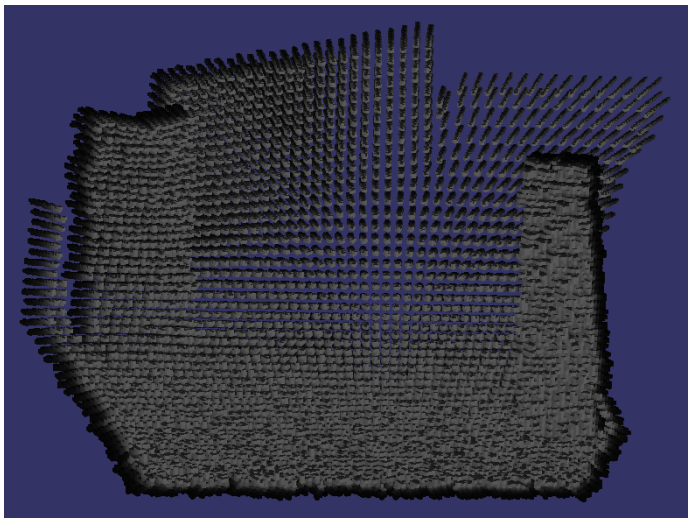
Um eine Karte nach Aufnahme eines Tiefenbildes mit einem *ToF-Sensor höherer Messgenauigkeit* vergleichsweise darzustellen, wurde der Versuch außerdem mit einer veränderten Sensorcharakteristik durchgeführt:

Der Sensorparameter c wurde dazu von $3,61$ auf $3,00$ verringert. Die resultierende Karte, bei Verwendung des bisherigen Verfahrens, ist in Abbildung 6.5 dargestellt, die Karte bei Verwendung des neuen, in Abbildung 6.6. Die geringere Sensorunsicherheit ist in der Karte des bisherigen Verfahrens, vor allem in der Draufsicht gut an den dargestellten, auf einem Strahl angeordneten, Voxel um einen Reflexionspunkt zu erkennen. Es sind jeweils

weniger und die Unterschiede der Besetzungswahrscheinlichkeiten benachbarter Voxel sind größer, da die Unsicherheit kleiner ist und die Besetzungswahrscheinlichkeit schneller, mit zunehmender Distanz zum gemessenen Reflexionspunkt, abnimmt. Das neue Verfahren modelliert diese höherwertigen Daten automatisch dadurch, dass die entsprechenden Bereiche höher aufgelöst werden, indem die Octree-Elemente auf geringeren Levels eingefügt werden.

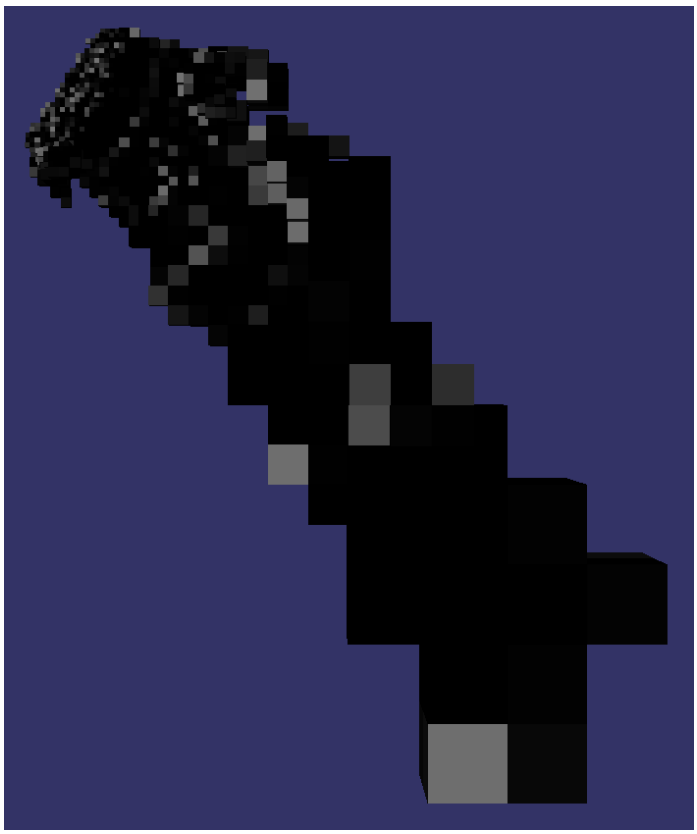


(a) Draufsicht.

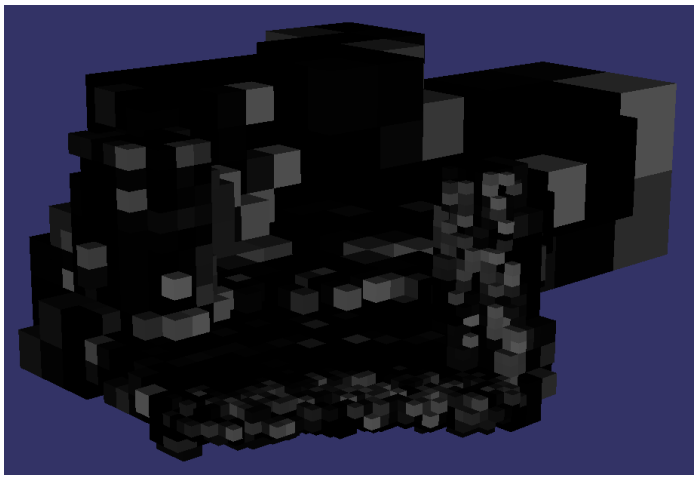


(b) Seitenansicht.

Abbildung 6.3: Resultierende Karte mit bisherigem Verfahren. Dargestellt sind alle Voxel mit einer Besetzungswahrscheinlichkeit größer 50 %.

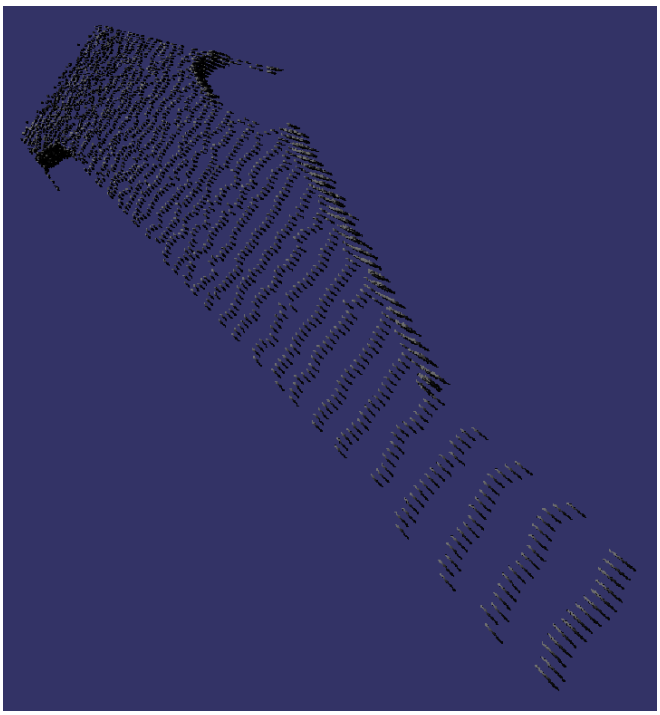


(a) Draufsicht.

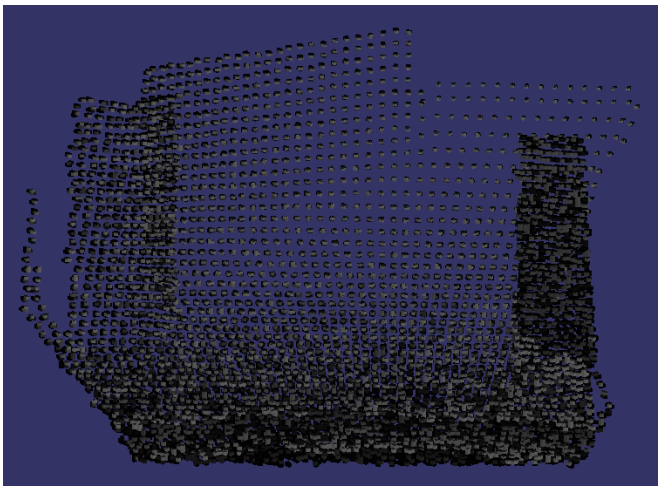


(b) Seitenansicht.

Abbildung 6.4: Resultierende Karte mit neuem Verfahren. Dargestellt sind alle Octree-Elemente mit einer Besetzungswahrscheinlichkeit größer 50 %.

