Dominik Joho

Exploration für mobile Roboter unter Verwendung dreidimensionaler Umgebungsmodelle

Albert-Ludwigs-Universität Freiburg
Fakultät für Angewandte Wissenschaften
Lehrstuhl Autonome Intelligente Systeme
Prof. Dr. Wolfram Burgard

Diplomarbeit

Exploration für mobile Roboter unter Verwendung dreidimensionaler Umgebungsmodelle

Dominik Joho

März 2007 – September 2007

Erstgutachter:	Prof. Dr. Wolfram Burgard
Zweitgutachter:	Dr. Sven Behnke
Abgabedatum:	3. September 2007

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen / Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Dominik Joho
Freiburg im Breisgau, den 3. September 2007

Danksagung

An erster Stelle möchte ich mich bei Prof. Dr. Wolfram Burgard für die Möglichkeit bedanken, die Diplomarbeit an seinem Lehrstuhl durchführen zu können. Ebenso gilt mein Dank Dr. Sven Behnke für die bereitwillige Übernahme der Zweitkorrektur. Bei Dr. Cyrill Stachniss möchte ich mich für die Betreuung der Diplomarbeit bedanken. Seine Türe stand mir jederzeit offen und die Diplomarbeit hat sehr von seiner Erfahrung profitiert. Besonderer Dank geht auch an Patrick Pfaff und Rainer Kümmerle, die mir mit unendlicher Geduld bei Fragen zum Thema Scan-Matching, Lokalisierung und Morpheus weiterhelfen konnten. Zeno Gantner und Thorsten Zitterell gilt mein Dank für die Durchsicht eines Entwurfs der Diplomarbeit.

Schließlich möchte ich mich bei meinen Eltern bedanken, die mich während des Studiums nicht nur finanziell unterstützt haben. Ohne das Vertrauen, das mir meine Eltern und meine Schwester entgegenbringen, wäre diese Diplomarbeit und vieles andere nicht möglich gewesen. Nicht zuletzt gilt mein Dank meiner Freundin Karla Alcázar. Sie musste die letzten Monate auf viel gemeinsame Zeit verzichten. Ich kann ihr nicht genug danken – für ihr Verständnis, die Ermunterungen, das Lachen, die Ruhe.

Zusammenfassung

In dieser Diplomarbeit wird ein Verfahren vorgestellt, das einem mobilen Roboter die autonome Exploration einer anfangs unbekannten Umgebung ermöglicht. Dabei berücksichtigen wir explizit, dass der Roboter in einer *dreidimensionalen* Umgebung agiert. Als Umgebungsrepräsentation verwenden wir MLS-Karten (*multi-level surface maps*), die eine kompakte, dreidimensionale Modellierung der Umgebung ermöglichen. Die Auswahl des nächsten Zielpunkts an dem eine Messung vorgenommen werden soll, wird die Fahrtkosten und den erwarteten Informationsgewinn berücksichtigen. Die Fahrtkosten werden durch die zu fahrenden Distanz, sowie durch die Neigung und die Rauheit des traversierten Geländes beeinflusst. Die dazu vorgestellte Traversierbarkeitsanalyse ist im Gegensatz zu 2D-Ansätzen auch in der Lage, negative Hindernisse zu erkennen. Der erwartete Informationsgewinn wird durch eine simulierte Messung den Rückgang der Unsicherheit in der bereits bekannten Karte, sowie die zu erwartende Erweiterung der Karte miteinbeziehen. Wir evaluieren das vorgestellte Verfahren in Simulationsumgebungen und mit einem echten Roboter.

Inhaltsverzeichnis

Kapitel 1

Einführung

> We define *exploration* to be the act of moving through an unknown environment while building a map that can be used for subsequent navigation.
>
> Brian Yamauchi

Seit jeher stellen Menschen technische Artefakte zur Erleichterung ihrer alltäglichen Arbeit her. Waren dies lange Zeit einfache Gegenstände, wie etwa Werkzeuge für die Feldarbeit, vollzog sich spätestens im Zuge der Industrialisierung ein entscheidender Wandel. Die einstmals passiven technischen Artefakte konnten nun vom Menschen unabhängig einfache, mechanische Arbeitsgänge durchführen. Im Zuge der weiteren Entwicklung wurde die Automatisierung immer komplexerer mechanischer Arbeit möglich. Anfang des 20. Jahrhunderts stand ein erneuter Umbruch bevor. Mit den von Konrad Zuse entwickelten Rechenmaschinen beginnt das Zeitalter der Computer. Die Möglichkeiten der Automatisierung waren seitdem nicht mehr nur auf physische Tätigkeiten begrenzt, sondern umfassten erstmals auch kognitive Tätigkeiten. Schnell wurden die Möglichkeiten dieser Maschinen jenseits des Einsatzes von numerischen Berechnungen erkannt. Der Dartmouth-Workshop im Sommer 1956 gilt heute als die Geburtsstunde der Künstlichen Intelligenz (KI). Computer konnten erstmals mathematische Beweise durchführen. Mit dem *General Problem Solver* von Newell und Simon [32] wurde versucht, menschliches Problemlöseverhalten zu simulieren. Der enthusiastischen Anfangsphase folgte bald eine gewisse Ernüchterung, nachdem nicht alle der wohl zu hoch gesteckten Erwartungen erfüllt werden konnten. Dennoch wurden über die Jahrzehnte wichtige Fortschritte erzielt und die KI ist heute ein fest etabliertes Teilgebiet der Informatik.

Gerade im Hinblick auf die grob skizzierte Geschichte technischer Artefakte nimmt

die sich heutzutage rasch entwickelnde Robotik, die als Teilgebiet der Künstlichen Intelligenz gesehen werden kann, eine vielleicht geradezu historische Stellung ein, da sie versucht, mechanische und kognitive Automatisierung zu kombinieren und damit eine neue Klasse technischer Artefakte entwickelt. Ihr Ziel ist die Konstruktion von physisch realisierten, autonomen, intelligenten Systemen, die flexibel in einer dynamischen und unvorhersehbaren Umgebung agieren.

Die Problemstellungen in der Robotik unterscheiden sich zum Teil sehr von jenen der Anfangsphase der KI. Die noch junge KI beschäftigte sich beispielsweise mit dem Schachspiel oder dem Beweisen von mathematischen Theoremen. Aufgaben also, bei denen man davon ausging, dass sie beim Menschen eine gewisse Form von „Intelligenz“ erfordern, um gelöst zu werden. Die Robotik sieht sich anderen Problemstellungen gegenüber. Für autonom agierende Systeme spielt die Sensorik und Bewegungskontrolle eine entscheidende Rolle. Ein modernes Leitproblem für die Robotik, und damit auch der KI, ist heute nicht mehr das Schachspiel, sondern das Fußballspiel. Der RoboCup [2] ist ein jährlich ausgetragener, internationaler Wettbewerb, in dem in verschiedenen Ligen Roboter-Fußballteams (unter anderem) gegeneinander antreten. Das Ziel ist, bis zum Jahr 2050 ein Roboterteam zu entwickeln, das gegen den amtierenden menschlichen Fußballweltmeister gewinnt. Die Herausforderung hat bisher viele in der Robotik tätige Forschergruppen angezogen. Im RoboCup 2007 in Atlanta nahmen über 300 Teams aus 39 Ländern teil. Der Wandel der Leitprobleme – Fußball statt Schach – zeigt auch, dass technischen Systemen oft Fertigkeiten Probleme bereiten, die für Menschen selbstverständlich und scheinbar mühelos einsetzbar sind. Die Robotik lädt damit auch immer wieder zum Staunen über die Fähigkeiten biologischer Systeme ein.

Eine wichtige Grundlage autonomer Systeme ist die Wahrnehmung der Umgebung in der sie agieren. Der Aufbau von Umgebungsmodellen, oder Karten, und die Lokalisierung, also Positionsbestimmung innerhalb einer solchen Karte, ist daher auch ein seit vielen Jahren untersuchtes Problem in der Robotik. Besonders schwierig ist das gleichzeitige Kartieren und Lokalisieren in der gerade im Aufbau befindlichen Karte (*simultaneous localization and mapping*, SLAM) [42], da dies einem Henne-Ei-Problem gleicht: Um sich zu lokalisieren, muss eine Karte vorhanden sein in der man sich lokalisiert – andererseits muss die eigene Position bekannt sein, will man neue Informationen in eine Karte eintragen.

Lösungen für das SLAM-Problem sind an sich passiv, in dem Sinne, dass sie nur eingehende Informationen über die Umgebung sowie der Bewegung des Roboters verarbeiten, um daraus eine konsistente Karte und Schätzung der Trajektorie des Roboters zu erstellen. Es werden jedoch keine Aktionen generiert, der Roboter handelt nicht selbsttätig, sondern wird von einem menschlichen Operator durch die Umge-

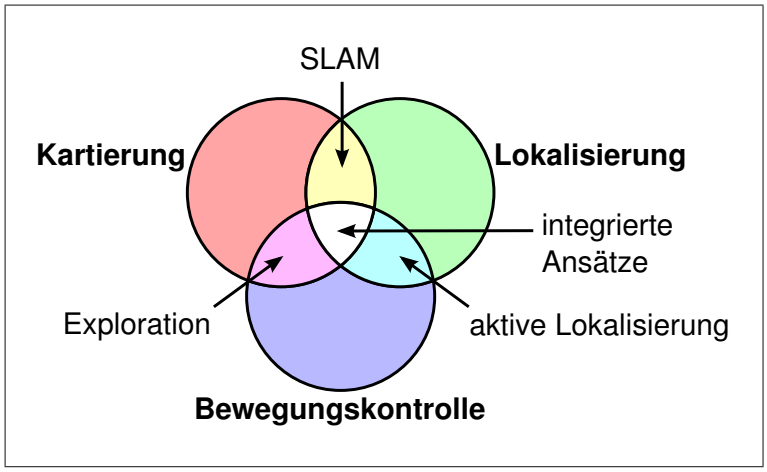


Abbildung 1.1: Zusammenhang wichtiger Teilprobleme der mobilen Robotik (nach [29, 41]).

bung gesteuert, um die Informationen aufzuzeichnen. Bei der autonomen Exploration übernimmt der Roboter zusätzlich die Rolle des Operators und steuert sich selbst, um eine anfangs unbekannte Umgebung zu erkunden. Nachdem ein Teil der Umgebung kartiert ist, muss entschieden werden, welche Stelle als nächstes angefahren werden soll, um eine neue Messung vorzunehmen und die Karte zu erweitern. Dieses Explorations-Problem ist Thema der vorliegenden Arbeit.

Integrierte Ansätze [29, 40, 41] beachten bei der Auswahl von Explorationsaktionen auch die Auswirkungen auf die Lokalisierung. Thrun *et al.* [42] identifizieren noch weitere Varianten des Explorations-Problems. Anstatt eine statische Umgebung anzunehmen, die es zu kartieren gilt, könnten sich während der Exploration auch Personen in der Umgebung bewegen. Aufgabe ist dann, während der Kartierung zu entscheiden, was ein Teil der Karte ist und was ein dynamisches Objekt, das es auszufiltern gilt. Zudem stellen dynamische Objekte neue Anforderungen an eine autonome Navigation. Anstatt nur die Umgebung zu kartieren, könnte dann die Aufgabe sein, eine Person in einer unbekannten Umgebung zu suchen und zu verfolgen (*pursuit evasion problem*) [18]. Da sich die Person bewegt, müssen im Gegensatz zur reinen Kartierung Teile der Umgebung mehrfach besucht werden. Eine weitere Variante entsteht, wenn man nicht die Information über die Umgebung maximieren möchte, sondern die Information über die eigene Position (*active localization*) [23]. Auch die Erkundung eines unbekannten Objekts durch einen Roboter, der mit einem Manipulator ausgestattet wurde, kann als Explorations-Problem angesehen werden. Anhand dieser Varianten heben Thrun *et al.* [42] hervor, dass Explorations-Probleme praktisch in allen Bereichen der Robotik auftreten.

Die Mehrzahl der verwendeten Umgebungsrepräsentationen ist zweidimensional.

Das heißt man geht davon aus, dass der Roboter in einer zweidimensionalen Welt agiert. Eine solche Annahme hat den Vorteil, dass sie im Gegensatz zu 3D-Ansätzen geringere Anforderungen an die Speicher- und Rechenkapazitäten stellt. Solange Roboter sich in einfach strukturierten Innenumgebungen mit nur positiven Hindernissen wie Wänden und Stühlen bewegen, ist dies eine praktikable Vereinfachung der tatsächlichen Umgebungsstruktur. Sobald die Innenumgebung komplizierter strukturiert ist oder der Roboter gar im Freien eingesetzt werden soll, ist eine dreidimensionale Repräsentation der Umgebung von Vorteil oder gar unumgänglich. Zudem bedeutet die Reduktion einer dreidimensionalen Welt auf eine zweidimensionale Karte immer auch einen Informationsverlust, der je nach Anwendung unerwünscht sein kann. Die meisten veröffentlichten Explorationsansätze gehen von einer zweidimensionalen Umgebung aus. Teil der vorliegenden Arbeit wird es daher auch sein, existierende Explorationstechniken auf die Anforderungen einer dreidimensionalen Umgebungen hin anzupassen.

1.1 Aufgabenstellung

Die im Rahmen dieser Diplomarbeit zu bearbeitende Aufgabenstellung war die Entwicklung und Implementierung eines Verfahrens, das einem mobilen Roboter die autonome Exploration einer dreidimensionaler Umgebung ermöglicht. Dabei soll auf die Besonderheiten eingegangen werden, die eine dreidimensionale Umgebung mit sich bringt, etwa negative Hindernisse oder unebenes Gelände. Zudem sollte das Verfahren den Informationsgewinn einer Messung abschätzen können.

1.2 Beitrag der Arbeit

Der Beitrag dieser Arbeit ist die Entwicklung eines Verfahrens, das einem mobilen Roboter die autonome Exploration einer anfangs unbekannten, dreidimensionalen Umgebung ermöglicht. Als Umgebungsrepräsentation werden so genannte MLS-Karten (*multi-level surface maps*) verwendet [45], wodurch eine kompakte, dreidimensionale Modellierung der Umgebung erreicht wird. Die Auswahl von anzufahrenden Zielpunkten wird die Fahrtkosten und den erwarteten Informationsgewinn miteinbeziehen. Dabei wird speziell auf die Probleme eingegangen die entstehen, wenn der Roboter in einer dreidimensionalen Umgebung agiert. Es wird eine neuen Traversierbarkeitsanalyse für MLS-Karten vorgestellt, welche unter anderem die Neigung und Rauheit des Geländes berücksichtigt. Zudem wird eine 3D-Raycasting-Methode für MLS-Karten vorgestellt, die bei der Generierung von möglichen Zielpunkten und Zwischenzielen eingesetzt wird sowie bei der Berechnung des erwarteten Informati-

onsgewinns. Das Verfahren wurde in verschiedenen Simulationsexperimenten sowie mit einem echten Roboter getestet und evaluiert.

Die Arbeit baut auf dem Softwaresystem Morpheus auf, das bereits einen Teil der benötigten Funktionalitäten, wie z. B. Kartierung und Lokalisierung, bereitstellt. Zu implementieren sind Module, die dem Roboter das autonome Navigieren sowie Entscheidungen über anzufahrende Zielpunkte ermöglichen.

Die Kernidee dieser Diplomarbeit wird in

> Dominik Joho, Cyrill Stachniss, Patrick Pfaff und Wolfram Burgard: *Autonomous Exploration for 3D Map Learning*. In: *Fachgespräche Autonome Intelligente Systeme (AMS)*, Kaiserslautern, 2007.

vorgestellt. Im Vergleich zu dieser Publikation präsentieren wir in der vorliegenden Arbeit umfangreichere Experimente. Zudem wurden Details wie das Verfahren zur Berechnung des Informationsgewinns verbessert.

1.3 Aufbau der Arbeit

In Kapitel 2 geben wir einen Überblick relevanter Literatur zum Thema Exploration mit mobilen Robotern. Kapitel 3 wird die verwendete Umgebungsrepräsentation (MLS-Karten) beschrieben. Kapitel 4 stellt den Beitrag dieser Diplomarbeit vor. Es wird auf die Explorationsstrategie eingegangen, sowie einige notwendige Definitionen und Techniken vorgestellt. Im folgenden Kapitel 5 gehen wir kurz auf Implementations-Details ein. Die Ergebnisse durchgeführter Experimente werden in Kapitel 6 besprochen. Schließlich geben wir in Kapitel 7 eine Zusammenfassung und einen Ausblick auf mögliche Erweiterungen des Verfahrens.