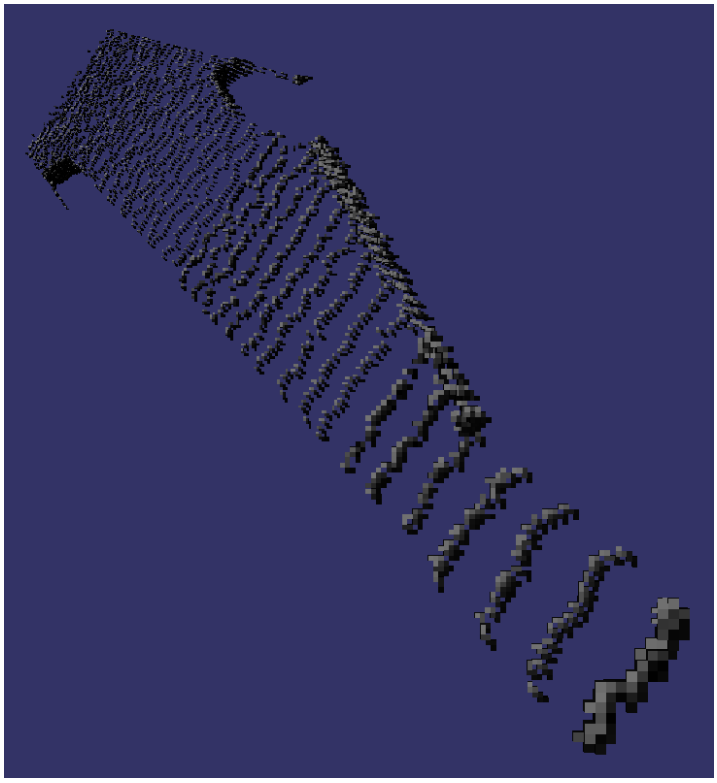


(a) Draufsicht.

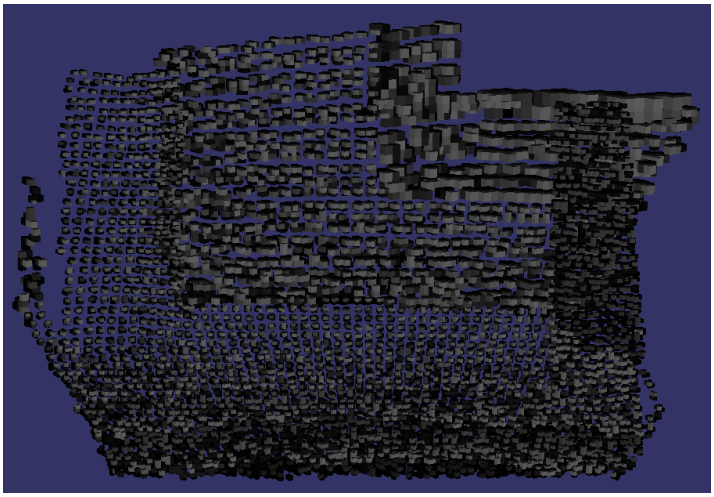


(b) Seitenansicht.

Abbildung 6.5: Resultierende Karte mit bisherigem Verfahren bei Simulation eines Sensors mit verringertem Sensorparameter c = 3,00. Dargestellt sind alle Voxel mit einer Besetzungswahrscheinlichkeit größer 50 %.



(a) Draufsicht.



(b) Seitenansicht.

Abbildung 6.6: Resultierende Karte mit neuem Verfahren bei Simulation eines Sensors mit verringertem Sensorparameter c = 3,00. Dargestellt sind alle Octree-Elemente mit einer Besetzungswahrscheinlichkeit größer 50 %.

6.3 Aufbau einer Karte während der Fahrt des Roboters

Um das entwickelte Verfahren, zum Zweck des kontinuierlichen Kartenaufbaus, in der vorhandenen Simulationsumgebung, mit dem bisherigen Verfahren vergleichen zu können, muss eine geringere maximale Auflösung, als bei den Versuchen zuvor, gewählt werden. Bei zu hoher Auflösung läuft, bei Verwendung des bisherigen Verfahrens, während der Kartenerstellung, der Arbeitsspeicher, auf dem ein Teil der Karte ausgelagert ist, über. Als maximale Auflösung wurden deshalb 50 mm gewählt.

Die Regel zur Erhöhung der Level aus Abschnitt 5.5 legt eine Erhöhung bei einer Überschreitung der Standardabweichung um $\frac{1}{3}$ der aktuellen Octree-Element Kantenlänge fest. Eine Levelerhöhung auf Level-1 würde demnach, bei der hier gewählten max. Auflösung von 50 mm, erst bei einer Entfernung von 1744 mm stattfinden. Dies entspricht der Qualität der Daten, da erst ab dieser Entfernung die Standardabweichung von 16,67 mm und somit $\frac{1}{3}$ der Länge der Octree-Element-Kante, erreicht wird. Jedoch kann damit der Verlauf der Levelerhöhungen schlecht visualisiert und das Verfahren demonstriert werden. Die Regel wurde für diesen Versuch deshalb abgeändert und die Schranke von $\frac{1}{3}$ auf $\frac{1}{40}$ abgesenkt. Mit diesem Wert findet eine Levelerhöhung bei folgenden Entfernungen statt: 83,33 mm, 300,0 mm, 1083 mm und 3900 mm.

Wenn nur das hier entwickelte Verfahren benutzt werden soll, kann eine deutlich höhere Auflösung, unter Verwendung der Levelerhöhungsvorschrift aus Abschnitt 5.5, benutzt werden. Lediglich der Vergleich mit dem bisherigen Verfahren erfordert aus oben genannten Gründen diese Anpassung.

Die Karte wird mit folgenden Parametern erstellt:

- Kantenlänge eines Octrees: 3200 mm
- Maximale Auflösung (Kantenlänge eines Voxels): 50 mm
- Mindestabstand zum Sensor: 12 mm
- Maximale Länge des Sensorstrahls: 7000 mm

Abbildung 6.7 zeigt die initiale Position des mit entsprechender Sensorik ausgestatteten Roboters in der Simulationsumgebung. Der für die acht ToF-Kameras, aufgrund ihrer speziellen Anordnung und generellen Charakteristika, theoretisch sichtbare Bereich ist rot eingefärbt. Bei dem hier durchgeführten Versuch wird der Roboter auf der Stelle um jeweils 45° gedreht, bis er sich wieder in seiner Ausgangspose befindet. Nach jeder Drehung erfolgt eine Messung aller acht ToF-Kameras, deren aufgenommenen Tiefenbilder in die Karte integriert werden.

In Abbildung 6.8 sind die entstandenen Karten dargestellt, wobei Abbildung 6.8(a) die Karte bei Verwendung des bisherigen Verfahrens und 6.8(b) des neuen Verfahrens zeigt.

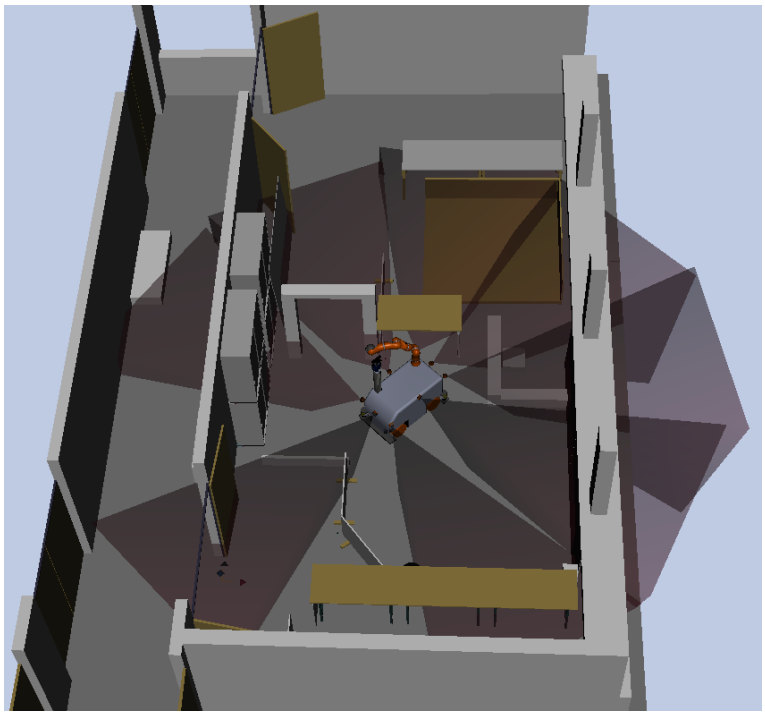


Abbildung 6.7: Initiale Roboterposition in der Simulationsumgebung. Der für die Sensoren theoretisch sichtbare Bereich ist rot eingefärbt. Während einer vollständigen Drehung auf der Stelle in 45° Schritten mit jeweils sich anschließender Messung, wird eine Karte der Umgebung erstellt.