Kapitel 2

Verwandte Arbeiten

In den vergangenen zwei Jahrzehnten wurden diverse Ansätze zur autonomen Exploration mit mobilen Robotern vorgestellt. Die meisten Ansätze basieren auf zweidimensionalen Karten [4, 15, 19, 20, 30, 38, 39, 40, 46, 48], seltener werden dreidimensionale Karten eingesetzt [14, 33, 44, 47]. Integrierte Ansätze [29, 40, 41] beziehen auch die Unsicherheit der Lokalisierung in die Auswahl von Explorationsaktionen mit ein. Neben der Exploration mit nur einem einzigen Roboter existieren Verfahren, die gleichzeitig mehrere Roboter einsetzen und koordinieren [9, 10, 21, 22, 49].

Yamauchi [48, 49] führt den populären grenzzellenbasierten Ansatz (*frontier-based approach*) für Gitterkarten ein. Die Zellen des Gitters werden in die Klassen „frei“, „belegt“ und „unbekannt“ eingeteilt. Die Zugehörigkeit zur jeweiligen Klasse richtet sich nach der Wahrscheinlichkeit, dass die Zelle belegt ist. Ist diese Wahrscheinlichkeit gleich der a-priori-Wahrscheinlichkeit, gilt sie als unbekannte oder nicht-explorierte Zelle. Als Grenzzellen werden jene freien Zellen definiert, die eine benachbarte, unbekannte Zelle besitzen. Zusammenhängende Grenzzellen werden zu Grenzregionen zusammengefasst und Grenzregionen, die eine gewisse Größe überschreiten, werden als Grenzen zum unbekannten Gebiet interpretiert. Viele der nachfolgend publizierten Explorationsansätze basieren, in der ein oder anderen Form, auf der Bestimmung von Grenzzellen oder Grenzlinien. Auch die in dieser Diplomarbeit vorgestellte Explorationsstrategie folgt dem grenzzellenbasierten Ansatz, jedoch wird eine Anpassung der Definition von Grenzzellen auf die dreidimensionale Struktur der MLS-Karten notwendig sein.

Moorehead *et al.* [31] präsentieren ein Verfahren, das bei der Auswahl von Zielpunkten mehrere Informationsquellen berücksichtigt. Die Umgebung wird in ein diskretes Gitter eingeteilt und mit jeder Zelle ist ein Informationsvektor $G \in [0,1]^n$ assoziiert. Jeder Eintrag entspricht dem Beitrag, den eine der n Informationsquellen liefert und der durch eine simulierte Messung bestimmt wurde. Der endgültige erwar-

tete Informationsgewinn ist die gewichtete Summe aller Einträge von G. Ein Pfad besteht aus einer Sequenz zu traversierender Zellen sowie aus Positionen, an denen eine Messung stattfinden soll. Der Nutzen eines Pfades ist eine gewichtete Differenz zwischen den erwarteten Informationsgewinnen an den Messpunkten und der Zeit, die für die gefahrene Strecke, die Messungen und die Planung aufgewandt werden muss. Die Autoren heben hervor, dass die Maximierung über alle möglichen Pfade dem PCTS-Problem gleicht (*prize-collecting salesmen problem*), für das gezeigt wurde, dass es NP-vollständig ist [5]. Daher werden zwei suboptimale Planungsstrategien vorgestellt. Die erste Strategie gleicht einer Zufallsbewegung und führt eine Messung mit einer Wahrscheinlichkeit proportional zum erwarteten Informationsgewinn der aktuellen Zelle aus, oder geht in eine benachbarte Zelle. Die zweite Strategie plant nur zum ersten Messpunkt und berechnet den Nutzen für jeden erreichbaren Punkt. In einer Implementierung der Strategie wird zur Bestimmung des Informationsgewinns unter anderem die Entropie der Zellen benutzt. Im Gegensatz zu dem in dieser Diplomarbeit vorgestellten Verfahren ist keine Exploration übereinanderliegender Ebenen möglich.

González-Baños und Latombe [19] verwenden eine polygonbasierte Umgebungsrepräsentation. Grenzen zwischen freien und noch nicht explorierten Gebieten werden durch spezielle, „freie“ Kanten modelliert. Potenzielle Zielpunkte für eine Messung werden zufällig im Sichtbarkeitsbereich freier Kanten erzeugt, wodurch sichergestellt ist, dass von jedem potenziellen Zielpunkt der noch nicht explorierte Bereich einsehbar ist. In einem zweiten Schritt werden Zielpunkte entfernt, zu denen kein kollisionsfreier Pfad existiert, oder die Gesamtlänge der vom Zielpunkt aus sichtbaren Segmente der „freien“ Kanten unterhalb eines Grenzwertes liegt. Jeder verbleibende Zielpunkt wird dann durch eine Evaluierungsfunktion bewertet, die den erwarteten Informationsgewinn und die Fahrtkosten berücksichtigt. Der erwartete Informationsgewinn ist die erwartete, vom Zielpunkt aus einsehbare Fläche des nicht explorierten Bereichs. Die Fahrtkosten entsprechen der Länge des kürzesten Pfades. Über eine Konstante können beide Faktoren gegeneinander abgewogen werden. Die Exploration terminiert, sobald es keine freien Kanten mehr gibt, deren Länge überhalb eines gewissen Grenzwertes liegen. Im Gegensatz dazu verwenden wir eine gitterbasierte 3D-Karte und einen probabilistischen Ansatz zur Berechnung des erwarteten Informationsgewinns. Zudem verwenden wir eine 3D-Raycasting-Methode um potenzielle Zielpunkte im Sichtbarkeitsbereich von Grenzlinien zu bestimmen.

Visser *et al.* [46] verbessern in ihrem Ansatz die Methode von González-Baños und Latombe [19] zur Abschätzung der einsehbaren Fläche jenseits einer Grenzlinie zum nicht explorierten Gebiet. Aus den Laserdaten werden zwei unterschiedliche Belegtheitskarten, A und B, aufgebaut. Für Karte A werden Laserstrahlen auf ei-

ne maximale Länge von 2 m begrenzt, was zu einer Karte führt, die aus „sicheren“ Bereichen besteht. Karte B begrenzt Laserstrahlen auf 20 m und führt zu einer Karte „eingesehener“ Bereiche. Grenzlinien sind diejenigen Teilbereiche der Kontur von Karte A, die keine Hindernisse darstellen. Pro Grenzlinie wird ein Zielpunkt generiert. Aus Karte B wird ebenfalls eine Kontur extrahiert. Der Informationsgewinn eines Zielpunkts entspricht der Fläche des Teilpolygons von Karte B, das hinter der zugehörigen Grenzlinie liegt. Zwar wurden diese Bereiche bereits von Laserstrahlen erreicht, werden jedoch noch nicht als ausreichend exploriert angesehen. Im Gegensatz dazu berechnen wir pro Grenzlinie mehrere Zielpunkte und machen den Informationsgewinn eines Zielpunkts nicht von bisherigen Messungen abhängig, sondern von simulierten Messungen am jeweiligen Zielpunkt.

Grabowski *et al.* [20] stellen eine Explorationsstrategie vor, die unter Berücksichtigung der Eigenschaften von Sonarsensoren versucht, die Genauigkeit der Karte zu erhöhen. Über ein inverses Sensormodell werden für jedes Hindernis Bereiche bestimmt, in denen durch eine Messung die Auflösung erhöht werden könnte. Die Bereiche jeder belegten Zelle werden überlagert und definieren Bereiche von hohem und geringem Interesse, die die Auswahl des nächsten Zielpunkts für die Exploration beeinflussen. Ähnlich zu diesem Ansatz sind wir bei der Auswahl von Zielpunkten darauf bedacht, die Genauigkeit der bereits bekannten Karte zu erhöhen. Wir erzeugen jedoch Zielpunkte in der Nähe zum unbekannten Gebiet und schätzen die Reduktion der Unsicherheit in der Karte durch simulierte Messungen an diesen Punkten ab. Zudem verwenden wir einen Laserscanner, der üblicherweise exaktere Abstandsinformationen liefert, als ein Sonarsensor.

Stachniss und Burgard [38, 39] verfolgen einen entscheidungstheoretischen Ansatz, in dem sie den erwarteten Informationsgewinn an einem Zielpunkt und die Fahrtkosten bis zu diesem Punkt gegeneinander abwägen. Um den erwarteten Informationsgewinn zu bestimmen, werden mögliche Messungen in der Karte simuliert und der Entropieunterschied bestimmt, der sich durch die temporäre Integration einer simulierten Messung ergeben würde. Der Nutzen eines Zielpunktes ergibt sich als gewichtete Summe des relativen, erwarteten Informationsgewinns und der relativen Fahrtkosten (jeweils relativ zum Maximalwert). Über eine unterschiedliche Gewichtung dieser zwei Faktoren kann die Explorationsstrategie variiert werden. Die Autoren gehen jedoch von einer zweidimensionalen Umgebung mit nur positiven Hindernissen aus. Das hier vorgestellte Verfahren geht auf die Besonderheiten einer 3D-Exploration ein und kann auch negative Hindernisse berücksichtigen.

Tovey und Koenig [43] untersuchen in einer theoretischen Arbeit *greedy mapping*, das den Roboter immer zur nächstgelegenen interessanten Stelle führt. Die Umgebung wird als Graph repräsentiert. Sie geben eine neue Obergrenze von $O(|V| \log |V|)$

Kantenzügen für die Laufzeit im schlechtesten Fall an, wobei $|V|$ die Anzahl der Knoten im Graph ist. Die Obergrenze gilt für alle vier untersuchten Varianten, die sich darin unterscheiden, wie interessante Punkte definiert werden: als nicht besuchte Knoten, nicht beobachtete Knoten, nicht beobachtete Knoten mit Neuplanung, oder informative Knoten.

Stachniss *et al.* [40] präsentieren einen integrierten Ansatz für die Kartierung, Lokalisierung und Exploration in einer zweidimensionalen Umgebung. Um das SLAM-Problem zu lösen, wird ein *Rao-Blackwellized particle filter* verwendet. Ziel der Exploration ist es, die Entropie über die Trajektorie des Roboters sowie die Entropie der mit den Partikeln assoziierten Belegtheitskarten zu reduzieren. Dazu wird zufällig ein Partikel entsprechend seiner Wahrscheinlichkeit ausgewählt. In der zugehörigen Karte werden Messungen durch eine Raycasting-Methode simuliert und zusammen mit den Kontrollbefehlen für die simulierte Fahrt in den Filter integriert. Über den Entropieunterschied ergibt sich der Informationsgewinn. Die Pfadkosten sind proportional zu Summe aller Belegtheitswahrscheinlichkeiten der traversierten Zellen. Der Nutzen eines Zielpunkts ergibt sich aus einer Abwägung zwischen Informationsgewinn und Pfadkosten. Zielpunkte werden an der Grenze zu unexploriertem Gebiet sowie an bereits besuchten Orten erzeugt, um aktiv Schleifen schließen zu können. Für MLS-Karten ist die Verwendung eines *Rao-Blackwellized particle filters* jedoch durch den hohen Speicher- und Rechenaufwand nicht praktikabel.

Freda *et al.* [15] stellen eine randomisierte Explorationsstrategie vor. Während der Exploration wird eine *sensor-based random tree* (SRT) genannte Datenstruktur aufgebaut, welche die vom Roboter erreichten Positionen und die mit jeder Position assoziierte sichere Region (*local safe region*) enthält. Die sichere Region kann konservativ (SRT-Ball) oder optimistisch geschätzt werden (SRT-Stern). Von den sicheren Regionen wird angenommen, dass sie frei von Hindernissen sind. Die Kontur der jeweils letzten sicheren Region wird in Segmente dreier Klassen eingeteilt: Hindernis, freies Segment, oder Grenzsegment (zum unexplorierten Gebiet). Freie Segmente liegen innerhalb einer anderen sicheren Region, Hindernissegmente liegen an detektierten Hindernissen und Grenzsegmente sind die restlichen Segmente. Es wird dann zufällig ein Grenzsegment mit einer Wahrscheinlichkeit proportional zu seiner Länge ausgewählt und eine zufällige Zielposition in der Nähe dieses Segments gewählt. Unser Verfahren wählt dagegen Zielpunkte nicht aufgrund der Länge der Grenzlinie aus, sondern aufgrund des erwarteten Informationsgewinns und der Fahrtkosten.

Amigoni und Gallo [4] stellen eine Explorationsansatz vor, der auf multikriterieller Optimierung (*multi-objective optimization*) basiert und damit versucht, die Vermengung verschiedener Bewertungskriterien in einer Nutzenfunktion zu vermeiden. Die Umgebung wird als Liste von freien Segmenten und Hindernissegmenten repräsen-

tiert. Die Auswahl eines Zielpunktes berücksichtigt die Distanz (Länge des geplanten Pfades), den Informationsgewinn (Gesamtlänge aller sichtbaren freien Segmente), und die Überlappung mit der bereits bekannten Karte (Gesamtlänge aller sichtbaren Hindernissegmente). Kandidatenpunkte werden zufällig entlang freier Kanten erzeugt und Punkte, die nicht pareto-optimal[1] bezüglich dieser Menge sind, verworfen. Aus der Menge der pareto-optimalen Punkte wird derjenige Punkt ausgewählt, der die Distanz im dreidimensionalen Bewertungsraum zu einer Ideallösung minimiert. Durch den letzten Schritt werden die Bewertungskriterien jedoch effektiv doch durch eine Nutzenfunktion, eben der Distanzfunktion im Bewertungsraum, vermengt.

Mei *et al.* [30] stellen eine energieeffiziente Explorationstrategie vor. Zielpunkte sind Grenzzellen zum unbekannten Gebiet. Die Auswahl des nächsten Zielpunktes richtet sich nach der relativen Lage zum Roboter. Präferiert werden Ziele, die links vom Roboter liegen, danach jene die vorne, rechts oder hinten liegen. Zudem wird darauf geachtet, dass der Zielpunkt genügend Abstand zum nächsten Hindernis besitzt, um den Laserscanner möglichst effektiv einsetzen zu können. Diese Auswahlstrategie reduziert die Wahrscheinlichkeit, dass die gleiche Umgebung mehrfach gescannt wird. Um weiter Energie zu sparen, berücksichtigt die vorgestellte Pfadplanung die Orientierung des Roboters, um nicht nur die Länge des geplanten Pfades zum Ziel zu reduzieren, sondern auch die Anzahl der Drehungen, die der Roboter ausführen muss. Es handelt sich hierbei wiederum um einen reinen 2D-Ansatz.

Howard *et al.* [21, 22] schlagen eine Umgebungsmodellierung für die Kartierung mit mehreren Robotern vor, die auf einer Mannigfaltigkeit basiert. Der Vorteil dieser Repräsentation ist, dass sie auch in kritischen Situationen kurz vor dem Schließen von Schleifen stets selbstkonsistent ist.

Gerbaud *et al.* [17] verwenden als Karte ein dreidimensionales Polygonnetz. Für die autonome Exploration wurden drei verschiedene Modi implementiert: Im reaktiven Modus fährt der Roboter vorzugsweise geradeaus. Falls die Traversierbarkeitskarte dies nicht zulässt, wird in eine andere Richtung ausgewichen, wobei Richtungen präferiert werden, die der ursprünglichen Richtung möglichst nahe kommen. Im zielbasierten Modus wird der Roboter von einem gesetzten Ziel angezogen. Der Modus unterscheidet sich nur dadurch vom reaktiven Modus, dass Richtungen bevorzugt werden, die zielführend sind. Nur im dritten und letzten Modus findet eine Entscheidung über mögliche neue Ziele statt. Es werden Grenzen zum unbekannten Gebiet bestimmt und ihr Nutzen anhand deren Größe und Entfernung zum Roboter evaluiert. Der Mittelpunkt der besten Grenze wird dann als neuer Zielpunkt gesetzt und es findet ein Wechsel in den zielorientierten Modus statt. Im Gegensatz zu unserer

[1] Ein Punkt ist pareto-optimal bezüglich einer Menge von Punkten, falls es keinen anderen Punkt in dieser Menge gibt, der in *allen* Bewertungskriterien besser ist.

Arbeit schätzen die Autoren den Informationsgewinn implizit über die Länge der Grenzlinie ab und nicht über simulierte Messungen.

Triebel *et al.* [44] verwenden einen Laserscanner, der auf einem Roboterarm montiert wurde und für die 3D-Kartierung der näheren Umgebung des Arms eingesetzt wird. Die Umgebung wird als 3D-Gitter mit Belegtheitswahrscheinlichkeiten modelliert. Die Bewertung von Pfaden, die der Arm während der Exploration nehmen soll, berücksichtigt den erwarteten Informationsgewinn und die Pfadkosten, wobei letztere als die inverse Distanz zum nächsten Objekt definiert sind, um die Wahrscheinlichkeit einer Kollision zu verringern. Der Informationsgewinn wird durch den Entropieunterschied bestimmt, den die Integration einer simulierten Messung im 3D-Gitter hervorrufen würde. Die Generierung von Zielpunkte für die nächste Messung richtet sich nach der Bounding Box[2] der gemessenen Punktwolke und versucht Positionen zu finden, an denen eine Kante der Bounding Box sichtbar ist. Von allen generierten Zielpunkten, werden die h nächstgelegenen nach Informationsgewinn und Pfadkosten bewertet. Punkte mit zu geringer Bewertung werden verworfen. In dieser Diplomarbeit hingegen verwenden wir einen mobilen Roboter und sind somit nicht auf die unmittelbare Umgebung des Roboters eingeschränkt.

Whaite und Ferrie [47] nutzen ebenfalls einen beweglichen Laserscanner, zur Exploration eines eingeschränkten Bereichs nahe des Lasers. Sie verwenden jedoch ein parametrisches Modell der Umgebung und die Wahl des Punkts für die nächste Messung richtet sich nach der Unsicherheit im Modell. Im Gegensatz dazu agiert der Roboter in unserem Verfahren innerhalb der konvexen Hülle des aufzubauenden Umgebungsmodells.

Nüchter [33] stellt ein Verfahren zur Exploration einer dreidimensionalen Umgebungen vor. Es wird ein 2D-Laserscanner eingesetzt, der um die horizontale Achse gedreht werden kann und somit die Erstellung einer 3D-Punktwolke ermöglicht. Aus der 3D-Punktwolke werden in horizontalen Schnitten Teilwolken extrahiert und aus diesen jeweils eine polygonbasierte Karte (Risspolygon) erstellt, die detektierte und „freie“ Kanten enthält. Existieren mehrere 3D-Scans, werden die entsprechenden Riss-Polygone eines jeden Scans zusammengeführt. Innerhalb jedes Risspolygons werden zufällige Kandidatenpositionen erzeugt. Die Evaluierungsfunktion für eine Kandidatenposition berücksichtigt den Distanz- und Winkelunterschied zur aktuellen Roboterposition. Zudem werden vom Kandidatenpunkt Messungen durch 2D-Raycasting simuliert und der Informationsgewinn als die Anzahl der Schnittpunkte der vom Kandidatenpunkt ausgesandten Linien mit den „freien“ Kanten definiert. Wir hingegen generieren Zielpunkte gezielt in der Nähe von unbekannten Bereichen und nutzen 3D- statt 2D-Raycasting um den Anforderungen auch komplex strukturierter

[2] Minimaler, achsenparalleler Quader, der alle Punkte beinhaltet.

Umgebungen gerecht zu werden.

Fournier *et al.* [14] stellen einen 3D-Explorationsansatz vor. Die Umgebung wird durch einen Octree mit dynamischer Auflösung repräsentiert. Jedes Blatt des Octrees speichert die Belegtheitswahrscheinlichkeit des dadurch repräsentierten Raumes. Der Baum wird in eine $2\frac{1}{2}$D-Repräsentation umgewandelt, um die Belegtheit der Voxel[3] entlang der z-Achse zu bestimmen und diese in fünf Kategorien einzuteilen („sicher belegt“, „unbekannt-belegt“, „unbekannt“, „unbekannt-frei“, „sicher frei“). Der Zustand einer Zelle hängt von der Anzahl der Voxel einer jeden Kategorie ab, die sich in dieser Zelle befinden. Aufgrund dieser Zellinformation werden nach dem bekannten Ansatz von Yamauchi Grenzzellen bestimmt. In die Bewertung einer Grenzzelle geht der Informationsgewinn und die Kosten der Zelle mit ein. Der Informationsgewinn ist proportional zur Anzahl der unbekannten Voxel, die durch eine simulierte Messung erreicht werden. Die Kosten einer Grenzzelle berücksichtigt die Distanz zur aktuellen Roboterposition, den Abstand zum nächstgelegenen Hindernis, sowie den Unterschied zwischen der Orientierung des Roboters an der Grenzzelle im Vergleich zur aktuellen Orientierung. Trotz der Verwendung eines 3D-Modells sind in diesem Ansatz einige Definitionen an die Zellen eines 2D-Gitters gebunden. Im Gegensatz dazu werden wir dies vermeiden, da sonst eine Exploration von übereinander liegenden Ebenen nicht mehr gewährleistet werden könnte.

Unser Verfahren unterscheidet sich von den vorgestellten Ansätzen durch den Einsatz von MLS-Karten (*multi-level surface maps*), wodurch eine kompakte, dreidimensionale Umgebungsmodellierung erreicht wird. Im Gegensatz zu zweidimensionalen und vielen dreidimensionalen Ansätzen sind wir damit in der Lage, auch übereinanderliegende Ebenen zu explorieren. Durch die vorgestellte Traversierbarkeitsanalyse können wir unter anderem auch negative Hindernisse erkennen und vermeiden – ein Problem, das in zweidimensionalen Ansätzen üblicherweise ausgeklammert wird. Zudem wird bei den Fahrtkosten die Rauheit und die Steigung des Geländes berücksichtigt. Sowohl bei der Generierung von potenziellen Zielpunkten für die nächste Messung, als auch bei der Bewertung der potenziellen Zielpunkte tragen wir der dreidimensionalen Struktur durch eine 3D-Raycasting-Methode Rechnung. Die Bewertung eines potenziellen Zielpunkts wird durch den erwarteten Informationsgewinn und durch die Fahrtkosten beeinflusst.

[3] Hier: dreidimensionaler Würfel festgelegter Kantenlänge.

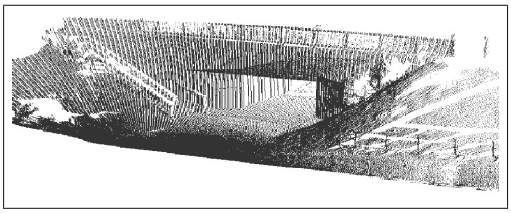
Kapitel 3

Multi-Level Surface Maps

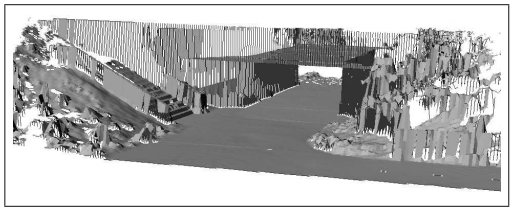
In der Robotik haben sich unterschiedlichste Formen zur Repräsentation der Umgebung etabliert, die üblicherweise nach topologischer und metrischer Repräsentation unterschieden werden [42]. Topologische Karten weisen eine graphenähnliche Struktur auf, wie z. B. Karten eines Straßenbahnnetzes. Die Knoten entsprechen markanten Orten oder bestimmten Merkmalen der Umgebung. Kanten modellieren die Erreichbarkeit oder Sichtbarkeit zwischen Knoten. Die räumliche Auflösung von topologischen Karten ist, verglichen mit metrischen Karten, eher schlecht, da sie höchstens die relative Lage markanter Orte zueinander modellieren, jedoch keine Informationen über den Raum zwischen den als Knoten modellierten Orten enthalten.

Bei einer metrischen Repräsentation wird die Umgebung häufig in ein Gitter mit einer festgelegter Zellgröße eingeteilt. Relevante Informationen über die Umgebung werden in den Zellen gespeichert. Die Zellgröße heute verwendeter Karten bewegen sich meist im Zentimeterbereich. Eine populäre metrische Repräsentationsform sind Belegtheitskarten (*occupancy grids*) [42], in der jede Zelle die Wahrscheinlichkeit speichert, dass der durch sie repräsentierte Raum durch ein Hindernis, etwa eine Wand, belegt ist.

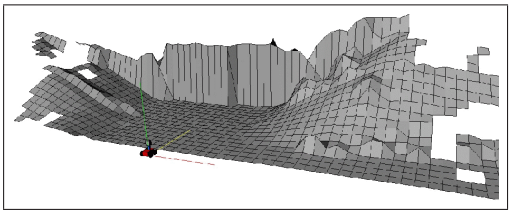
Dreidimensionale metrische Repräsentationen, wie etwa Polygonnetze oder 3D-Gitter, stellen hohe Anforderungen an die Rechen- und Speicherkapazitäten und erschweren somit die Kartierung von größeren Gebieten. Ein beliebter Ansatz diese Schwierigkeiten zu umgehen, ist die Verwendung von $2\frac{1}{2}$D-Repräsentationen, wie sie etwa Höhenkarten (*elevation maps*) darstellen [6, 28, 35, 37]. Dabei wird in den Zellen eines zweidimensionalen Gitters die Höheninformation des Geländes gespeichert. Mit dieser Repräsentation ist es jedoch nicht möglich Brücken oder den Innenraum eines mehrstöckigen Hauses zu kartieren, da Höhenkarten keine übereinanderliegenden Ebenen unterstützen. Um dies zu verdeutlichen ist in Abb. 3.1a die 3D-Punktwolke einer Messung vor einer Brücke dargestellt. Eine einfache Höhenkarte mittelt über



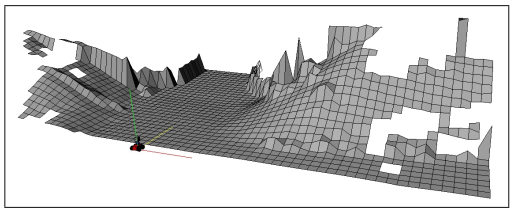
(a) 3D-Punktwolke einer Brücke.



(b) MLS-Karte [45].



(c) Standard-Höhenkarte.



(d) Variation einer Höhenkarte [36].

Abbildung 3.1: Vergleich verschiedener Möglichkeiten, eine dreidimensionale Umgebung zu repräsentieren. Im Gegensatz zu Höhenkarten, können MLS-Karten den Brückendurchgang und auch die Brücke selbst repräsentieren. (Abbildung mit freundlicher Genehmigung von Patrick Pfaff.)

die Höhe aller Punkte innerhalb einer Zelle, wodurch sich die in Abb. 3.1c dargestellte Karte ergibt. Der Brückendurchgang kann nicht modelliert werden – vielmehr stellt er für den Roboter nun ein Hindernis dar. Eine von Pfaff und Burgard [36] vorgestellte Variante ist in der Lage den Durchgang zu erkennen, kann jedoch die Brücke selbst nicht modellieren (Abb. 3.1d).

MLS-Karten [45] können als eine Erweiterung von Höhenkarten angesehen werden. Sie bestehen ebenfalls aus einem zweidimensionalen Gitter mit frei wählbarer Zellgröße. Im Gegensatz zu Höhenkarten können die Zellen des Gitters jedoch mehrere sogenannter Patches beinhalten (Abb. 3.2). Ein Patch repräsentiert eine Oberfläche, meist also den Boden. Im Fall des gerade angesprochenen Beispiels der Brücke enthielte das Gitter im Bereich des Brückendurchgangs pro Zelle zwei Patches: ein Patch, der die Oberfläche der unterhalb der Brücke verlaufenden Straße modelliert und ein Patch, der die Oberfläche der Straße auf der Brücke repräsentiert (Abb. 3.1b). Im Gegensatz zur Verwendung von Höhenkarten konnte damit erreicht werden, dass die Struktur sowohl des Brückendurchgangs, als auch der Brücke selbst bewahrt wird.

Die Höheninformation eines Patches wird durch eine Gaußverteilung mit Mittelwert μ und Varianz σ^2 modelliert, wodurch es möglich ist, die Unsicherheit auszudrücken, mit der die Schätzung der vertikalen Höhe behaftet ist. Neben Mittelpunkt und Varianz verfügt jeder Patch noch über eine vertikale Tiefe d, welche angibt, wieviel des Bereichs unterhalb des Patches belegt ist. Dies erlaubt es auch vertikale

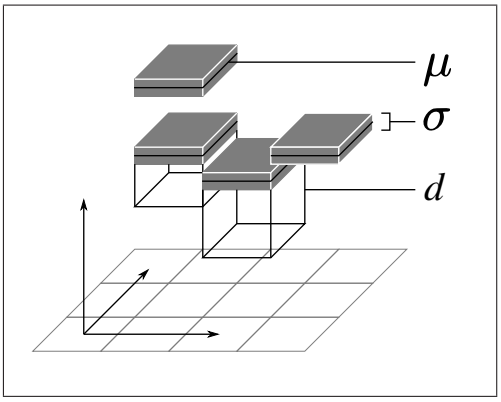


Abbildung 3.2: Schematischer Aufbau einer MLS-Karte. Die Patches werden in einem 2D-Gitter gespeichert und durch Mittelwert μ, Standardabweichung σ und vertikale Tiefe d parametrisiert. Zellen können auch mehrere Patches enthalten (siehe linke Zelle).

Strukturen, z. B. Hauswände, zu kartieren. Durch MLS-Karten wird eine kompakte, dreidimensionale Umgebungsmodellierung erreicht, die alle für die Navigation und Lokalisierung notwendigen Informationen bewahrt.

Die folgenden Abschnitte geben einen Überblick der für die Registrierung neuer Informationen relevanter Schritte zum Aufbau einer globalen MLS-Karte. Für eine detailliertere Darstellung sei auf [45] verwiesen.

3.1 Kartenerstellung aus einer Punktwolke

Eine Punktwolke eines dreidimensionalen Laserscans besteht aus einer Menge von n Punkten $\{p_1, \ldots, p_n\}$ im dreidimensionalen Raum $\mathbb{R}^3$. Jeder Punkt ist mit einer Varianz σ_i^2 assoziiert, die direkt proportional zur gemessenen Distanz ist. Den Zellen werden jeweils jene Punkte zugewiesen, deren x-y-Koordinaten innerhalb dieser Zelle liegen. Für jede Zelle wird aus den in ihr enthaltenen und nach z-Koordinaten sortierten Punkten eine Menge von vertikalen Intervallen berechnet. Solange zwei aufeinanderfolgende z-Koordinaten einen Mindestabstand γ unterschreiten, werden sie als zum gleichen Intervall zugehörig definiert, womit gleichzeitig sichergestellt ist, dass die Intervalle mindestens einen Abstand von γ besitzen. Der Mindestabstand wird so gewählt, dass er mindestens der Roboterhöhe entspricht (der hier verwendete Wert liegt bei $\gamma = 1$ m) und der Roboter somit in jede vertikale Lücke zwischen den Patches einer Zelle passt. Die Intervalle werden dann anhand ihrer Intervalllänge in horizontale und vertikale Intervalle kategorisiert (der verwendete Grenzwert liegt bei 10 cm). Ein vertikales Intervall kann in ein Patch umgewandelt werden, indem der Mittelwert und die Varianz des höchstgelegenen Punkts in diesem Intervall über-

nommen werden und die vertikale Tiefe gleich der Länge des Intervalls gesetzt wird. Bei horizontalen Intervallen wird die vertikale Tiefe auf Null gesetzt und der Mittelwert und die Varianz ergibt sich durch die eindimensionale Kalman-Filterung aller z-Koordinaten und den mit ihnen assoziierten Varianzen (vgl. Kap. A.3, S. 75).

3.2 Aktualisierung der Karte

Wenn ein neue Messung (p, σ) in die Karte eingefügt werden soll, muss entschieden werden, ob die Messung zu einem bereits in der Karte existierenden Patch gehört, oder als neuer Patch der Karte hinzugefügt werden muss. Dazu wird zunächst die Zelle (i, j) bestimmt, innerhalb derer die Messung liegt. In dieser Zelle wird der Patch $(\mu_{ij}^k, \sigma_{ij}^k)$ bestimmt, dessen Mittelwert am nächsten zur Höhe p_3 des gemessenen Punkts liegt. Falls der vertikale Abstand $\left|\mu_{ij}^k - p_3\right|$ kleiner als $3\sigma_{ij}^k$ ist, wird der Patch mit der neuen Messung durch eine eindimensionale Kalman-Filterung aktualisiert. Liegt die Messung außerhalb des $3\sigma_{ij}^k$-Bereichs, wird überprüft, ob die Messung Teil des vertikalen Bereichs eines Patches innerhalb der Zelle ist – in diesem Fall kann die Messung ignoriert werden. Ansonsten wird die Messung als neuer Patch eingefügt.

3.3 Scan-Matching

Für jeden durchgeführten 3D-Scan wird aus der 3D-Punktwolke eine lokale Karten m_i^L, mit assoziierter Positionsschätzung $p_i = (x, y, z, \phi, \vartheta, \psi)$ erstellt. Sollen aus mehreren lokalen Karten $m_1^L, \ldots, m_t^L$ eine globale Karte m_t^G erstellt werden, ist es notwendig, die Positionsschätzungen p_i durch einen Abgleich der lokalen Karten zu verbessern, da Positionsschätzungen der Lokalisierung aufgrund ungenauer Odometrieinformationen und verrauschten Sensordaten nie als fehlerfrei angesehen werden können.