Der Aufbau ist, wie aus den vorherigen Beschreibungen hervorgeht, von zu erwartender Struktur.

Die Laufzeiten der Mapping-Verfahren werden hier nicht isoliert betrachtet, da auch andere notwendige Komponenten bei der Versuchsdurchführung, wie insbesondere die SLAM-Komponente Rechenzeit beanspruchen. Der Vergleich der Gesamtlaufzeiten sowie die Speicherbelegung der erstellten Karten als .obj-Dateien $stor$, der beiden Verfahren ist in Tabelle 6.2 dargestellt. $t_{ges,ideal}$ bezeichnet dabei die Dauer des Durchlaufs ohne die nötige Zeit zur Bewegung des Roboters, t_{ges} die gesamte Dauer inklusive der Bewegung.

Durch die in der Simulationsumgebung modellierten Messunsicherheiten sowie Fehler der Positionsschätzung der SLAM-Komponente, weichen die Werte der Laufzeiten für jeden Durchlauf ab. Zur Datenerhebung wurden deshalb 10 Durchläufe durchgeführt und in der Tabelle jeweils die Mittelwerte und ihre empirischen Standardabweichungen angegeben.

Während der am Institut verwendeten Exploration mit diesem Roboter wird, nach dem Anfahren einer Position im Raum, welcher durch vorangegangene Bewertung verschie-

Tabelle 6.2: Auswertung von Laufzeiten und Speicherbedarf der beiden Verfahren für die Verarbeitung und Integration mehrerer Tiefenbilder in eine Karte.

	bisheriges Verfahren		**neues Verfahren**		**Differenz**	
	M	S	M	S	ΔM abs.	ΔM rel.
$t_{\text{ges,ideal}}$	37,908 s	2,2674 s	26,5976 s	2,1973 s	11,310 s	29,8 %
t_{ges}	69,295 s	2,2674 s	57,9850 s	2,1973 s	11,310 s	16,3 %
$stor$	4.292,5 kB	237,84 kB	1.190,3 kB	179,39 kB	3.102,2 kB	72,3 %

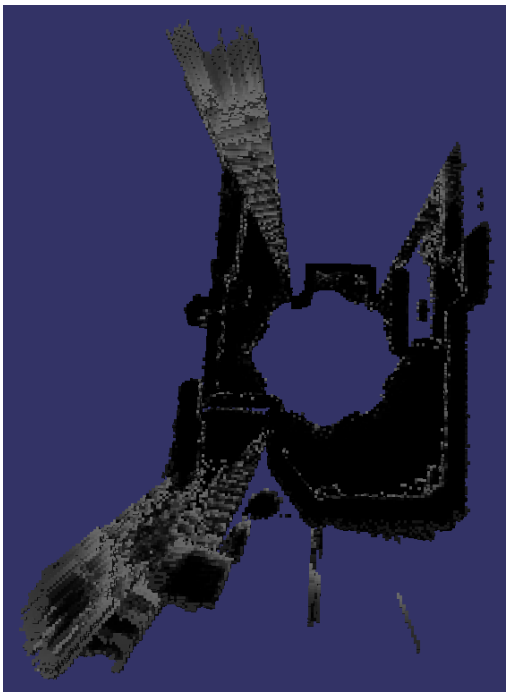
dener möglicher Positionen bestimmt wurde, ein solcher Scan durchgeführt [13]. Durch die schrittweise Drehung mit den damit verbundenen Messungen kann der Raum um den Roboter ausreichend vermessen werden. Durch die Charakteristika der verwendeten Sensoren ist, wie durch die Sichtbarkeitsbereiche aus Abbildung 6.7 aufgezeigt, keine vollständige Abdeckung des Bereichs um den Roboter durch Vermessung ohne dessen Drehung möglich.

Der Vorgang dauert, unter Verwendung des neuen Verfahrens, mit 58 Sekunden für dieses Beispiel 16 % kürzer. Die entstandene Karte ist mit einer Speicherbelegung von 1,16 MB 72 % kleiner. Durch die Dauer der Bewegung und anderer nötiger Prozesse, neben dem Mapping-Prozess selbst, fällt die relative Zeitersparnis durch das neue Verfahren mit 16 % nicht so groß aus, wie bei dem vorherigen Versuchsfall, als nur ein Tiefenbild verarbeitet wurde. Die Laufzeit zur Kartenerstellung stellt hier nur einen (kleinen) Teil der Gesamtlaufzeit dar. Das liegt außerdem daran, dass die Auflösung der Karte, um auch das bisherige Verfahren lauffähig zu halten, so gering gewählt wird und die Regel zur Levelerhöhung angepasst werden musste. Bei höherer Auflösung, welche für diesen Fall generell wünschenswert ist um Objekte genauer zu modellieren, steigt der Anteil der benötigten Rechenzeit zur Kartenerstellung an der Gesamtzeit. Der Anstieg der Rechenzeit beim bisherigen Verfahren steigt dabei in etwa kubisch mit der Auflösung an. Ein zu modellierender Volumenwürfel x-facher Kantenlänge beinhaltet die x^3-fache Anzahl an Voxeln, welche im bisherigen Verfahren einzeln verarbeitet werden.

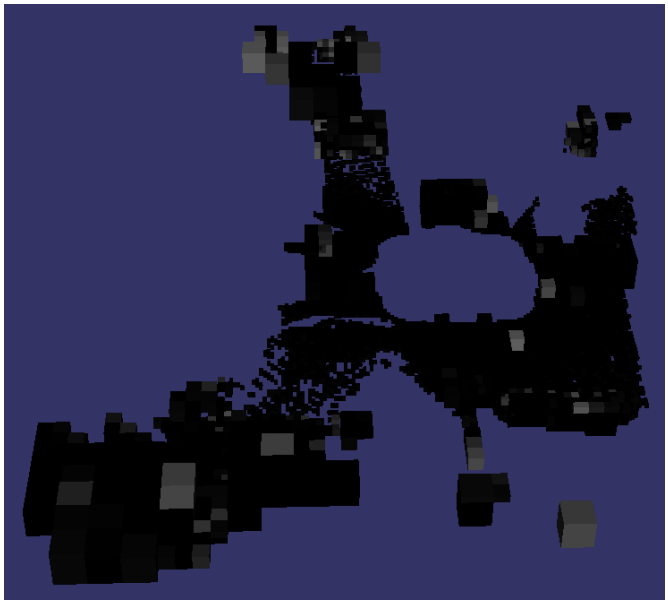
Einen großen Anteil an der Gesamtlaufzeit beansprucht die Bewegung des Roboters, welcher, unabhängig vom gewählten Verfahren, eine konstante Zeit benötigt. Ohne Berücksichtigung dieser Bewegung würde sich der Zeitvorteil mit 30 % auf fast den doppelten Wert belaufen.

Ein weiterer Aspekt, warum die relative Zeitersparnis sich im Vergleich zum vorherigen Versuch verringert hat, liegt darin, dass die Karte nur einmal, nach Integration aller Messungen, als Datei abgespeichert wird. Es wird also t_2 aus 6.1 acht mal ausgeführt, jedoch t_3 nur einmal am Ende. t_3 wird zwar absolut länger dauern, da nach Integration mehrerer Messungen die Größe der Karte steigt, jedoch wäre ein Speichervorgang nach jeder in-

tegrierten Messung im Gesamten deutlich länger. Eine relative Zeitersparnis gemäß dem Wert von t_{ges} war also auch aufgrund dieser Tatsache nicht zu erwarten.



(a) Resultierende Karte mit bisherigem Verfahren.



(b) Resultierende Karte mit neuem Verfahren.

Abbildung 6.8: Resultierende Karten nach Vermessungsvorgang unter Drehung des Roboters um seine Hochachse. Dargestellt sind alle Octree-Elemente mit einer Besetzungswahrscheinlichkeit größer 50 %.

6.4 Mapping von Testräumen durch einen mit Stereokameras ausgestatteten mobilen Roboter

Das Verfahren wurde zusätzlich zu den Versuchen auf dem omniRob mit ToF-Kameras, auf einem mit Stereokameras ausgestatteten Pioneer 3-AT Roboter nach [11] getestet, welcher in Abbildung 6.9 dargestellt ist. Zur Versuchsdurchführung wurde eine zuvor aufgenommene und gespeicherte Fahrt durch Laborräume verwendet und diese in einer Simulationsumgebung, unter Verwendung der beiden Mapping-Verfahren, abgespielt.