Ziel des Abgleichs zweier lokaler Karten m und m' ist es, eine Rotationsmatrix R und einen Translationsvektor t zu finden, der die relative Lage der Karten zueinander beschreibt und durch die Minimierung einer Fehlerfunktion $e(R, t)$ durch den ICP-Algorithmus (*iterative closest point*) bestimmt wird. Aus den zwei Karten werden zwei Mengen X und Y bestimmt, die jeweils dreidimensionale Punkte mit assoziierter Varianz beinhalten und den Patches entsprechen. Der ICP-Algorithmus berechnet dann eine Menge von Indexpaaren $C = \{(i_1, j_1), \ldots, (i_k, j_k)\}$, sodass jeweils Punkt

$x_i \in X$ mit Punkt $y_j \in Y$ assoziiert ist. Die Fehlerfunktion ist dann definiert als

$$e(R,t) = \frac{1}{|C|} \sum_{(i,j)\in C} \left(x_i - (Ry_j + t)\right)^T \Sigma^{-1} \left(x_i - (Ry_j + t)\right) \tag{3.1}$$

$$= \frac{1}{|C|} \sum_{(i,j)\in C} d(x_i, y_j), \tag{3.2}$$

wobei $d(\cdot,\cdot)$ der quadrierten Mahalanobisdistanz entspricht. Die Autoren stellten jedoch fest, dass die Transformation (R,t) stabiler und effizienter berechnet werden kann, wenn die Punkte in Klassen eingeteilt werden und nur Punkte innerhalb der gleichen Klasse assoziiert werden. Dazu werden die Punkte in drei Mengen U, V, W für Karte m, bzw. U', V', W' für Karte m' aufgeteilt. U enthält alle Punkte aus m die aus vertikalen Patches erzeugt wurden, wobei zufällig vier Punkte pro Meter vertikaler Tiefe erzeugt werden. V enthält Punkte von traversierbaren Patches und W von nicht traversierbaren Patches (entsprechend enthalten U', V', W' Punkte aus Karte m'). Der ICP-Algorithmus erstellt für jede Kategorie eine Menge von Indexpaaren (C_u, C_v, C_w). Die Fehlerfunktion lässt sich dann definieren als

$$e(R,t) = \sum_{(i,j)\in C_u} d_v(u_i, u_j) + \sum_{(i,j)\in C_v} d(v_i, v_j) + \sum_{(i,j)\in C_w} d(w_i, w_j), \tag{3.3}$$

wobei $d_v(\cdot,\cdot)$ die jeweils tiefsten Punkte der jeweiligen Zellen vergleicht.

Das Scan-Matching-Verfahren verbessert die Positionsschätzung einer lokalen Karte durch den Vergleich mit der vorherigen lokalen Karte. Doch auch durch dieses Verfahren können kleinere Fehler nicht ausgeschlossen werden. Da aber neue lokale Karten jeweils nur zur letzten Karte verglichen werden, können sich über die Zeit kleine Fehler akkumulieren und die entstehende globale Karte für die Navigation eventuell unbrauchbar werden. Kehrt der Roboter jedoch zu einer vorher besuchten Stelle zurück, ist auch der Vergleich zweier zeitlich weiter auseinanderliegender Karten möglich. Dadurch stehen neue Informationen bezüglich der relativen Lage der lokalen Karten zueinander zur Verfügung, die mittels eines iterativen Verfahrens durch den Graph der Nachbarschaftsbeziehungen der lokalen Karten propagiert werden kann, um global konsistente Positionsschätzungen zu berechnen („*loop closing*“ für Details siehe [45]).

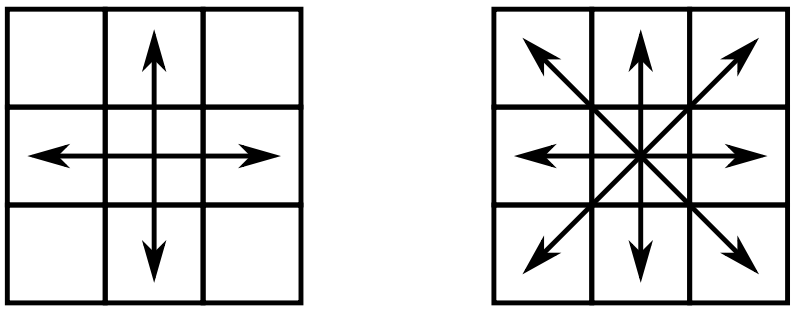
Kapitel 4

Explorationsstrategie für MLS-Karten

In diesem Kapitel beschreiben wir unseren Explorationsansatz. Die ersten Abschnitte bis einschließlich Kapitel 4.4 werden auf notwendige Definitionen und Techniken eingehen. Es wird die Nachbarschaftsbeziehungen zwischen Patches eingeführt und die für die Navigation notwendige Traversierbarkeitsanalyse vorgestellt. Bei der Exploration müssen wir herausfinden, welche Patches bereits exploriert sind und werden darauf aufbauend bestimmen, welche Patches an der Grenze zu noch nicht explortiertem Gebiet liegen. Zudem stellen wir eine 3D-Raycasting-Methode für MLS-Karten vor, die bei der Generierung und Auswahl von Zielpunkten und Zwischenzielen eingesetzt wird. Die Kapitel 4.5 und 4.6 beschreiben die eigentliche Explorationsstrategie. Es werden zunächst Kandidatenzielpunkte bestimmt und diese bewertet, um den besten als Zielpunkt auszuwählen. Die Bewertung wird die Fahrtkosten zu einem Zielpunkt und den an diesem Punkt zu erwartenden Informationsgewinn berücksichtigen. In Kapitel 4.7 werden wir darauf eingehen, wie wir auf dem Weg zum ausgewählten Zielpunkt Zwischenziele bestimmen, um zu gewährleisten, dass für das Scan-Matching-Verfahren genügend Überlappung zwischen zwei lokalen Karten besteht. Schließlich wird im letzten Teilkapitel darauf eingegangen, wann der Explorationsprozess endet.

4.1 Nachbarschaft

In zweidimensionalen Gitterkarten sind die Nachbarn einer Zelle eindeutig durch das Gitter bestimmt. Die gebräuchlichsten Nachbarschaftsbeziehungen sind die 4- und 8-Zellen-Nachbarschaft (Abb. 4.1). Wir werden die 8-Zellen-Nachbarschaft verwenden, da sie den tatsächlichen Freiheitsgraden des Roboters näher kommt, als die 4-Zellen-Nachbarschaft. Die 8-Zellen-Nachbarschaft muss jedoch für den Einsatz mit MLS-Karten angepasst werden. Denn MLS-Karten basieren zwar ebenfalls auf einem



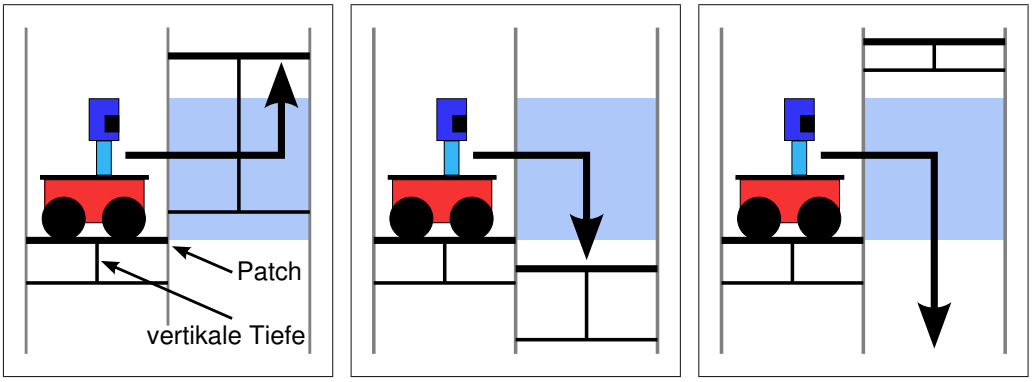
(a) 4-Zellen-Nachbarschaft. **(b)** 8-Zellen-Nachbarschaft.

Abbildung 4.1: Gebräuchliche Nachbarschafts-Definitionen.

zweidimensionalen Gitter, die grundlegenden Einheiten sind jedoch nicht Zellen, sondern Patches, die eine dreidimensionale Struktur besitzen.

Um die Nachbarn eines Patches zu bestimmen, muss für jede der acht Nachbarzellen entschieden werden, welcher der darin enthaltenen Patches die Kriterien eines Nachbar-Patches erfüllt. Eine mögliche Definition wäre denjenigen Patch zu wählen, der den vertikalen Abstand zwischen den Nachbar-Patches minimieren würde, wobei man hier entweder allein nach dem Mittelwert oder auch unter Berücksichtigung der vertikalen Tiefe des Nachbar-Patches entscheiden könnte. Diese Definition hat allerdings den Nachteil, dass ein Patch als Nachbar definiert werden kann, obwohl er vom Roboter niemals erreicht würde. Sind etwa in einer Nachbarzelle zwei flache Patches enthalten, wobei der eine drei Meter unterhalb, der andere zwei Meter oberhalb der Position des Roboters liegt, dann würde der obere Patch als Nachbar bestimmt werden, obwohl der Roboter beim Einfahren in diese Zelle auf dem unteren Patch landen würde.

Motiviert durch dieses Beispiel werden wir stattdessen als Nachbar-Patch denjenigen Patch einer Nachbarzelle definieren, den der Roboter berühren würde, wenn er in diese Zelle einfährt. Ist das vertikale Intervall, das der Roboter in der Nachbarzelle einnehmen würde, durch ein Patch belegt, wird dieser als ein Nachbar angesehen (Abb. 4.2a). Da bei der Konstruktion der MLS-Karten die Roboterhöhe als vertikaler Mindestabstand zwischen Patches berücksichtigt wird, ist sichergestellt, dass in diesem Intervall nicht mehr als ein Patch enthalten sein kann. Ist das Intervall frei, bleiben noch zwei Möglichkeiten: Der Roboter fällt auf ein Patch (Abb. 4.2b), wodurch dieser der Nachbar-Patch wird, oder er fällt ins „Bodenlose“ (Abb. 4.2c), wodurch diese Zelle kein Nachbar-Patch enthielte, selbst wenn überhalb des Inter-



(a) Nachbarschaft durch belegtes Intervall. **(b)** Nachbarschaft durch freien Fall. **(c)** Keine Nachbarschaft.

Abbildung 4.2: Bestimmung des Nachbar-Patches durch das vertikale Intervall (hellblau), das der Roboter beim Einfahren in diese Zelle einnehmen würde. Zu Illustrationszwecken wurde die Zellgröße so gewählt, dass der Roboter ganz in ihr enthalten ist. Durch ein Patch belegte Bereiche sind durch schwarzen Balken dargestellt.

valls Patches existierten. Die so ermittelte Menge an Nachbarn eines Patches p soll im Folgenden als $N(p)$ bezeichnet werden.

4.2 Traversierbarkeitsanalyse

Für eine autonome Navigation ist es unabdingbar, dass Hindernisse der Umgebung erkannt und befahrbare Bereiche bestimmt werden können. Die meisten Ansätze zur zweidimensionalen Kartierung gehen von einer flachen Umgebung aus, die nur positive Hindernisse, wie etwa Wände oder Tische, beinhaltet. Im dreidimensionalen Fall, besonders bei der Kartierung im Freien, sind zudem auch die Eigenschaften unebenen Geländes und negative Hindernisse, etwa Treppen, zu berücksichtigen.

Die Traversierbarkeitanalyse wird daher drei Faktoren miteinbeziehen: positive und negative Hindernisse, sowie die Neigung und Rauheit des Geländes (vgl. [13] für einen ähnlichen Ansatz). Dazu werden zunächst für jeden Patch p drei Traversierbarkeitswerte $\tau_n(p)$ (Neigung), $\tau_r(p)$ (Rauheit) und $\tau_h(p)$ (Hindernis) berechnet. Der kombinierte Wert ergibt sich als

$$\tau(p) = \tau_n(p) \cdot \tau_r(p) \cdot \tau_h(p). \tag{4.1}$$

Als Hindernisse definieren wir nun alle Patches mit $\tau(p) = 0$, alle anderen Patches sind traversierbar. Für die Planung eines kollisionsfreien Pfades ist es günstig, wenn Hindernisse um den Durchmesser des Roboters vergrößert werden (*obstacle*

growing). Dadurch kann der Roboter während der Pfadplanung als punktförmig angenommen werden und die Kollisionsdetektionen reduziert sich auf die Überprüfung, ob ein gegebener Patch traversierbar ist. Wir werden deshalb die ermittelten Traversierbarkeitswerte über eine Distanz, die vom Durchmesser des Roboters abhängt, propagieren. Zudem werden wir die Traversierbarkeitskarte falten, um Informationen über die direkte Nachbarschaft von Patches hinaus einbeziehen zu können.

4.2.1 Neigung

Um die Neigung des Geländes an einem Patch p zu bestimmen, wird mittels der Methode der kleinsten Quadrate (*least squares fit*) eine Ebene eingepasst [16]. Die dabei zu minimierende Fehlerfunktion g gibt den vertikalen Abstand der Nachbar-Patches $N(p)$ zur Ebene an. Dabei wird angenommen, dass die Ebene durch den Patch p geht, und die Koordinaten der Nachbarpunkte relativ zu p angegeben sind. Zu schätzen sind dann die Steigung s_1 der Ebene in x-Richtung und die Steigung s_2 in y-Richtung. Die relative Position eines Nachbar-Patches n sei durch $(n_1, n_2, n_3)^T$ gegeben. Die zu minimierende Funktion ist dann

$$g(s_1, s_2) = \sum_{n \in N(p)} (n_3 - (n_1 s_1 + n_2 s_2))^2. \tag{4.2}$$

Um die gesuchten Parameter s_1 und s_2 zu finden, berechnen wir die partiellen Ableitungen

$$\frac{\partial g}{\partial s_1} = 2 \sum_{n \in N(p)} (n_3 - (n_1 s_1 + n_2 s_2))(-n_1) \tag{4.3}$$

$$\frac{\partial g}{\partial s_2} = 2 \sum_{n \in N(p)} (n_3 - (n_1 s_1 + n_2 s_2))(-n_2) \tag{4.4}$$

und setzen

$$\frac{\partial g}{\partial s_1} = 0 \qquad\qquad \frac{\partial g}{\partial s_2} = 0. \tag{4.5}$$

Als Lösung dieser Gleichungen erhalten wird die Steigungen der best eingepassten Ebene als

$$s_1 = \frac{\left(\sum n_1 n_3\right)\left(\sum n_2^2\right) - \left(\sum n_1 n_2\right)\left(\sum n_2 n_3\right)}{\left(\sum n_1^2\right)\left(\sum n_2^2\right) - \left(\sum n_1 n_2\right)^2} \tag{4.6}$$

$$s_2 = \frac{\left(\sum n_1 n_3\right)\left(\sum n_1 n_2\right) - \left(\sum n_2 n_3\right)\left(\sum n_1^2\right)}{\left(\sum n_1 n_2\right)^2 - \left(\sum n_1^2\right)\left(\sum n_2^2\right)}. \tag{4.7}$$

Hierbei steht $\sum$ als Abkürzung für $\sum_{n \in N(p)}$. Existieren weniger als zwei Nachbar-Patches, ist das Minimierungsproblem unterbestimmt. Für den Fall, dass keine Nachbarn existieren, nehmen wir daher eine Ebene parallel zur x-y-Ebene an und setzen also $s_1 = s_2 = 0$. Für den Fall, dass nur ein Nachbar existiert, nehmen wir eine Ebene an, die den geringsten Neigungswinkel zur x-y-Ebene aufweist. Dazu setzen wir $s_1 = \frac{n_3}{n_1}$ und $s_2 = \frac{n_3}{n_2}$, falls n_i nicht Null ist, sonst wird s_i ebenfalls auf Null gesetzt. Sind die gesuchten Steigungen bestimmt, ist die Oberflächennormale

$$v = \frac{(1, 0, s_1)^T \times (0, 1, s_2)^T}{|(1, 0, s_1)^T \times (0, 1, s_2)^T|} \tag{4.8}$$

und der Neigungswinkel ergibt sich als

$$\alpha = \cos^{-1}(v_3). \tag{4.9}$$

Die Traversierbarkeit $\tau_n(p) \in [0, 1]$ unter Berücksichtigung eines gegebenen maximalen Neigungswinkels $\alpha_{\max}$ ist schließlich definiert als

$$\tau_n(p) = \begin{cases} 0 & \alpha \geq \alpha_{\max} \\ 1 - \frac{\alpha}{\alpha_{\max}} & \alpha < \alpha_{\max} \end{cases}. \tag{4.10}$$

4.2.2 Rauheit

Die Rauheit wird über das normalisierte Residuum der Minimierungsfunktion g in (4.2) definiert und entspricht der durchschnittlichen, quadrierten, vertikalen Abweichung der Patches von der eingepassten Ebene:

$$\rho = \frac{g(s_1, s_2)}{|N(p)|} = \frac{1}{|N(p)|} \sum_{n \in N(p)} (n_3 - (n_1 s_1 + n_2 s_2))^2. \tag{4.11}$$

Ähnlich der Definition der Traversierbarkeit $\tau_n(p)$, ergibt sich die Traversierbarkeit $\tau_r(p) \in [0, 1]$ bezüglich der Rauheit durch einen gegebenen maximalen Rauheitswert $\rho_{\max}$ als

$$\tau_r(p) = \begin{cases} 0 & \rho \geq \rho_{\max} \\ 1 - \frac{\rho}{\rho_{\max}} & \rho < \rho_{\max} \end{cases}. \tag{4.12}$$

4.2.3 Hindernisse

Im Gegensatz zu den zwei oben definierten Traversierbarkeitswerten, ist die Entscheidung, ob ein Patch ein Hindernis darstellt, eine kategoriale Entscheidung. Ein Patch wird als nicht traversierbar angesehen, falls die maximale vertikale Abweichung

$$\delta = \max_{n \in N(p)} (n_3 - (n_1 s_1 + n_2 s_2)) \tag{4.13}$$

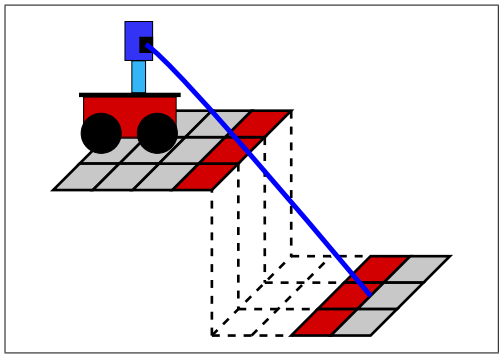


Abbildung 4.3: Wird eine Lasermessung an einem Abgrund vorgenommen, sind einige Patches nicht sichtbar. Über die geringere Anzahl an Nachbarn der Patches am Rand des Abgrunds, kann dieser dennoch als Hindernis erkannt werden.

eines Nachbars einen gegebenen Grenzwert $\delta_{\max}$ überschreitet. In der Praxis sollte der Grenzwert dem maximalen Höhenunterschied entsprechen, den der Roboter gerade noch in der Lage ist zu bewältigen. Durch den maximalen Abstand können bereits positive und negative Hindernisse erkannt werden. Es treten jeden auch Situationen auf, in denen negative Hindernisse damit noch nicht erfasst werden. In der in Abb. 4.3 dargestellten Situation liegen einige Patches in einem für den Laser nicht einsehbaren Bereich und die Patches am Rand des Abgrunds besitzen somit weniger als acht Nachbarn. Die vorhandenen Nachbarn implizieren eine gute Traversierbarkeit, da die eingepasste Ebene keine Neigung aufweist. Um auch solche Patches am Rande eines Abgrunds dennoch als Hindernis erkennen zu können, fordern wir zusätzlich, dass ein traversierbarer Patch alle acht Nachbarn besitzen muss. Die Traversierbarkeit $\tau_h(p) \in \{0, 1\}$ eines Patches p bezüglich positiver und negativer Hindernisse definieren wir somit als

$$\tau_h(p) = \begin{cases} 1 & \delta < \delta_{\max} \wedge |N(p)| = 8 \\ 0 & \text{sonst} \end{cases}. \tag{4.14}$$

4.2.4 Faltung der Traversierbarkeitskarte

Die so ermittelte Traversierbarkeit bezieht nur direkte Nachbarn eines Patches in die Berechnung mit ein und die Information über das Gelände ist damit äußerst lokal begrenzt. Die Zellgröße wird jedoch meist um ein vielfaches kleiner sein, als die Fläche, die der Roboter einnimmt. Für die Navigation ist daher weniger die Traversierbarkeit eines einzelnen Patches interessant, als vielmehr die Traversierbarkeit der gesamten vom Roboter eingenommenen Fläche. Eine Möglichkeit dem Rechnung zu tragen wäre die Einpassung einer Ebene, die sich nicht nur über die Nachbar-Patches

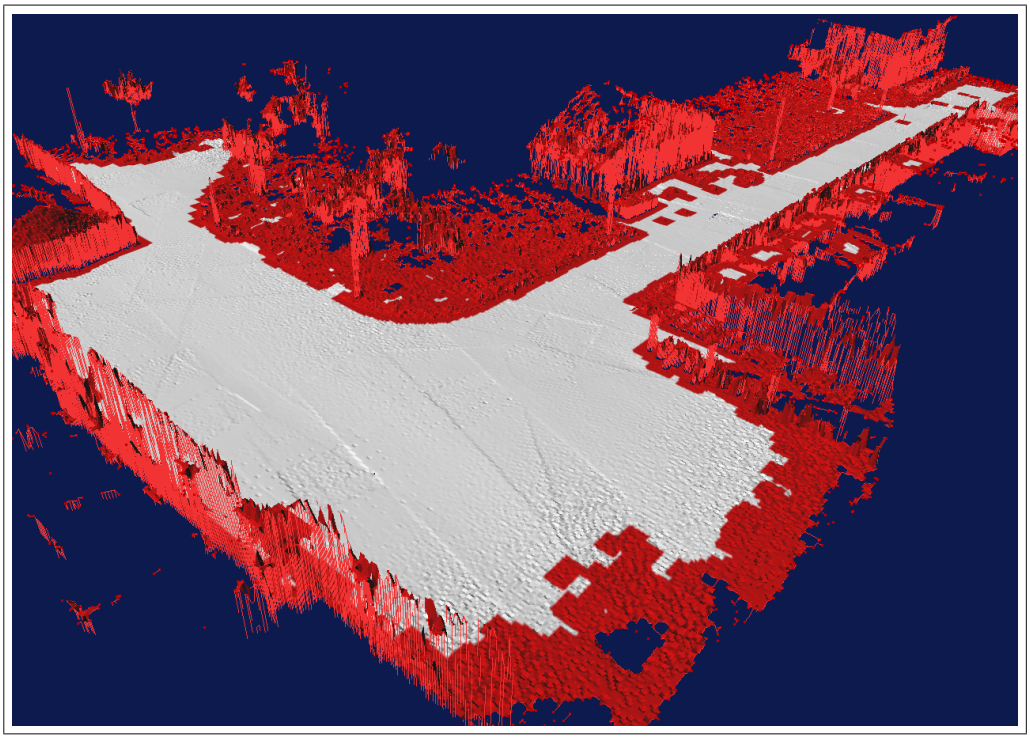


Abbildung 4.4: Traversierbarkeitskarte des teilweise explorierten Campus der Universität Freiburg. Weiße Patches sind traversierbar ($\tau(p) > 0$), rote Patches nicht ($\tau(p) = 0$).

erstreckt, sondern über die gesamte vom Roboter eingenommenen Fläche. Dies würde man sich jedoch mit dem Nachteil erkaufen, kleinere Unstetigkeiten im Gelände nicht mehr erkennen zu können.

Der hier gewählte Ansatz ist ein anderer. Die Traversierbarkeitskarte wird diskret gefaltet und dabei ein (diskreter) Gauß-Glättungskern eingesetzt, welcher der vom Roboter eingenommenen Fläche entspricht. Wir nehmen vereinfachend an, dass diese Fläche quadratisch ist – in der Praxis sollte die Größe des Quadrats so gewählt werden, dass der Roboter ganz in ihm enthalten ist. Die Implementierung der Faltung vereinfacht sich, wenn man die Tatsache ausnutzt, dass die Glättung mit einem großen Gaußkern der wiederholten Faltung mit einem kleineren Gaußkern entspricht. Es wird daher ein Gaußkern

$$G = \begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix} \tag{4.15}$$

gewählt, der sich gerade über die direkte Nachbarschaft eines Patches erstreckt. Die Anzahl $i_{\max}$ der notwendigen Iterationen ergibt sich aus der Zellgröße z, dem maxi-

malen Durchmesser des Roboters d und einem Sicherheitsabstand s

$$i_{\max} = \frac{\frac{d}{2} + s}{z}. \tag{4.16}$$

Das iterative Falten der Karte kann gleichzeitig zur Vergrößerung von Hindernissen verwendet werden. Wir führen daher keine Faltung durch, falls einer der Nachbarn ein Hindernis darstellt, also nicht traversierbar ist ($\tau = 0$), sondern setzen in diesem Fall die Traversierbarkeit direkt auf Null. Für nicht existente Nachbarn nehmen wir $\tau = 0.5$ an. Wir erhalten folgende Iterationsvorschrift:

$$\tau_{i+1}(p) \leftarrow \begin{cases} 0 & \exists n \in N(p) \cup p : \tau_i(n) = 0 \\ \frac{1}{9} \sum\limits_{n \in N(p) \cup p} \tau_i(n) \cdot G(n_1 - p_1, n_2 - p_2) & \text{sonst} \end{cases}. \tag{4.17}$$

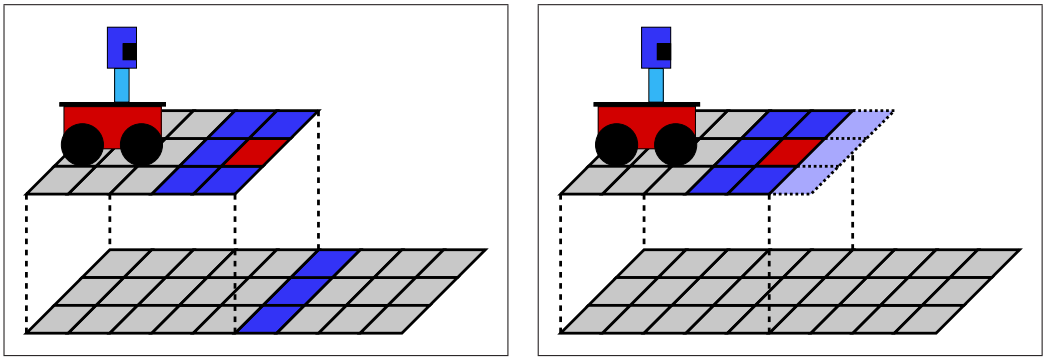
Nach Durchführung aller Iterationsschritte sind die endgültigen Traversierbarkeitswerte aller Patches festgelegt.

4.3 Exploriertheit

Die meisten Explorations-Algorithmen erfordern, dass bereits explorierte und noch zu explorierende Bereiche der Karte unterschieden werden können. Dies ist einerseits notwendig um festzustellen, wann ein bestimmtes Areal ausreichend exploriert ist und der Explorationsprozess terminiert werden kann, andererseits beeinflusst die Lage der nicht explorierten Bereiche den weiteren Verlauf der Exploration. Auch das hier vorgestellte Verfahren ist auf die Unterscheidung in explorierte und nicht explorierte Bereiche angewiesen.

Bei Verwendung von Belegtheitskarten können explorierte Zellen als Zellen definiert werden, die eine a-priori-Belegtheitswahrscheinlichkeit besitzen [48]. Im Endeffekt sind das genau diejenigen Zellen, die noch nicht aktualisiert wurden, da sie bisher nicht im Bereich einer Lasermessung lagen. Im Gegensatz zu Belegtheitskarten, die eine explizite Modellierung freier und belegter Bereiche zulassen, werden in MLS-Karten durch Patches nur belegte Bereiche modelliert. Ob die vertikalen Intervalle über oder unter den Patches einem freien oder einem noch nicht von einem Laserstrahl vermessenen Bereich entsprechen, kann nicht direkt entschieden werden.

Um zu entscheiden, ob ein Patch als exploriert angesehen werden kann, werden wir dessen lokale Nachbarschaft sowie die Unsicherheit beachten, mit der seine vertikale Positionsschätzung behaftet ist. Bezüglich der Nachbarschaft eines Patches legen wir die Annahme zugrunde, dass sich die Umgebung stetig ändert. Wir erwarten daher für einen Patch p in jeder seiner Nachbarzellen einen Patch, der innerhalb eines



(a) Nachbarschaft $N(p)$. **(b)** Intervallnachbarschaft $N_I(p)$.

Abbildung 4.5: Zur Motivation der Einführung einer Intervallnachbarschaft $N_I(p)$. Der rot markierte Patch p besitzt eine vollständige Nachbarschaft $N(p)$, jedoch eine unvollständige Intervallnachbarschaft $N_I(p)$ mit nur fünf Patches (dunkelblau) – die drei hellblauen Patches zeigen an, in welcher Region weitere Patches vermutet werden.

vertikalen Toleranzbereichs $[\mu - c_n, \mu + c_n]$ um den Mittelwert μ von p liegt. Wir wollen die Menge der in diesen Bereichen gefundenen Patches die Intervallnachbarschaft $N_I(p)$ von p nennen – in Abgrenzung zu der in Kapitel 4.1 im Hinblick auf Navigationsaufgaben und Traversierbarkeitsanalysen eingeführte Nachbarschaftsdefinition $N(p)$. Wir fordern, dass ein Patch eine vollständige Intervallnachbarschaft von acht Intervallnachbarn besitzen muss, um als exploriert zu gelten. Dies wird von nahezu allen erkundeten, befahrbaren Bereiche der Umgebung erfüllt: dem Boden, dem Übergang vom Boden zur Wand, etc. Patches, die am Rande von negativen Hindernissen liegen, etwa Treppen oder Abgründen, erfüllen diese Forderung jedoch nicht, selbst wenn ihre nähere Umgebung vollständig kartiert ist. Für diese Patches müssen wir zunächst annehmen, dass sie nicht exploriert sind. Im folgenden Kapitel beschreiben wir, wie wir durch die Verfolgung der Exploriertheit über die Zeit diese irrige Annahme korrigieren können. Der weitaus häufigere Fall einer unvollständigen Intervallnachbarschaft tritt jedoch dann ein, wenn die Intervalle tatsächlich noch nicht erfasste Patches enthalten müssten. Dies ist z. B. praktisch immer am Rand der Karte der Fall.

Es sei darauf hingewiesen, dass wir hier nicht die eingangs eingeführte Nachbarschaftsdefinition $N(p)$ nutzen können. Wird vom Roboter z. B. zuerst das Erdgeschoss eines Hauses exploriert und danach das darüber liegende Obergeschoss, dann besitzen die Patches dieses Obergeschosses eine vollständige Nachbarschaft $N(p)$, die auch Patches aus dem Erdgeschoss miteinschließt (Abb. 4.5a). Die Intervallnachbarschaft $N_I(p)$ enthält jedoch keine Patches aus dem Erdgeschoss und ist somit unvollständig für jene Patches, die am Rand des explorierten Bereichs des Oberge-

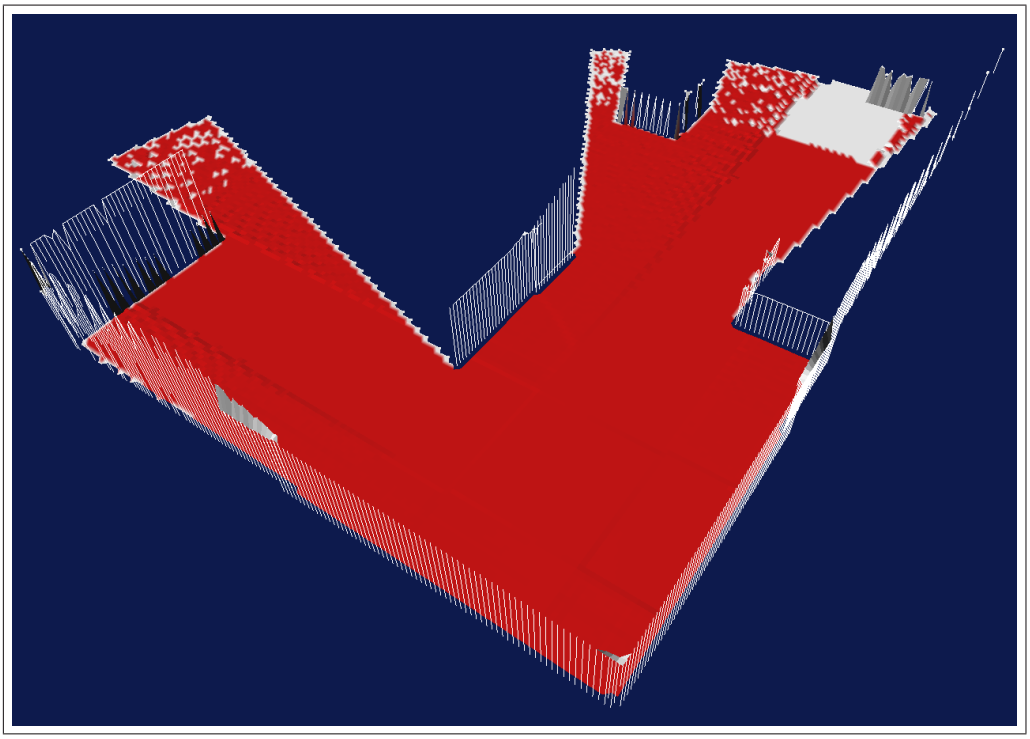


Abbildung 4.6: MLS-Karte, in der explorierte Patches rot markiert wurden.

schosses liegen (Abb. 4.5b).