Als Sensorik stehen zwei verschiedene Stereokamera-Systeme zur Verfügung:

1. Ein Schmalwinkelsystem:

   Brennweite der Linsen $f = 5$ mm, Guppy F-080B Kameras (1/3 Zoll Chip-Größe, Auflösung: 1032 x 778)

2. Ein Weitwinkelsystem:

   Brennweite der Linsen $f = 1,28$ mm, Gruppy PRO F-125B Kameras (1/3 Zoll Chip-Größe, Auflösung: 1292 x 964)

Der effektive Sichbarkeitsbereich der Kameras wird durch die maximal verarbeitbare Bildgröße des FPGAs auf $50,6^\circ$ für das Schmalwinkelsystem und $112,8^\circ$ für das Weitwinkelsystem limitiert.

Für den hier durchgeführten Versuch wird das Schmalwinkelsystem verwendet.

Das veranschlagte Fehlermodell der Sensoren ist in Abbildung 6.10 dargestellt. Der durchschnittliche Stereo-Fehler ΔP_z ist dort über der von den Sensoren gemessenen Distanz z für das Weitwinkelsystem (grün gestrichelte Linie), sowie das Schmalwinkelsystem (blaue Linie) dargestellt. Aus diesem Modell ergeben sich, nach der Berechnungsvorschrift aus Abschnitt 5.5, diejenigen Distanzen, ab welchen jeweils das Level der Octree-Elemente, bei der Datenintegration in die Karte, erhöht.

Die Karte wird mit folgenden Parametern erstellt:

- Kantenlänge eines Octrees: 51200 mm
- Maximale Auflösung (Kantenlänge eines Voxels): 50 mm
- Mindestabstand zum Sensor: 710 mm
- Maximale Länge des Sensorstrahls: 7000 mm

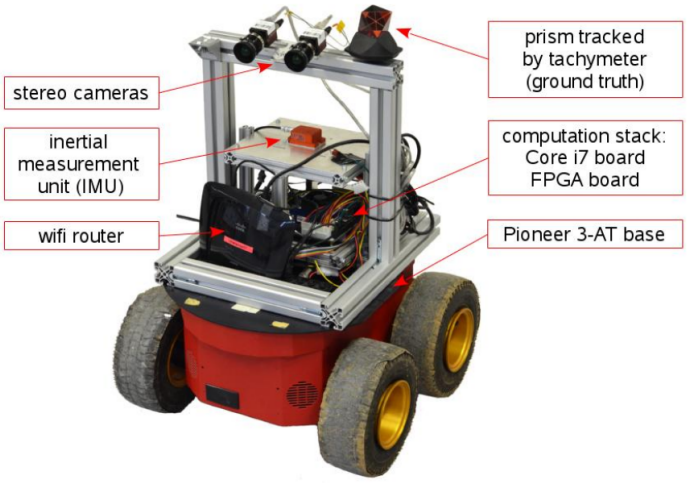


Abbildung 6.9: Aus [11]. Pioneer 3-AT Roboter mit entsprechender Ausstattung.

Der Roboter bewegt sich nach einem vordefinierten Bewegungsprofil. Die Geschwindigkeit des Roboters ist unabhängig der erfassten und verarbeiteten Daten und kann somit nicht an die Verarbeitungsgeschwindigkeit der auflaufenden Messdaten angepasst werden. So kann es passieren, dass mehr neue Daten erfasst werden, als durch das Mapping-Verfahren verarbeitet und in die Karte integriert werden können. Um diesem Problem zu begegnen existiert ein Datenpuffer, welcher die akquirierten Tiefendaten der Sensoren zwischenspeichert. Das Mapping-Verfahren greift auf diesen Speicher zu, um die dort abgelegten Daten in die Karte zu integrieren. Bei dem bisher verwendeten Verfahren kommt es vor, dass dieser Zwischenspeicher überläuft. Dabei werden die jeweils ältesten Daten aus dem Speicher verworfen, sodass neue zwischengespeichert werden können. Dies ist notwendig, da der Zwischenspeicher nicht beliebig groß gewählt werden kann. Insbesondere für lange Messfahrten müsste der Zwischenspeicher, sofern dies technisch möglich ist, so groß gewählt werden, dass dies in keinem sinnvollen Kosten/Nutzen-Verhältnis stünde.

Abbildung 6.11 zeigt die zu vermessende Umgebung in der Simulation mit rot eingezeichneten Pfad des Roboters, welcher während der Vermessung abgefahren wird.

In Abbildung 6.12 sind die entstandenen Karten dargestellt, wobei Abbildung 6.12(a) die Karte bei Verwendung des bisherigen Verfahrens und 6.12(b) des neuen Verfahrens zeigt. Octree-Elemente werden in den dargestellten Karten der verwendeten Simulationsumgebung stets in Voxel aufgeteilt angezeigt, obgleich sie so nicht in der Karte gespeichert sind. Durch die spezielle Anordnung der Voxel eines Octree-Elements sind diese jedoch dennoch gut zu erkennen. Die unterschiedlichen Farben der Voxel visualisieren die Höhenlage der Elemente in der Karte.

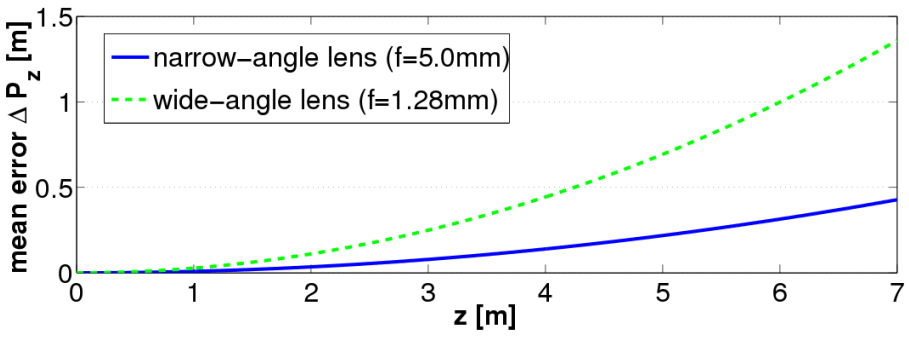


Abbildung 6.10: Aus [11]. Durchschnittlicher veranschlagter Stereo-Fehler ΔP_z über der gemessenen Distanz z für das Weitwinkelsystem (grün gestrichelte Linie), sowie das Schmalwinkelsystem (blaue Linie).

Tabelle 6.3: Auswertung von verarbeiteten Tiefenbildern, Anzahl an Voxeln in der Karte und Speicherbedarf der beiden Verfahren

	bisheriges Verfahren		**neues Verfahren**		**Differenz**	
	M	S	M	S	ΔM abs.	ΔM rel.
N_{di}	677,13	18,166	1995,2	84,738	1.318,1	195 %
N_{vox}	75.963	71,846	154.677	244,92	78.714	104 %
$stor$	833.954 kB	843,37 kB	174.046 kB	316,68 kB	659.908 kB	79,13 %

In Tabelle 6.3 ist dargestellt, wie viele Tiefenbilder während der Fahrt in die Karte integriert werden konnten N_{di}. Außerdem in der Tabelle dargestellt, ist die Anzahl der Voxel der entstandenen Karte N_{vox}, wobei Octree-Elemente mit Level $L \in \mathbb{N} = \{1, 2, ...\}$ so gezählt werden, als ob sie mit Voxeln gefüllt wären. Die Speichergröße $stor$ der Karte als .obj-Datei korreliert nicht direkt mit dieser Anzahl an Voxeln der Karte, da diese beim neuen Verfahren nur theoretisch aus Octree-Elementen höheren Levels bestimmt wurden und außerdem Voxel und Octree-Elemente in der Octree-Datenstruktur bei gleichem Inhalt zusammengefasst gespeichert werden. Es sind in der Tabelle die Mittelwerte und die empirische Standardabweichung nach 10 Durchläufen angegeben.