Neben der Anzahl an Intervallnachbarn eines Patches sind wir daran interessiert, dass ein Patch die Umgebung an dieser Stelle möglichst akkurat modelliert. Da die Patches als Gauß-Verteilungen repräsentiert sind, verfügen wir mit der Varianz über ein Maß der Unsicherheit mit der die Schätzung der vertikalen Position des Patches behaftet ist. Da wir später bei der Auswertung von anzufahrenden Zielpunkten die Entropie der Patches verwenden werden, wollen wir aus Gründen der Einheitlichkeit auch an dieser Stelle die Entropie [11]

$$H = \frac{1}{2}\log_2(2\pi e\sigma^2) \tag{4.18}$$

verwenden, die jedoch direkt proportional zum Logarithmus der Varianz σ^2 ist.[1] Ein Patch gilt nun als exploriert, falls er acht Intervallnachbarn besitzt und seine Entropie unterhalb einem Grenzwert $H_{\max}$ liegt.

In der MLS-Karte in Abb. 4.6 wurden die explorierten Patches rot markiert. Man erkennt, dass die nicht explorierten Patches jene sind, die am Rand der Karte liegen

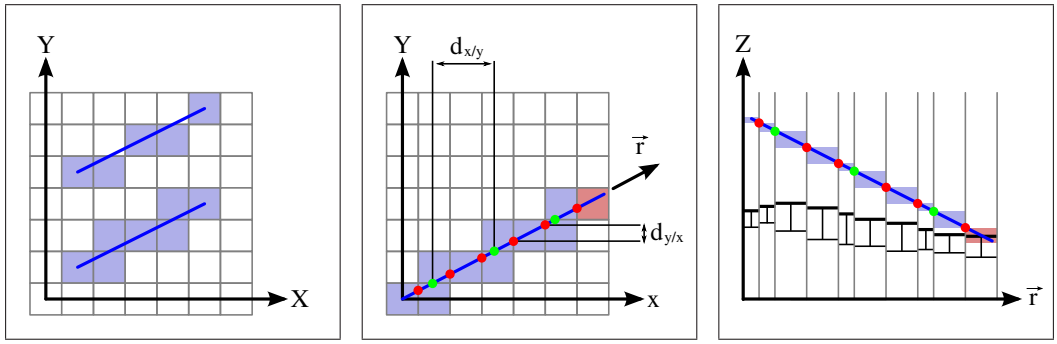
[1] Die Entropie spiegelt die Unsicherheit einer Wahrscheinlichkeitsverteilung wider und kann für diskrete und kontinuierliche Verteilungen definiert werden.

und somit zu wenig Intervallnachbarn besitzen, sowie einige weit entfernte Patches, deren Entropie zu hoch ist. Ein 3D-Scan besteht aus vier Teilscans, die sich teilweise überlappen: der Laserscanner fährt vier horizontale Winkel im Abstand von 90 Grad an und wird an jeder dieser Stellen gekippt. Der relativ große unexplorierte Bereich oben rechts im Bild befand sich nur im Messbereich eines Teilscans, weswegen die Entropie der Patches in diesem Bereich höher ist, als die Entropie der Patches die in Bereichen lagen, die von zwei Teilscans erfasst wurden.

4.3.1 Verfolgung der Exploriertheit über die Zeit

Es gibt Situationen in denen ein Patch auch nach mehrfachen Messungen zu wenig Intervallnachbarn oder eine zu hohe Entropie besitzt, um als exploriert zu gelten. Es ist sinnvoll einen solchen Patch dennoch als exploriert einzustufen, da weitere Messungen den Zustand offensichtlich nicht beheben würden. Um solche problematischen Patches erkennen zu können, müssen beide Werte, d. h. Entropie und Anzahl der Intervallnachbarn, über mehrere Zeitschritte verfolgt werden. Ist die Entropie nach einer Messung über dem Grenzwert und konnte sie seit der letzten Messung des Patches nicht um einen Mindestwert verringert werden, wird ein Zähler heraufgesetzt. Übersteigt dieser Zähler einen Grenzwert, wird die Entropie als ausreichend gering angesehen, auch wenn sie den eigentlich einzuhaltenden Grenzwert H_{max} übersteigt. Entsprechend wird mit der Anzahl der Intervallnachbarn verfahren, wobei wir hier natürlich an einer Erhöhung der Intervallnachbarn interessiert sind. Dabei sind nur solche Patches zu aktualisieren, die durch den letzten Laserscan erfasst wurden.

Um die Änderung der entsprechenden Werte (Nachbarn, Entropie) der Patches zu verfolgen, müssen wir bestimmen, mit welchen Patches sie in der Karte vor der letzten Messung korrespondieren. Neu entdeckte Patches haben keine Entsprechung in der alten Karte, bereits bekannte Patches haben durch die neue Messung eventuell ihre vertikale Position innerhalb der Zelle geändert. Für die Assoziation von alten und neuen Patches innerhalb einer Zelle zweier zeitlich aufeinanderfolgenden Karten verwenden wir die ungarische Methode [26]. Die Kostenfunktion ist über den quadrierten, vertikalen Abstand der Patches definiert. Hat sich die Anzahl der Patches in der Zelle geändert, müssen zusätzliche Patches in die Kostenmatrix aufgenommen werden. Eine Assoziation eines Patches mit einem solchen zusätzlichen Patch bedeutet, dass der Patch „verloren ging“ oder neu hinzukam. Die Kosten für ein solches Ereignis sind Null. Um die Terminierung der Exploration zu gewährleisten, kann ein einmal als exploriert eingestufter Patch später nicht mehr als nicht exploriert eingestuft werden.



(a) Vergleich zum Bresenham-Algorithmus.

(b) Ablaufen der Zellen.

(c) Kollisionsdetektion.

Abbildung 4.7: Für das Raycasting in MLS-Karten werden zunächst die Zellen des 2D-Gitters abgelaufen und anschließend für jede Zelle eine Kollisiondetektion mit den in ihr enthaltenen Patches durchgeführt.

4.4 Raycasting

Will man die Lage potenzieller Zielpunkte bestimmen und deren Informationsgehalt abschätzen, wird es notwendig sein, dass mögliche Messungen des Roboters simuliert werden. Die Fragestellung hierbei ist, an welcher Position der Karte am meisten unexploriertes Gebiet zu sehen sein wird, bzw. welche Position den meisten Informationsgewinn verspricht. Die verwendete Methode um virtuelle Laserstrahlen in der Karte verfolgen zu können, wird in diesem Kapitel beschrieben.

Die Parametrisierung eines Laserstrahls könnte durch Orts- und Richtungsvektor erfolgen. Da wir aber auf einer uniformen Gitterkarte operieren, ist es von Vorteil eine Parametrisierung durch den Horizontalwinkel α_h (Azimut) und den Vertikalwinkel α_v (Elevation), sowie Startpatch p, Laserhöhe h und maximale Reichweite $r_{\max}$ zu wählen. Es wird angenommen, dass sich der Anfangspunkt A des Laserstrahls h Meter über dem Patch p befindet – und zwar zentriert innerhalb dieser Zelle. Der Endpunkt E wird exakt bestimmt, d. h. er wird nicht auf einen Zellmittelpunkt zentriert, da dies zwangsläufig einer Veränderung der Winkel gleichkommt.

Der erste Schritt ist nun das Ablaufen der Zellen im Gitter, die vom auf die x-y-Ebene projizierten Laserstrahl traversiert werden. Eine bekanntes und effizientes Verfahren zum Ablaufen von Zellen in einem uniformen, zweidimensionalen Gitter entlang einer Linie, ist der Bresenham-Algorithmus [8]. Für den hier verfolgten Verwendungszweck ist er jedoch eher ungünstig, da er einige Zellen auslässt, die vom Laserstrahl traversiert werden. Dies ist in Abb. 4.7a verdeutlicht, welche oben die vom Bresenham-Algorithmus traversierten Zellen illustriert und unten die tatsäch-

lich vom Laserstrahl durchlaufenen Zellen. Stattdessen werden, ähnlich dem in [12, S. 324-328] vorgestellten Verfahren die Schnittpunkte mit den Zellgrenzen abgelaufen. Dem Verfahren liegt die Feststellung zugrunde, dass die Schnittpunkte mit dem Gitter in zwei Klassen unterteilt werden können: solche, die auf Zellgrenzen liegen die parallel zur x-Achse verlaufen, und jene, die auf Zellgrenzen parallel zur y-Achse liegen. In Abb. 4.7b sind dies die roten, bzw. grünen Schnittpunkte. Die Abstände zwischen den jeweiligen Klassen von Schnittpunkten sind konstant – um den nachfolgenden Schnittpunkt einer Klasse zu bestimmen, genügt es also einen konstanten Vektor, der nur vom horizontalen Winkel des Lasers abhängt, zu addieren (Δ_x, bzw. Δ_y in Alg. 1, Zeile 6). Die aktuellen Schnittpunkte beider Klassen werden in zwei Variablen X und Y gehalten, die wir dann nach und nach um jeweils Δ_x, bzw. Δ_y inkrementieren. Die Schnittpunkte sind immer relativ zur Anfangszelle A angegeben, was einige nachfolgende Schritte vereinfacht. Anfangs setzen wir beide Schnittpunktvariablen auf die Zellgrenzen der Zelle im Nullpunkt (Alg. 1, Zeilen 7 und 8). In der Folge muss nun bestimmt werden, welcher der zwei aktuellen Schnittpunkte (X oder Y) als nächstes vom Laser durchlaufen wird und dieser Schnittpunkt dann durch seinen Nachfolgeschnittpunkt ($X + \Delta_x$ oder $Y + \Delta_y$) ersetzt werden. Da die Schnittpunkte relativ zur Anfangszelle angegeben sind, ist dies jeweils der Schnittpunkt mit der betragsmäßig kleineren x-Koordinate, oder gleichwertig, der betragsmäßig kleineren y-Koordinate. Um Sonderfälle abfangen zu können, in denen der Laserstrahl exakt parallel zur x- oder y-Achse verläuft, entscheiden wir uns jedoch in Zeile 9, ob wir den nächsten Schnittpunkt anhand der x- oder y-Koordinate wählen. Die in Zeile 2 berechnete Anzahl an traversierten Schnittpunkten ist zu hoch gewählt, falls Schnittpunkte zusammenfallen, was bspw. der Fall ist, wenn der Laserstrahl in einem horizontalen Winkel von 45° verläuft. Dieser Fall wird in Zeile 11 abgefangen und die Anzahl der Schnittpunkte um Eins verringert. Die korrekte Abfragebedingung wäre $|X_x - Y_x| = 0$, unter Verwendung von Floating-Point-Operationen müssen wir numerischen Ungenauigkeiten Rechnung tragen und einen kleinen Toleranzbereich ε zulassen.

Sind die zweidimensionalen Schnittpunkte mit dem Gitter bestimmt, kennen wir durch den vertikalen Winkel des Laserstrahls, den Startpatch und die Laserhöhe auch die dreidimensionalen Koordinaten. Jede vom Laserstrahl traversierte Zelle ist von zwei Schnittpunkten (oder einem Schnittpunkt und dem Endpunkt des Laserstrahls) begrenzt, die durch ihre z-Koordinaten das vertikale Intervall definieren, das der Laserstrahl in dieser Zelle überstreicht. Für die Kollisionsdetektion des Laserstrahls mit der Karte muss nun nur noch für jede traversierte Zelle überprüft werden, ob das jeweilige vertikale Intervall von einem Patch belegt ist (Abb. 4.7c). In Abb. 4.8 sind die Laserstrahlen einer simulierten Messung in einer MLS-Karte dargestellt.

Algorithmus 1 Schnittpunkte einer Linie mit einem uniformen 2D-Gitter.

Eingabe: Anfangszelle $A = (A_x, A_y)^T$, Endpunkt $E = (E_x, E_y)^T$

Ausgabe: traversierte Zellen $Z^{[i]}$, Schnittpunkte $S^{[i]}$, Anzahl Schnittpunkte $anzSP$

1: $Z^E \leftarrow A + \text{runden}(E - A)$

2: $anzSP \leftarrow \left|A_x - Z_x^E\right| + \left|A_y - Z_y^E\right|$

3: $\delta_x \leftarrow E_x - A_x$

4: $\delta_y \leftarrow E_y - A_y$

5: $\delta_{y/x} \leftarrow \begin{cases} 0 & \delta_x = 0 \\ \delta_y/\delta_x & \text{sonst} \end{cases}$ $\quad$ $\delta_{x/y} \leftarrow \begin{cases} 0 & \delta_y = 0 \\ \delta_x/\delta_y & \text{sonst} \end{cases}$

6: $\Delta_x \leftarrow \begin{cases} (1, \delta_{y/x})^T & \delta_x \geq 0 \\ -(1, \delta_{y/x})^T & \text{sonst} \end{cases}$ $\quad$ $\Delta_y \leftarrow \begin{cases} (\delta_{x/y}, 1)^T & \delta_y \geq 0 \\ -(\delta_{x/y}, 1)^T & \text{sonst} \end{cases}$

7: $X = \frac{1}{2}\Delta_x$

8: $Y = \frac{1}{2}\Delta_y$

9: **if** $\delta_x < \delta_y$ **then**

10: $\quad$ **for** $i \leftarrow 1, \ldots, anzSP$ **do**

11: $\quad\quad$ **if** $|X_x - Y_x| < \varepsilon$ **then**

12: $\quad\quad\quad$ $S^{[i]} \leftarrow A + (X_x, Y_y)^T$

13: $\quad\quad\quad$ $Z^{[i]} \leftarrow A + \text{runden}((X_x, Y_y)^T)$

14: $\quad\quad\quad$ $X \leftarrow X + \Delta_x$

15: $\quad\quad\quad$ $Y \leftarrow Y + \Delta_y$

16: $\quad\quad\quad$ $anzSP \leftarrow anzSP - 1$

17: $\quad\quad$ **else if** $|X_x| < |Y_x|$ **then**

18: $\quad\quad\quad$ $S^{[i]} \leftarrow A + X$

19: $\quad\quad\quad$ $Z^{[i]} \leftarrow A + \text{runden}(X)$

20: $\quad\quad\quad$ $X \leftarrow X + \Delta_x$

21: $\quad\quad$ **else**

22: $\quad\quad\quad$ $S^{[i]} \leftarrow A + Y$

23: $\quad\quad\quad$ $Z^{[i]} \leftarrow A + \text{runden}(Y)$

24: $\quad\quad\quad$ $Y \leftarrow Y + \Delta_y$

25: $\quad\quad$ **end if**

26: $\quad$ **end for**

27: **else**

28: $\quad$ ... verfahre entsprechend, vergleiche aber y-Koordinaten (X_y und Y_y) ...