Die Auswertung zeigt, dass mit dem neu implementierten Verfahren deutlich weniger Daten aus dem Zwischenspeicher verworfen werden müssen, da das Mapping weniger Zeit in Anspruch nimmt. Es konnte durch eine Verarbeitung mit dem neuen Mapping-Verfahren von durchschnittlich 1995 Tiefenbildern pro Durchlauf, im Vergleich zu 677 bisher, eine Steigerung um 195 % erreicht werden. Auch die theoretische Anzahl an Voxeln wurde gesteigert, von $76 \cdot 10^3$ auf $155 \cdot 10^3$, was einer Steigerung um 104 % entspricht. Es stehen

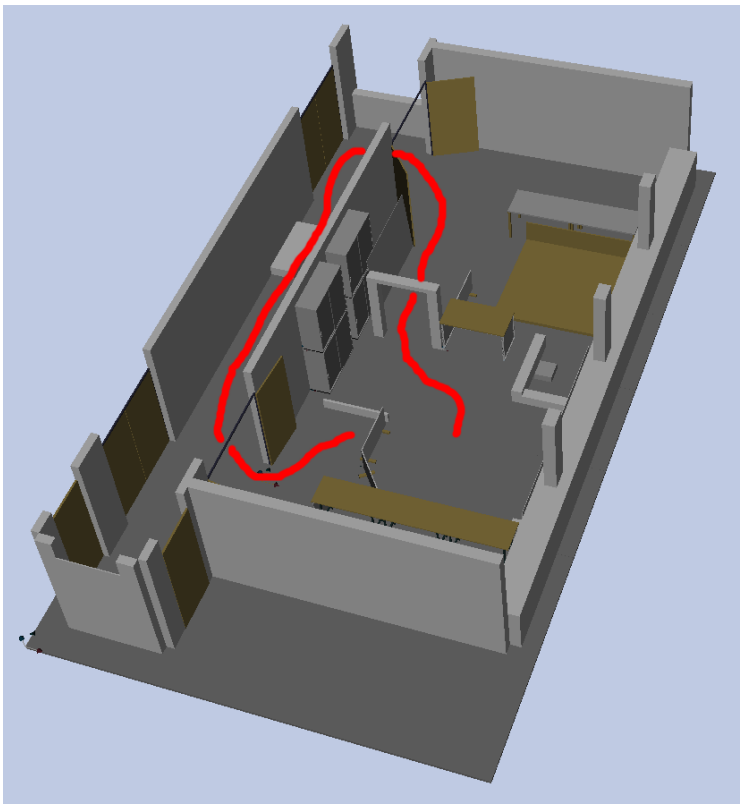
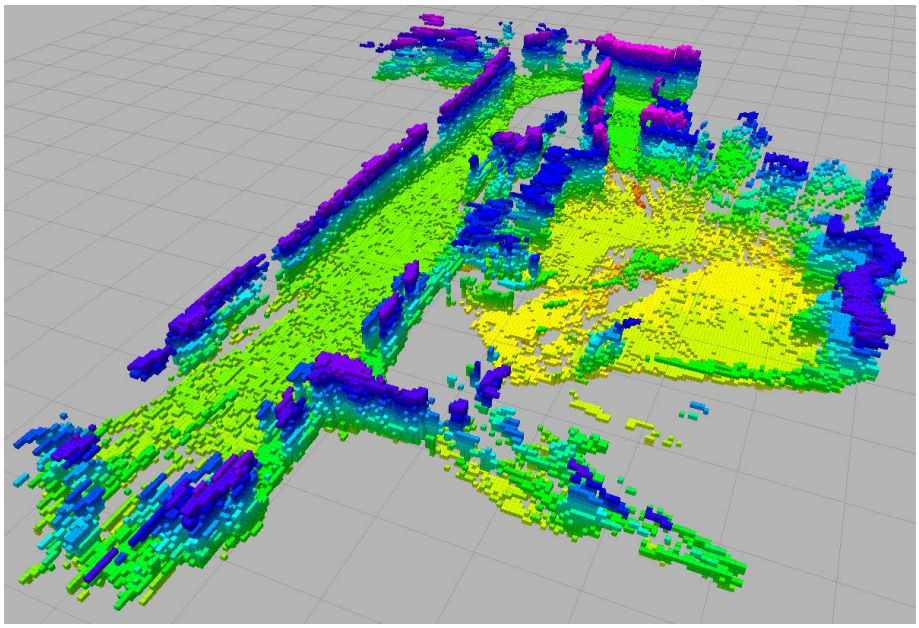


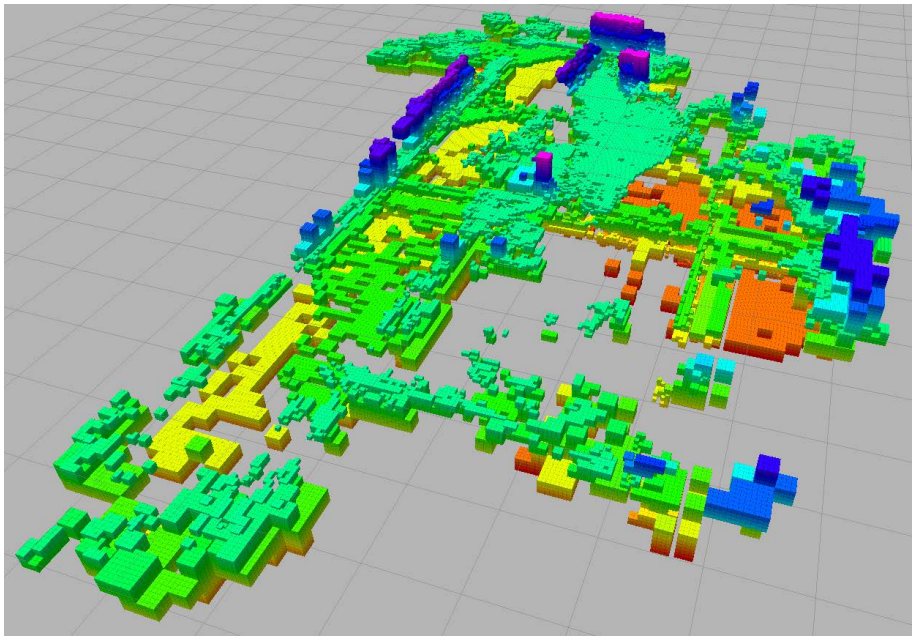
Abbildung 6.11: Simulationsumgebung mit rot eingezeichnetem Pfad des Roboters, welcher während der Vermessung abgefahren wird.

also für mehr Volumenelemente des vermessenen Raumes, repräsentiert durch Voxel, in der Karte integrierte Messdaten zur Verfügung. Der Speicherbedarf als Datei wurde dabei um 79 % gesenkt.

Anmerkung: Bei der Karte mit dem neuen Verfahren wird der Boden nicht korrekt angezeigt. Dies liegt daran, dass Elemente, die sich (teilweise) unterhalb des Bodens befinden, gelöscht werden. Die Höhe des Bodens war vor der Vermessungsdurchführung bekannt und wurde als konstant angenommen. Deshalb wurde eine Funktionalität implementiert um Daten, welche aufgrund dieses Vorwissens, fälschlicherweise dort liegen, zu verwerfen. Bei dem neuen Verfahren entstehen größere Octree-Elemente, sodass diese, aufgrund ihrer Ausdehnung, bei Messung eines Reflexionspunktes oberhalb des Bodens, auch unterhalb des Bodens Raum allozieren können und deshalb verworfen werden. Diese Funktionalität ist, bei Verwendung von dynamischen Octree-Element-Größen, entsprechend anzupassen.



(a) Resultierende Karte mit bisherigem Verfahren.



(b) Resultierende Karte mit neuem Verfahren.

Abbildung 6.12: Resultierende Karten nach Vermessung der Testräume durch den Roboter aus Abb. 6.9. Darzustellende Octree-Elemente sind stets vollständig in Voxel zerlegt visualisiert. Die unterschiedliche Einfärbung der Voxel repräsentiert ihre Höhenlage in der Karte.

7 Zusammenfassung und Ausblick

Am Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt wurde Mapping auf einer Voxelkarte in dem probabilistischen Raum nach [10] unter Verwendung des Dynamic Octree Raumes nach [80] durchgeführt. Jenes Mapping-Verfahren sollte in dieser Arbeit weiterentwickelt werden, indem dazu in aktuellen Verfahren nach Ideen gesucht und eine solche adaptiert, implementiert und getestet wird.

Es wurde in der Arbeit, nach einer generellen Übersicht über diverse Lösungsansätze zum Problem des Mappings, eine Übersicht über aktuelle Mapping-Verfahren gegeben und deren Charakteristika herausgearbeitet. Als Anregung daraus, zur Weiterentwicklung des momentan am Institut verwendeten Verfahrens, wurde die Idee aufgenommen, Daten in der Karte in verschiedenen Auflösungsstufen zu speichern, welche von der jeweiligen Qualität der in die Karte zu integrierenden Daten abhängig sind. Als Qualitätskriterium wurde dabei die Messunsicherheit des Sensors benutzt, welche mit größerer Distanz zwischen Sensor und Messpunkt zunimmt.

Eine solche Funktionalität konnte erfolgreich entwickelt, auf das bisherige Verfahren angewandt und getestet werden. Die Auflösung von Karten wird jetzt während ihrer Erstellung dynamisch angepasst, indem die Größe der Elemente eines gleichmäßigen Gitters, während des Einfügens neuer Daten, entsprechend gewählt wird. Daten, die aus unsicheren Messungen resultieren, die Abweichungen der Messdaten des Sensors also entsprechend groß sind, werden in größeren Elementen gespeichert, als Daten aus Messungen mit geringerer Unsicherheit. Je größer die Entfernung der gemessenen Reflexionspunkte zum Sensor, desto größer ist deren Streuung oder Abweichung bzgl. der gemessenen Entfernung anzunehmen. Um diese Abhängigkeit der gemessenen Entfernung zur Abweichung dieser Werte für den verwendeten Sensor zu modellieren, wurde ein experimentell erprobtes Modell verwendet und dessen Parameter entsprechend der verwendeten Sensoren gesetzt.

Es konnte gezeigt werden, dass bei Verwendung des hier weiterentwickelten Verfahrens mit multiplen Auflösungsstufen, bei Wahl geeigneter Parameter, die benötigte Zeit zum Aufbau einer Karte deutlich reduziert werden konnte. Auch der Speicherbedarf der entstehenden Karte wurde dabei reduziert. In einem repräsentativen Versuchsfall zur Integration eines Tiefenbildes in eine Karte wurde die Rechenzeit um 33 % reduziert. Die Zeit zur Integration des Tiefenbildes sowie der anschließenden Speicherung der Karte als obj-Datei reduzierte sich um 69 %. Der Speicherbedarf der erzeugten Datei wurde dabei um 96 %

verringert. Die entstandene Karte ist dabei für den Zweck der Exploration, trotz der großteils geringeren Auflösung, ausreichend genau. Die Versuche wurden in einer Simulationsumgebung auf dem omniRob von Kuka mit ToF-Kameras als Tiefensensoren durchgeführt. Auf diesen so ausgestatteten Roboter soll das Verfahren später portiert werden.

Darüber hinaus, um die Verwendbarkeit auch mit anderen Systemen zu demonstrieren, wurde ein Versuch auf einem mit Stereokameras ausgestatteten Roboter erfolgreich durchgeführt. Dabei konnte gezeigt werden, dass bei Verwendung des hier weiterentwickelten Mapping-Verfahrens mit entsprechenden Parametern, die Messdaten schneller in die Karte integriert werden können. Dadurch könnte der Roboter den Weg bei gleicher Geschwindigkeit des Kartenaufbaus schneller abfahren, oder es könnten mehr Daten während der unveränderten Fahrt integriert werden. Der Speicherbedarf der entstandenen Karte ist auch hier geringer.

Ein weiterer Vorteil der neuen Funktionalität liegt darin, dass die maximale Auflösung der Karte sehr hoch gewählt werden kann, ohne dass diese für die gesamte Karte benutzt werden muss. So können interessante Bereiche von einem Roboter nah angefahren und in hoher Auflösung vermessen und gespeichert werden und andere weiter entfernte Bereiche in geringerer Auflösung in die Karte integriert werden. Die gesamte Karte kann somit bzgl. des Speicherbedarfs klein gehalten werden, obwohl sie hoch aufgelöste Bereiche enthält. Dies war zuvor nicht möglich, da die Auflösung der ganzen Karte gleich sein musste.

Das neue Verfahren bietet außerdem die Möglichkeit die Auflösung während des Betriebs manuell auf einen bestimmten Wert zu setzen. Dies kann für Anwendungen geeignet sein, bei denen ein Roboter ferngesteuert wird. Während der Fahrt zu interessanten Objekten kann die Auflösung gering gehalten werden. Sie muss lediglich ausreichend zur Navigation und Kollisionsvermeidung sein. Für den Scan bestimmter Objekte kann dann die jeweils nötige (höhere) Auflösung gewählt werden. Es entsteht somit eine zusammenhängende Karte, in der jeder Bereich in einer sinnvollen Auflösung dargestellt wird. Im bisherigen Verfahren hätte jede Änderung der Auflösung nur durch den Bau einer neuen Karte bewerkstelligt werden können. Der vermessene Raum wäre dann nicht mehr zusammenhängend in einer Karte gespeichert.

Bei der Integration von neuen Messdaten in einem Bereich der Karte, indem bereits Daten vorhanden sind, werden die neuen Daten nur integriert, wenn ihre Qualität gleich- oder höher ist. Es wäre zu testen, wie sich eine gewichtete Integration auch von Daten geringerer Qualität auf die Gesamtqualität der Karte auswirken und sich dabei die Rechenzeit verändern würde.

Es wurde angedacht, die bisher verwendeten ToF-Kameras zukünftig durch neuere mit höherer Auflösung zu ersetzen. Dies könnte bei Verwendung des bisherigen Mapping-Verfahrens ein Problem darstellen, da für eine Karte, die den hoch aufgelösten Daten neuer Sensoren gerecht werden soll, eine entsprechend hohe Auflösung der Karte gewählt werden müsste. Wie ein in dieser Arbeit durchgeführter Versuch zeigt, kann eine

Erhöhung der Auflösung dabei problematisch sein, da der Arbeitsspeicher bei verwendeter Hardware unter Umständen überläuft und außerdem das Verfahren extrem langsam wird. Das Verfahren mit Erweiterung der multiplen Auflösungsstufen würde diesem Problem begegnen, sodass die höhere Auflösung neuer Sensoren einfach genutzt werden könnte, ohne dabei den Speicherbedarf der Karten zu stark zu erhöhen.

Das Verfahren kann nach den erfolgreichen Tests in der Simulationsumgebung auf die reale Hardware portiert und dort weiter evaluiert werden. Des weiteren ist ein Umbau der Sensorik auf höher auflösende Sensoren nun leichter möglich. Dadurch ließen sich neue Anwendungsgebiete erschließen, welche hoch aufgelöste Daten in der Karte erfordern.

Literaturverzeichnis

1. S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S.M. Seitz und R. Szeliski. Building rome in a day. In *Communications of the ACM*, 54(10), S. 105–112, 2011. URL http://dl.acm.org/citation.cfm?id=2001293.

2. M. Agrawal, K. Konolige und R.C. Bolles. Localization and mapping for autonomous navigation in outdoor terrains: A stereo vision approach. In *IEEE Workshop on Applications of Computer Vision, 2007.*, S. 7–7. IEEE, 2007.

3. G. Aretz. *Sonar in Theorie und Praxis für Unterwasser-Anwendungen*. Verlag-Haus Monsenstein und Vannerdat, 2006.

4. Microsoft Artikel. Kinect for Windows features. 2014. URL http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx. Aufgerufen am 13.10.2014.

5. Microsoft Developer Network Artikel. Kinect for Windows Sensor Components and Specifications. 2014. URL http://msdn.microsoft.com/en-us/library/jj131033.aspx. Aufgerufen am 13.10.2014.

6. J. Aue, D. Langer, B. Muller-Bessler und B. Huhnke. Efficient Segmentation of 3D LIDAR Point Clouds Handling Partial Occlusion. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, S. 423–428. IEEE, 2011. doi:10.1109/IVS.2011.5940442. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5940442.

7. P.J. Besl und N.D. Mckay. A Method for Registration of 3-D Shapes. In *The IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2), S. 239–256, Februar 1992. ISSN 0162-8828. doi:10.1109/34.121791. URL http://dx.doi.org/10.1109/34.121791.

8. G. Blais und M.D. Levine. Registering Multiview Range Data to Create 3D Computer Objects. In *The IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 17(8), S. 820–824, 1995. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=400574.

9. J.F. Blinn und M.E. Newell. Texture and reflection in computer generated images. In *Communications of the ACM*, 19(10), S. 542–547, 1976. URL http://dl.acm.org/citation.cfm?id=360353.

10. T. Bodenmüller. *Streaming Surface Reconstruction from Real Time 3D Measurements*. Dissertation, Technische Universität München, Munich, 2009.

11. C. Brand, M.J. Schuster, H. Hirschmuller und M. Suppa. Stereo-vision based obstacle mapping for indoor/outdoor SLAM. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14–18, 2014*, S. 1846–1853. IEEE, Chicago, Illinois, USA, September 2014. URL `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6926644`.

12. S.D. Burd. *Systems Architecture*. 6. Auflage. Cengage Learning, 2010.

13. C.P.U. Buschor. *Next-Best-View-Planning auf Voxelkarten zur Exploration durch einen mit Tiefensensoren ausgestatteten mobilen Roboter*. Bachelorarbeit, Technische Universität München, August 2013.