29: **end if**

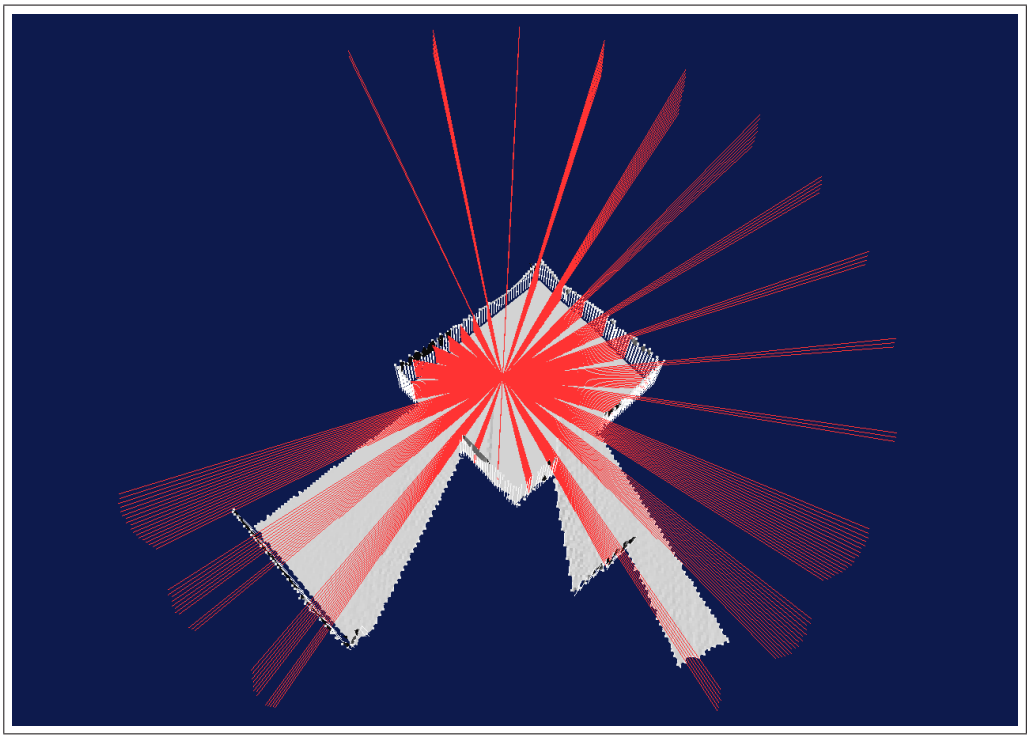


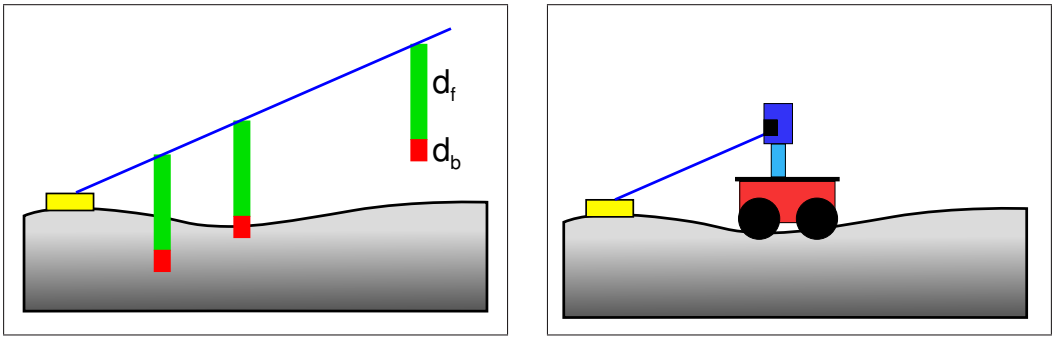
Abbildung 4.8: Raycasting in einer MLS-Karte einer teilweise explorierten Umgebung.

4.5 Generierung von Zielpunkten

In diesem Kapitel wird die Menge der potenziellen Zielpunkte bestimmt. Im folgenden Kapitel 4.6 beschreiben wir dann, wie wir die in dieser Menge enthaltenen Zielpunkte bewerten, um den besten Zielpunkt auszuwählen.

Die vorgestellte Explorationsstrategie folgt dem populären grenzzellenbasierten Ansatz (*frontier-based exploration*) [48]. Dabei werden Grenzen zwischen unbekannten und traversierbaren, bekannten Gebieten bestimmt und der Roboter an eine dieser Grenzen geschickt, um eine Messung durchzuführen. Die Messung wird teilweise in das unbekannte Gebiet reichen und so die Grenze zum unbekannten Gebiet weiter verschieben. Nach und nach exploriert der Roboter so die komplette Umgebung.

Nachdem bereits die Exploriertheit eines Patches definiert wurde, können wir nun leicht die Bereiche zu unbekanntem Gebiet bestimmen: Ein Grenz-Patch ist ein nicht explorierter Patch, der mindestens einen explorierten Nachbar-Patch besitzt. Zusammenhängende Grenz-Patches fassen wir zu Grenzlinien zusammen und verwerfen alle Grenz-Patches, die Teil einer zu kurzen Grenzlinie sind. Dadurch vermeiden wir die Exploration von Bereichen, die keinen lohnenden Informationsgewinn versprechen. Zudem wird eine verlässlichere Terminierung des Explorationsprozesses erreicht. Da



(a) (b)

Abbildung 4.9: Bestimmung eines Zielpunkts von dem aus ein Grenz-Patch (gelbes Rechteck) sichtbar ist. Entlang des simulierten Laserstrahls wird die Belegtheit vertikaler Intervalle geprüft. Grüne Intervalle entsprechen der Laserhöhe d_f und müssen frei sein. Die roten Intervalle der Höhe d_b müssen durch einen Patch belegt sein, um dem Roboter einen Standpunkt zu bieten.

Patches dreidimensionale Strukturen modellieren, sind auch die Grenzlinien dreidimensional, d. h. sie können beispielsweise entlang der Kante einer Rampe verlaufen.

Da Grenz-Patches nicht explorierte Patches sind, werden die meisten dieser Patches weniger als acht Nachbarn besitzen und somit nicht traversierbar sein, da sie am Rand eines Abgrunds liegen könnten. Der Roboter kann also nicht direkt zu einem Grenz-Patch fahren, sondern muss in dessen Umgebung einen traversierbaren Patch finden, der dann einen potenziellen Zielpunkt für die nächste Messung darstellt. Zur Bestimmung dieser potenziellen Zielpunkte verwenden wir die im obigen Abschnitt vorgestellte 3D-Raycasting-Methode. Die Idee ist, Laserstrahlen ausgehend von einem Grenz-Patch rückwärts zu möglichen Laserposition zu verfolgen, von denen der Grenz-Patch sichtbar wäre (Abb. 4.9). Für jeden Grenz-Patch werden Laserstrahlen in unterschiedlichen horizontalen und vertikalen Winkeln ausgesandt. Dabei schränken wir die maximale Länge des Laserstrahls auf wenige Meter ein, um gezielt Zielpunkte in der Nähe des Grenz-Patches zu generieren. Sei nun h die mittlere Laserhöhe eines Laserstrahls innerhalb einer der traversierten Zellen, d_f die Höhe, die der Laser über dem Boden angebracht ist und d_b ein vertikaler Toleranzbereich. Dann wird überprüft, ob das vertikale Intervall $[h - d_f, h]$ frei ist und ob das Intervall $[h - (d_b + d_f), h - d_f]$ belegt ist. Ist dies der Fall, wird geprüft, ob der im zweiten Intervall gefundene Patch vom Roboter aus erreichbar ist. Dies ist in konstanter Zeit möglich, da wir die durch den Dijkstra-Algorithmus bestimmte Distanz zu einem Patch in diesem Patch speichern. Ein Patch ist erreichbar, falls seine Distanz nicht unendlich ist.

Da in der Regel die Anzahl der Grenz-Patches deutlich geringer ist als die Menge

Algorithmus 2 Bestimmung von potenziellen Zielpunkten.

Eingabe: Karte m, Grenz-Patches G, Laserhöhe d_f über dem Boden, Toleranzbereich d_b, maximale Laserreichweite $r_{\max}$

Ausgabe: potenzielle Zielpunkte Z

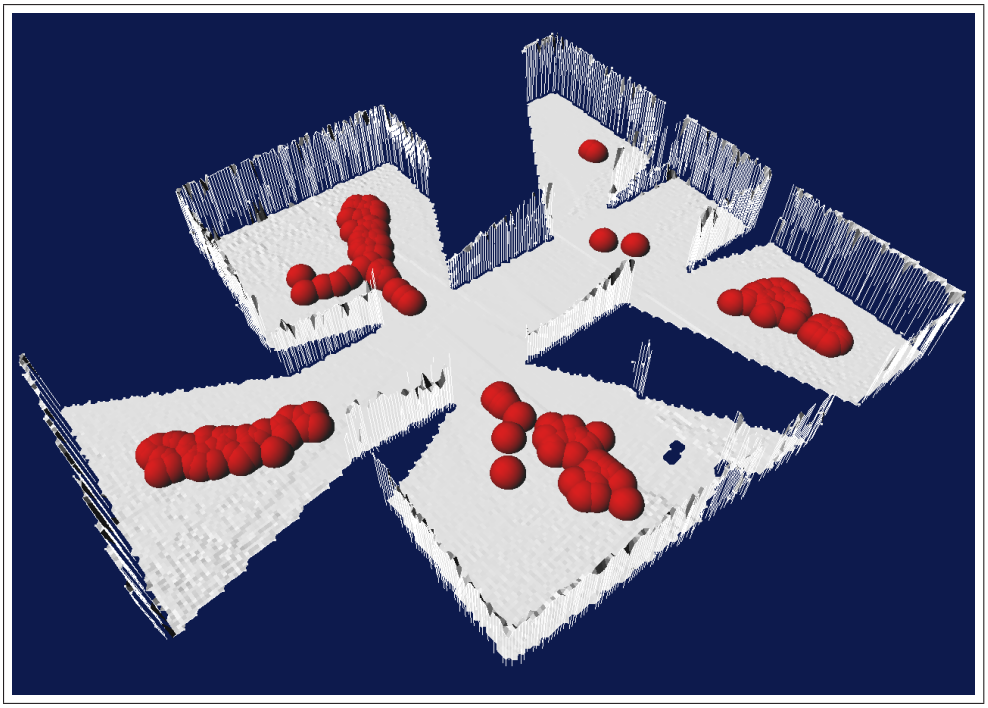
```
1:  Z ← ∅
2:  for all g ∈ G do
3:    for all horizontale Winkel α_h und vertikale Winkel α_v do
4:      // bestimme traversierte Zellen und mittlere Laserhöhe in den Zellen:
5:      (⟨c_1, ..., c_n⟩, ⟨h_1, ..., h_n⟩) ← raycast(m, g, d_f, α_h, α_v, r_max)
6:      for all i ∈ [1, ..., n] do
7:        if frei(c_i, [h_i − d_f, h_i]) ∧ belegt(c_i, [h_i − (d_b + d_f), h_i − d_f]) then
8:          p ← patchInIntervall(c_i, [h_i − (d_b + d_f), h_i − d_f])
9:          if erreichbar(p) then
10:           Z ← Z ∪ p
11:         end if
12:       end if
13:     end for
14:   end for
15: end for
```

der erreichbaren Patches, ist die Methode Laserstrahlen rückwärts zu verfolgen effizienter als für jeden erreichbaren Patch festzustellen, ob ein Grenz-Patch sichtbar ist.

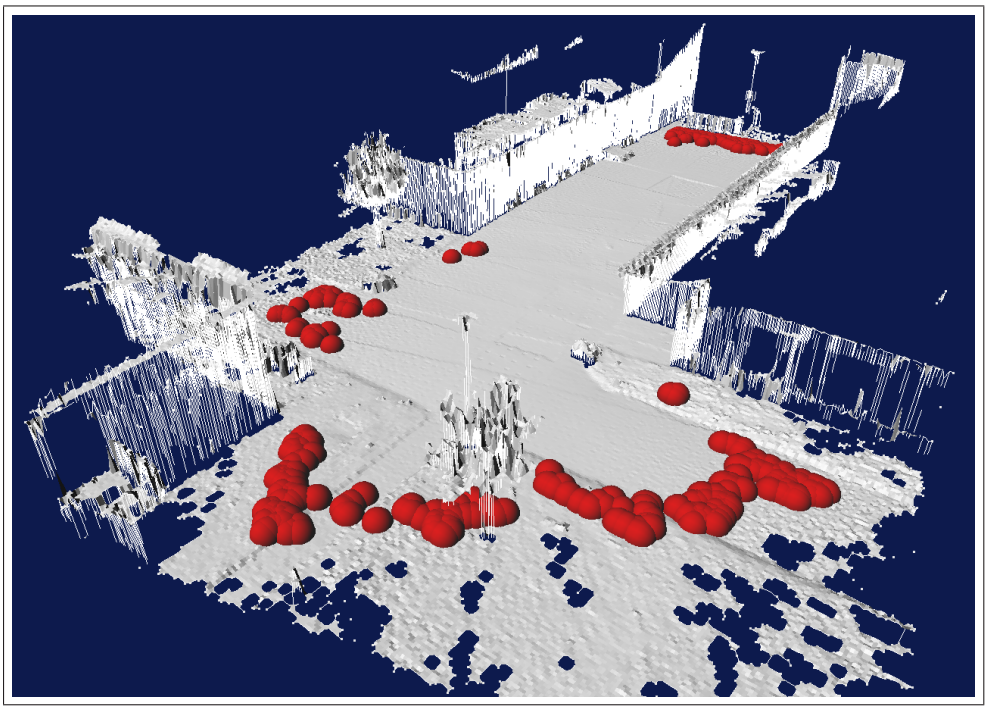
Die Anzahl der so bestimmten Zielpunkte kann schon in kleinen Umgebungen in die Tausende gehen. Die Bewertung der Zielpunkte ist jedoch relativ rechenaufwendig, da wir hierbei wiederum die Raycasting-Methode einsetzen werden, um der dreidimensionalen Umgebungsstruktur Rechnung zu tragen. Wir werden deshalb aus der Menge der Zielpunkte zufällig eine fest vorgegebene Anzahl auswählen. Während ausgiebiger Experimente haben sich 250 bis 500 Zielpunkte als sinnvoller Kompromiss zwischen Rechenzeit und möglichst vollständiger Bewertung aller Zielpunkte erwiesen.

4.6 Bewertung der Zielpunkte

Der nächste anzufahrende Zielpunkt sollte idealerweise nicht allzu weit entfernt sein und möglichst viel neue Informationen liefern. Wir werden den Nutzen $u(p)$ eines Zielpunkts p daher durch den erwarteten Informationsgewinn $E\{I(p)\}$, sowie die Fahrtkosten $t(p)$ bestimmen. Diese Faktoren wirken meist antagonistisch auf die



(a) Zufällig ausgewählte Teilmenge der generierten Zielpunkte in einer simulierten Umgebung.



(b) Zufällig ausgewählte Teilmenge der generierte Zielpunkte in einer realen Außenumgebung.

Abbildung 4.10: Generierte Zielpunkte (rote Halbkugeln) in verschiedenen Umgebungen.

Auswahl eines Zielpunkts. Während unter Berücksichtigung der Fahrtkosten nahe Zielpunkte bevorzugt werden, sind es meist die entfernt gelegenen Zielpunkte, die an der Grenze zum nicht explorierten Gebiet liegen und damit einen hohen Informationsgewinn erwarten lassen. Um beide Faktoren gegeneinander abzuwägen, bilden wir die gewichtete Summe (vgl. [39])

$$u(p) = \alpha \frac{E\{I(p)\}}{\max_x E\{I(x)\}} + (1-\alpha)\frac{\max_x t(x) - t(p)}{\max_x t(x)}. \tag{4.19}$$

Da es unintuitiv wäre zwei verschiedene Größen (Information und Fahrtkosten) zu subtrahieren, werden in (4.19) der relative Informationsgewinn und die relativen Fahrtkosten verwendet, um die Dimensionslosigkeit der Größen zu erreichen. Die relativen Werte erhalten wir durch die Normalisierung über den jeweiligen Maximalwert. Über die Konstante $\alpha \in [0,1]$ kann die Explorationsstrategie variiert werden. Bei geringem α werden nahe Zielpunkte bevorzugt, während für hohe Werte von α die Fahrtkosten weniger schwer wiegen und die bevorzugten Zielpunkte jene sind, die am meisten Informationsgewinn versprechen. In den folgenden zwei Abschnitten werden wir nun darauf eingehen, wie $t(p)$ und $E\{I(p)\}$ bestimmt werden.

4.6.1 Fahrtkosten

Um die Fahrtkosten $t(p)$ eines Zielpunktes p zu bestimmen, wird mittels des Dijkstra-Algorithmus simultan der kürzeste Pfad zu allen Patches berechnet. Die verwendeten Kosten (vgl. S. 74, Alg. 3, Zeilen 12 und 13)

$$c(p,p') = d(p,p') + \nu\left(1 - \tau(p')\right) \tag{4.20}$$

von einem Patch p zu einem traversierbaren Nachbar-Patch $p' \in N(p)$ berücksichtigen dabei nicht nur die euklidische Distanz

$$d(p,p') = \|p - p'\|_2 = \sqrt{\sum_{i=1}^{3}(p_i - p'_i)^2}, \tag{4.21}$$

sondern auch die Traversierbarkeit $\tau(p')$ des Nachbar-Patches p'. Über die Konstante ν kann festgelegt werden, wie stark die Pfadkosten von der Traversierbarkeit beeinflusst sein sollen. Die Einbeziehung der Traversierbarkeit erlaubt es bei der Pfadplanung, und somit letztendlich auch bei der Auswahl von Zielpunkten, die Steigung und Unebenheiten zu berücksichtigen, die bis zum Zielpunkt überwunden werden müssen.