14. T.F. Chan, G.H. Golub und R.J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, S. 30–41. Springer, 1982. URL `http://link.springer.com/chapter/10.1007/978-3-642-51461-6_3`.

15. T. Che, L. Xin, R. Jin, R. Armstrong und T. Zhang. Snow depth derived from passive microwave remote-sensing data in china. In *Annals of Glaciology*, 49(1), S. 145–154, 2008. URL `http://www.ingentaconnect.com/content/igsoc/agl/2008/00000049/00000001/art00024`.

16. Y. Chen und G. Medioni. Object modelling by registration of multiple range images. In *Image and vision computing*, 10(3), S. 145–155, 1992. URL `http://www.sciencedirect.com/science/article/pii/026288569290066C`.

17. A. Chiuso, P. Favaro, H. Jin und S. Soatto. 3-d motion and structure from 2-d motion causally integrated over time: Implementation. In *Computer Vision ECCV 2000*, S. 734–750. Springer, 2000. URL `http://link.springer.com/chapter/10.1007/3-540-45053-X_47`.

18. B. Curless und M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, S. 303–312. ACM, 1996. URL `http://dl.acm.org/citation.cfm?id=237269`.

19. A.H. Daniel Arbuckle und M. Mataric. Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, September 30 - October 4, 2002*, Band 1, S. 409–414. IEEE, Lausanne, Switzerland, September/Oktober 2002. ISBN 0-7803-7398-7. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1041424`.

20. D.M.D. Di Vasto. *Probabilistic Modeling of Dynamic Environments for Mobile Robots*. Dissertation, Albert-Ludwigs-Universitt Freiburg im Breisgau, 2011.

21. A. Dirafzoon, J. Betthauser, J. Schornick, D. Benavides und E. Lobaton. Mapping of unknown environments using minimal sensing from a stochastic swarm. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14–18, 2014*, S. 3842–3849. IEEE, Chicago, Illinois, USA, September 2014. URL `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6926644`.

22. L. Dorst, M. van Lambalgen und F. Voorbraak (Hrsg.). *Reasoning with Uncertainty in Robotics, International Workshop, RUR '95, Amsterdam, The Netherlands, December 4–6, 1995, Proceedings*, Band 1093 von *Lecture Notes in Computer Science*. Springer, 1996. ISBN 3-540-61376-5.

23. H.F. Durrant-Whyte. Uncertain geometry in robotics. In *IEEE Journal of Robotics and Automation*, 4(1), S. 23–31, 1988. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=768`.

24. A. Elfes. A sonar-based mapping and navigation system. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation, San Francisco, California, USA*, Band 3, S. 1151–1156. IEEE, 1986. doi:10.1109/ROBOT.1986.1087534. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1087534`.

25. A. Elfes. Sonar-based real-world mapping and navigation. In *IEEE Journal of Robotics and Automation*, 3(3), S. 249–265, 1987. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1087096`.

26. A. Elfes. *Occupancy grids: A probabilistic framework for robot perception and navigation*. Dissertation, Carnegie Mellon University, 1989.

27. F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers und W. Burgard. An evaluation of the RGB-D SLAM system. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14–18 May, 2012, St. Paul, Minnesota, USA*, S. 1691–1696. IEEE, St. Paul, Minnesota, USA, Mai 2012. ISBN 978-1-4673-1403-9. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6225199`.

28. D.N. Fiedler. *Beiträge zur Analyse, Modellierung und Kalibrierung von Kameras und 3D-Tiefensensoren*. Dissertation, Technische Universität Dortmund, Fakultät für Informatik, 2014. URL `https://eldorado.tu-dortmund.de/handle/2003/33118`.

29. D. Fofi, T. Sliwa und Y. Voisin. A comparative survey on invisible structured light. In *Electronic Imaging 2004*, S. 90–98. International Society for Optics and Photonics, 2004. URL `http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=837538`.

30. V.J. Fowler. Laser scanning techniques. In *Acquisition & Analysis of Pictorial Data*, S. 30–43. International Society for Optics and Photonics, 1974. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1225085.

31. V.J. Fowler und J. Schlafer. A survey of laser beam deflection techniques. In *Applied Optics*, 5(10), S. 1675–1682, 1966.

32. S. Fuchs. *Calibration and multipath mitigation for increased accuracy of time-of-flight camera measurements in robotic applications*. Dissertation, Universitätsbibliothek der Technischen Universität Berlin, 2012.

33. F. Gambino, G. Oriolo und G. Ulivi. Comparison of three uncertainty calculus techniques for ultrasonic map building. In *Aerospace/Defense Sensing and Controls*, S. 249–260. International Society for Optics and Photonics, 1996. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=1019206.

34. J. Geng. Structured-light 3D surface imaging: a tutorial. In *Advances in Optics and Photonics*, 3(2), S. 128–160, 2011. URL http://www.opticsinfobase.org/abstract.cfm?uri=aop-3-2-128.

35. A. Georgopoulos, C. Ioannidis und A. Valanis. Assessing the performance of a structured light scanner. In *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Commission V Symposium.*, Band 38. Citeseer, 2010.

36. D. Hähnel, R. Triebel, W. Burgard und S. Thrun. Map building with mobile robots in dynamic environments. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14–19, 2003, Taipei, Taiwan*, Band 2, S. 1557–1563. IEEE, Taipei, Taiwan, September 2003. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1241816.

37. P. Henry, M. Krainin, E. Herbst, X. Ren und D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *The 12th International Symposium on Experimental Robotics (ISER)*. Citeseer, 2010. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.226.91.

38. A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss und W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. In *Autonomous Robots*, 34(3), S. 189–206, Februar 2013. doi:10.1007/s10514-012-9321-0. URL http://octomap.github.com.

39. E.G. Josberger und N.M. Mognard. A passive microwave snow depth algorithm with a proxy for snow metamorphism. In *Hydrological Processes*, 16(8), S. 1557–1568, 2002. URL http://onlinelibrary.wiley.com/doi/10.1002/hyp.1020/abstract.

40. T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman und A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7), S. 881–892, 2002.

41. M. Kazhdan und H. Hoppe. Screened poisson surface reconstruction. In *ACM Transactions on Graphics (TOG)*, 32(3), S. 29, 2013. URL `http://dl.acm.org/citation.cfm?id=2487237`.

42. K. Khoshelham und S.O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. In *Sensors*, 12(2), S. 1437–1454, 2012.

43. S. Kielhöfer. *Fehleranalyse und Modellierung eines 3D-Laserscansystems*. Masterarbeit, Technische Universität München, 2003.

44. S. Kim und J. Kim. Continuous occupancy maps using overlapping local gaussian processes. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3–7, 2013*, S. 4709–4714. IEEE, Tokyo, Japan, November 2013. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6697034`.

45. T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, Band 78, S. 1464–1480. IEEE, September 1990.

46. T. Kohonen, M.R. Schroeder und T.S. Huang (Hrsg.). *Self-Organizing Maps*. Springer-Verlag New York, Inc., 2001.

47. K. Konolige. Improved occupancy grids for map building. In *Autonomous Robots*, 4(4), S. 351–367, 1997.

48. K. Konolige, M. Agrawal, R.C. Bolles, C. Cowan, M. Fischler und B. Gerkey. Outdoor mapping and navigation using stereo vision. In *Experimental Robotics*, S. 179–190. Springer, 2008.

49. T. Krajnik, J. Pulido Fentanes, O. Martinez Mozos, T. Duckett, J. Ekekrantz, M. Hanheide et al.. Long-term topological localisation for service robots in dynamic environments using spectral maps. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14–18, 2014*, S. 4537–4542. IEEE, Chicago, Illinois, USA, September 2014. URL `http://eprints.lincoln.ac.uk/14423`.

50. T. Kucner, J. Saarinen, M. Magnusson und A.J. Lilienthal. Conditional transition maps: Learning motion patterns in dynamic environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3–7, 2013*, S. 1196–1201. IEEE, Tokyo, Japan, November 2013. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6696502`.

51. L. Li. *Time-of-Flight Camera – An Introduction*. Technischer Bericht, Texas Instruments, 2014.

52. W.E. Lorensen und H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, S. 163–169. ACM, New York, NY, USA, 1987. ISBN 0-89791-227-6. doi:10.1145/37401.37422. URL `http://doi.acm.org/10.1145/37401.37422`.

53. M. Luber, G.D. Tipaldi und K.O. Arras. Place-dependent people tracking. In *The International Journal of Robotics Research*, 30, S. 280–293, March 2011. URL `http://ijr.sagepub.com/content/early/2011/01/08/0278364910393538.abstract`.

54. M. Magnusson, A. Lilienthal und T. Duckett. Scan registration for autonomous mining vehicles using 3D-NDT. In *Journal of Field Robotics*, 24(10), S. 803–827, 2007. URL `http://onlinelibrary.wiley.com/doi/10.1002/rob.20204/abstract`.

55. T. Markus und D.J. Cavalieri. *Snow depth distribution over sea ice in the Southern Ocean from satellite passive microwave data*. Wiley Online Library, 1998.

56. H.P. Matthias Zwicker, Markus H. Gross. *A survey and classification of real time rendering methods*. Technischer Bericht 332, Swiss Federal Institute of Technology Zurich, Dezember 1999.

57. L. Matthies und A. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation, Philadelphia, Pennsylvania, USA*, S. 727–733. IEEE, 1988. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=12145`.

58. O.C. Miles Hansard, Seungkyu Lee und R. Horaud. *Time of Flight Cameras: Principles, Methods, and Applications*. Springer Briefs in Computer Science, 2012.

59. A. Milstein. *Occupancy grid maps for localization and mapping*, Kapitel 19, S. 381–408. InTech, 2008.

60. H. Moravec und A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, Missouri, USA*, Band 2, S. 116–121. 1985. doi:10.1109/ROBOT.1985.1087316.

61. D. Murray und C. Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation, 1997*, Band 2, S. 1694–1699. IEEE, 1997.

62. R.A. Newcombe, A.J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim und A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, S. 127–136. IEEE, 2011. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6162880`.

63. A. Nüchter, K. Lingemann, J. Hertzberg und H. Surmann. 6D SLAM with approximate data association. In *12th International Conference on Advanced Robotics, 2005. ICAR'05. Proceedings*, S. 242–249. IEEE, 2005.

64. S.T. O'Callaghan und F.T. Ramos. Continuous occupancy mapping with integral kernels. In *Proceedings of Twenty-Fifth AAAI Conference on Artificial Intelligence*. 2011.

65. H. Pfister, M. Zwicker, J. Van Baar und M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, S. 335–342. ACM Press/Addison-Wesley Publishing Co., 2000. URL `http://dl.acm.org/citation.cfm?id=344936`.

66. J.C. Platt. Probabilities for SV Machines. In *Advances in Large Margin Classifiers*, S. 61–74. MIT Press, 1999.

67. F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas und R. Siegwart. Noise characterization of depth sensors for surface inspections. In *2nd International Conference on Applied Robotics for the Power Industry (CARPI), 2012*, S. 16–21. IEEE, 2012.

68. R.B. Rusu, N. Blodow und M. Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12–17, 2009*. IEEE, Kobe, Japan, Mai 2009. URL `http://files.rbrusu.com/publications/Rusu09ICRA.pdf`.

69. J. Saarinen, H. Andreasson und A.J. Lilienthal. Independent markov chain occupancy grid maps for representation of dynamic environment. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7–12, 2012*, S. 3489–3495. IEEE, Vilamoura, Algarve, Portugal, Oktober 2012. ISBN 978-1-4673-1737-5. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6385629`.

70. J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala und A.J. Lilienthal. Normal distributions transform occupancy maps: Application to large-scale online 3D mapping. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6–10, 2013*, S. 2233–2238. IEEE, Karlsruhe, Germany, Mai 2013. ISBN 978-1-4673-5641-1. URL `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6615630`.

71. J. Saarinen, T. Stoyanov, H. Andreasson und A.J. Lilienthal. Fast 3D mapping in highly dynamic environments using normal distributions transform occupancy maps. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3–7, 2013*, S. 4694–4701. IEEE, Tokyo, Japan, November 2013. URL `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6679723`.

72. H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processin, and GIS*. Addison-Wesley Publishing Company, Inc., 1989.

73. A. Segal, D. Haehnel und S. Thrun. Generalized-ICP. In *Robotics: Science and Systems*, Band 2. 2009.

74. F.T.R. Simon T O'Callaghan. Gaussian process occupancy maps. In *The International Journal of Robotics Research*, 31(1), S. 42–62, January 2012. doi: 10.1177/0278364911421039.

75. R.C. Smith und P. Cheeseman. On the representation and estimation of spatial uncertainty. In *The international journal of Robotics Research*, 5(4), S. 56–68, 1986. URL `http://ijr.sagepub.com/content/5/4/56.short`.

76. F. Steinbrücker, J. Sturm und D. Cremers. Volumetric 3D mapping in real-time on a CPU. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 – June 7, 2014*. IEEE, Hong Kong, China, Mai 2014. URL `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6895053`.

77. T. Stoyanov, J. Saarinen, H. Andreasson und A.J. Lilienthal. Normal distributions transform occupancy map fusion: Simultaneous mapping and tracking in large scale dynamic environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3–7, 2013*, S. 4702–4708. IEEE, Tokyo, Japan, November 2013. URL `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6679723`.

78. M. Strauch und A. Zell. Topologische Umweltmodellierung und Pfadplanung bei mobilen Robotern. In *Fakultät für Informatik Prof. Dr. Andreas Zell Uni Tübingen*, 5, 2002.

79. J. Stückler und S. Behnke. Multi-resolution Surfel Maps for Efficient Dense 3D Modeling and Tracking. In *Journal of Visual Communication and Image Representation*, 25(1), S. 137–147, Januar 2014. ISSN 1047-3203. doi:10.1016/j.jvcir.2013.02.008. URL `http://dx.doi.org/10.1016/j.jvcir.2013.02.008`.

80. M. Suppa. *Autonomous Robot Work Cell Exploration Using Multisensory Eye-in-hand Systems*. Dissertation, Universität Hannover, 2008.

81. R. Szeliski. Image alignment and stitching: A tutorial. In *Foundations and Trends in Computer Graphics and Vision*, 2(1), S. 1–104, 2006. URL `http://dl.acm.org/citation.cfm?id=1295185`.

82. J.D. Tardós, J. Neira, P.M. Newman und J.J. Leonard. Robust mapping and localization in indoor environments using sonar data. In *The International Journal of Robotics Research*, 21(4), S. 311–330, 2002.

83. S. Thrun, W. Burgard und D. Fox. *Probabilistic Robotics*. MIT Press, MA, 2005. ISBN 0262201623.

84. B. Tian, V.A. Shim, M. Yuan, C. Srinivasan, H. Tang und H. Li. RGB-D based cognitive map building and navigation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3–7, 2013*, S. 1562–1567. IEEE, Tokyo, Japan, November 2013. URL `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6679723`.

85. A.P. Tirumalai, B.G. Schunck und R.C. Jain. Evidential reasoning for building environment maps. In *IEEE Transactions on Systems, Man, and Cybernetics*, Band 25, S. 10–20. IEEE, 1995. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=362968`.

86. C. Tomasi und R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, S. 839–846. IEEE, 1998. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=710815`.

87. T. Tykkala, A.I. Comport und J.K. Kamarainen. Photorealistic 3d mapping of indoors by rgb-d scanning process. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3–7, 2013*, S. 1050–1055. IEEE, Tokyo, Japan, November 2013. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6696480`.

88. T. Tykkälä, A.I. Comport, J.K. Kämäräinen und H. Hartikainen. Live rgb-d camera tracking for television production studios. In *Journal of Visual Communication and Image Representation*, 25(1), S. 207–217, 2014. URL `http://www.sciencedirect.com/science/article/pii/S1047320313000291`.

89. R.J. Urick. *Principles of Underwater Sound 3rd Edition*. 3. Auflage. Peninsula Pub, August 1996.

90. V. Varveropoulos. Robot localization and map construction using sonar data. In *The Rossum Project*, S. 10, 2005.

91. N.N. Victor Castaneda. Time-of-flight and kinect imaging. Kinect Programming for Computer Vision, June 2011.

92. C.C. Wang und C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11–15, 2002, Washington, DC, USA*, Band 3, S. 2918–2924. IEEE, Washington, DC, USA, Mai 2002. ISBN 0-7803-7273-5. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1013675`.

93. Z. Wang, R. Ambrus, Patric, Jensfelt und J. Folkesson. Modeling motion patterns of dynamic objects by iohmm. In *2014 IEEE/RSJ International Conference on Inelligent Robots and Systems*. 2014.

94. D.F. Wolf und G.S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. In *Autonomous Robots*, 19(1), S. 53–65, 2005. URL `http://link.springer.com/article/10.1007/s10514-005-0606-4`.

95. S. Zelinka und M. Garland. Permission grids: Practical, error-bounded simplification. In *ACM Transactions on Graphics (TOG)*, 21(2), S. 207–229, 2002. URL `http://dl.acm.org/citation.cfm?id=508363`.