Abb. 4.11 illustriert die Fahrtkosten der erreichbaren Patches in einer MLS-Karte einer simulierten Umgebung. Der Roboter befindet sich in der Mitte der blauen Region.

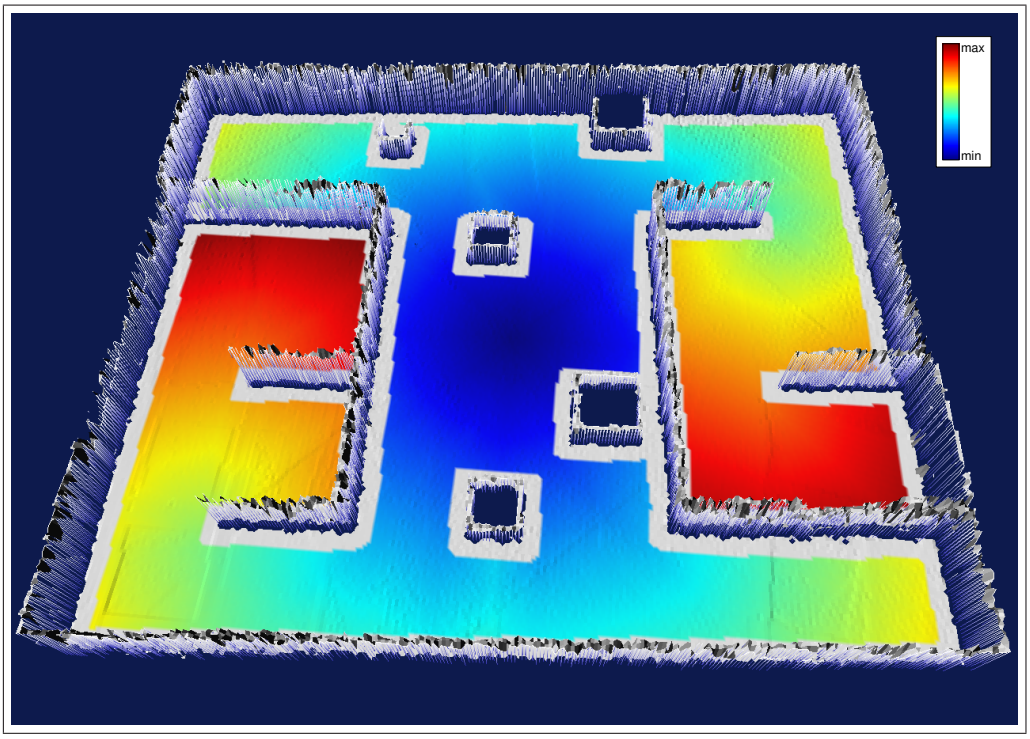


Abbildung 4.11: Illustration der Fahrtkosten in einer Karte einer simulierten Umgebung. Der Roboter befindet sich in der Mitte der blauen Region.

4.6.2 Informationsgewinn

Der Informationsgewinn $I(p)$ eines Zielpunktes p wird über den erwarteten Rückgang der Unsicherheit in der Karte ausgedrückt, der sich durch eine Messung an diesem Zielpunkt ergeben würde. Als quantitatives Maß der Unsicherheit werden wir die Entropie verwenden. Der Informationsgewinn wird sowohl die Unsicherheit in den bereits in der Karte enthaltenen Patches berücksichtigen, als auch die zu erwartenden neuen Patches. Der tatsächliche Informationsgewinn einer Messung an einem Zielpunkt kann natürlich erst nach der Durchführung der Messung bestimmt werden. Wir werden daher Messungen simulieren und vom *erwarteten* Informationsgewinn $E\{I(p)\}$ sprechen.

Um eine Messung zu simulieren, verfolgen wir mit der in Kapitel 4.4 vorgestellten Raycasting-Methode virtuelle Laserstrahlen. Durch die Angabe des Zielpunktes p und der Laserhöhe h_l über dem Boden ist der Startpunkt dieser Laserstrahlen festgelegt. Die Richtung der Laserstrahlen wird durch die horizontalen und vertikalen Winkel bestimmt. Die Wahl dieser Winkel orientiert sich an der Art und Weise, wie der Roboter einen tatsächlichen 3D-Scan durchführt. Während eines solchen 3D-Scans

werden vier horizontale Winkel im Abstand von 90° angefahren. Für jeden horizontalen Winkel wird der Laserscanner gekippt und überstreicht dabei einen vertikalen Winkelbereich von 40°. Laserdaten werden nur während des vertikalen Kippens aufgenommen, nicht während des Anfahrens der horizontalen Winkel. Wir approximieren diesen Ablauf eines 3D-Scans dadurch, dass wir für verschiedene horizontale Winkel, jeweils mehrere vertikale Winkel im Bereich von ±20° wählen. Dabei berücksichtigen wir die bereits während der Traversierbarkeitsanalyse berechnete Neigung des Patches von dem aus die simulierte Messung vorgenommen werden soll. Mit der maximalen Laserreichweite $r_{\max}$ sind dann alle notwendigen Parameter festgelegt.

Sei nun $L = \langle l_1, \ldots, l_m \rangle$ eine äquidistante Diskretisierung der maximalen Laserreichweite $l_m = r_{\max}$. Wenn der Endpunkt des simulierten Laserstrahls auf ein bekanntes Hindernis in der Karte trifft, können wir die Sequenz L in drei Teilsequenzen

$$L^f = \langle l_1, \ldots, l_{k-1} \rangle \tag{4.22}$$
$$L^k = \langle l_k \rangle \tag{4.23}$$
$$L^n = \langle l_{k+1}, \ldots, l_m \rangle \tag{4.24}$$

aufteilen. L^f entspricht den hindernisfrei durchlaufenen Strecken, L^k entspricht der ermittelten (und diskretisierten) Strecke l_k bis zum Hindernis und L^n enthält schließlich alle nicht durchlaufenen Strecken des Laserstrahls. Für den Fall, dass der simulierte Laserstrahl nicht auf ein Hindernis trifft, erhalten wir die entsprechende Aufteilung

$$L^f = L = \langle l_1, \ldots, l_m \rangle \tag{4.25}$$
$$L^k = \langle\rangle \tag{4.26}$$
$$L^n = \langle\rangle, \tag{4.27}$$

wobei $\langle\rangle$ die leere Sequenz ist. Für jede der durchlaufenen Strecken $l \in L^f \cup L^k$ besteht eine Wahrscheinlichkeit $p(l)$, dass der Laserstrahl bei einer tatsächlichen Messung nach der Strecke l auf ein Hindernis stößt. Falls der Endpunkt l des Laserstrahl in L^f liegt, entspricht dies der Entdeckung eines unbekannten Patches, dessen Entdeckung für uns einen Informationsgewinn $I_f(l)$ bedeutet. Falls der Endpunkt $l \in L^k$ durch ein bekanntes Hindernis festgelegt ist, werden wir ebenfalls einen Informationsgewinn $I_k(l)$ erwarten, der jedoch nicht die Entdeckung eines neuen Patches widerspiegelt, sondern eine durch die Messung bedingte Änderung in der Unsicherheit der Positionsschätzung des bereits bekannten Patches. Den erwartete

Informationsgewinn $E\{I(r)\}$ eines Laserstrahls r können wir nun durch

$$E\{I(r)\} = \sum_{l \in L} p(l) \cdot I(l) \tag{4.28}$$

$$= \sum_{l \in L^f} p(l) \cdot I_f(l) + \sum_{l \in L^k} p(l) \cdot I_k(l) \tag{4.29}$$

berechnen. Wir werden nun näher darauf eingehen, wie wir die Informationsgewinne $I_f(l)$ und $I_k(l)$, sowie die Wahrscheinlichkeiten $p(l)$ bestimmen.

Für den Fall, dass der Laserstahl ein bekannten Patch p_k in einer Distanz $l \in L^k$ trifft, wird diese Messung mit einer Unsicherheit $\sigma^2 \propto l$ temporär in die zugehörige Zelle eingefügt. Die Zelle wird dann aktualisiert, wie dies auch bei einer realen Messung der Fall wäre. Dabei wird sich Mittelpunkt und Varianz (und damit die Entropie) des Patches p_k ändern. Sei nun $H_k(p_k)$ die Entropie des Patches und $H_k(p_k \mid m)$ die Entropie nach der temporären Aktualisierung des Patches mit der simulierten Messung. Dann ist der Informationsgewinn für die Distanz l definiert als

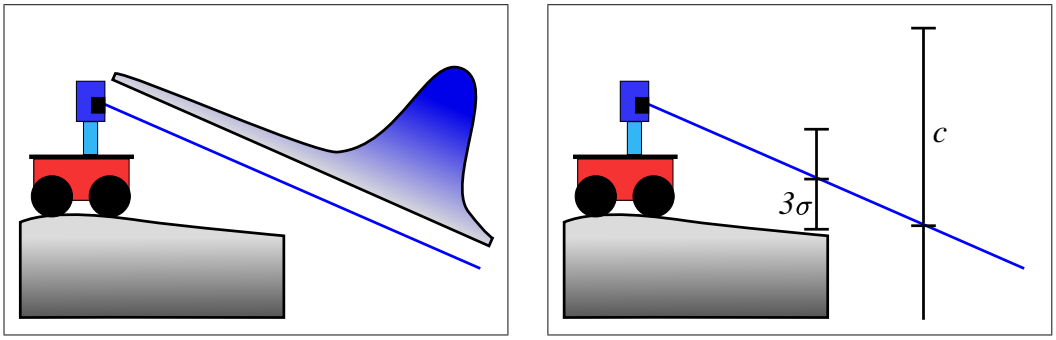
$$I_k(l) = H_k(p_k) - H_k(p_k \mid m) \qquad l \in L^k. \tag{4.30}$$

Für den Fall, dass wir den Informationsgewinn für die Entdeckung eines neuen Patch p_f bestimmen müssen, kennen wir die Unsicherheit σ^2 und damit die Entropie $H_f(p_f)$ vor der Integration der Messung nicht. Die Unsicherheit, die ja eine Unsicherheit ob der vertikalen Position des Patches widerspiegelt, ist jedoch durch eventuell darüber oder darunter liegende Patches begrenzt. Wir werden daher folgende Heuristik verwenden: Die Varianz σ^2 des Patches p_f vor der Beobachtung soll so gewählt werden, dass der vertikale Abstand zum nächsten Patch gerade 3σ entspricht (vgl. Abb. 4.12b). Können wir keinen Abstand bestimmen, weil die Zelle leer ist, nehmen wir für die Varianz eine Konstante c an, deren Wert wir nach Erprobung in verschiedenen Experimente festgelegt haben. Die Entropie $H_f(p_f \mid m)$, die der Patch nach der Messung besitzt, ist leichter zu bestimmen, da neue Patches mit einer Varianz proportional zur gemessenen Distanz in die Karte eingetragen werden. Mit den so festgelegten Entropiewerten $H_f(p_f)$ und $H_f(p_f \mid m)$ können wir auch den Informationsgewinn

$$I_f(l) = H_f(p_f) - H_f(p_f \mid m) \qquad l \in L^f \tag{4.31}$$

für die Entdeckung eines neuen Patches p_f berechnen.

Um die Wahrscheinlichkeiten $p(l)$ für $l \in L$ festzulegen, wurden Statistiken durch simulierte Messungen in einer MLS-Karte des Campus der Universität erstellt. An jeder erreichbaren Position innerhalb der Karte wurde eine Messung simuliert und in Abhängigkeit des vertikalen Winkels (Elevation) α_v eines Laserstrahls die Distanz



(a) (b)

Abbildung 4.12: (a) Illustration der Wahrscheinlichkeit, nach einer gewissen Strecke entlang eines Laserstrahls auf ein Hindernis zu stoßen. **(b)** Bestimmung der Varianz für neu entdeckte Patches.

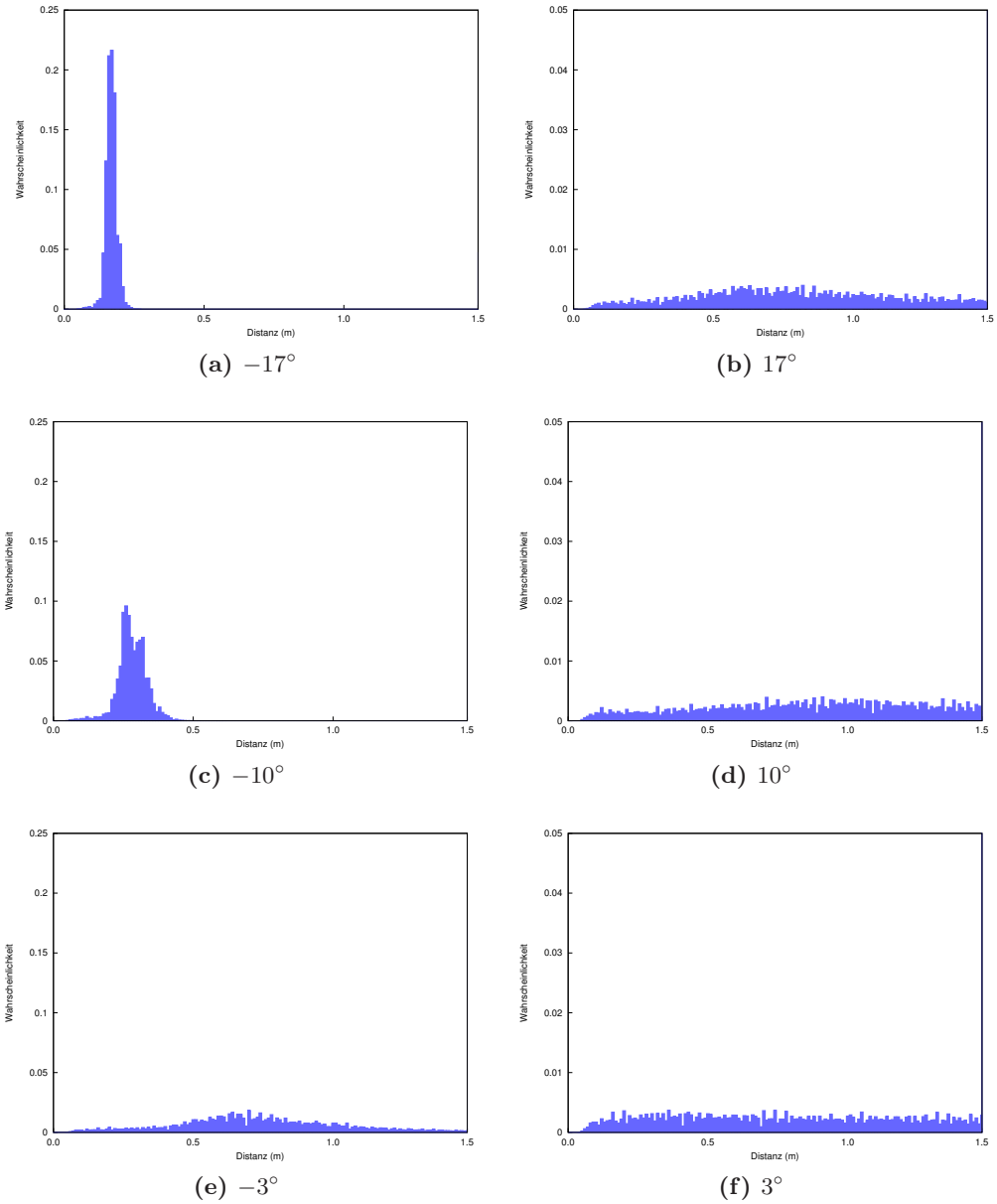
zum gemessenen Endpunkt geloggt. Daraus lässt sich dann eine Wahrscheinlichkeitsverteilung $p_s(\cdot \mid \alpha_v)$ über die Distanz des Endpunktes berechnen (Abb. 4.13). Die Verteilung macht eine Aussage über die Lage des Endpunkts eines Laserstrahls, nicht über die Belegtheit einer Zelle nach einer Distanz l. Die Wahrscheinlichkeit $p(l)$ nach der Distanz $l \in L^f$ einen neuen Patch zu entdecken, ist damit $p_s(l \mid \alpha_v)$, wobei α_v wieder der vertikale Winkel des Laserstrahls ist. Ist jedoch $l \in L^k$ die Distanz des simulierten Laserstrahls bis zu einem bekannten Hindernis in der Karte, akkumuliert die Wahrscheinlichkeit $p(l)$ die Wahrscheinlichkeiten der Endpunkte L^n hinter diesem Hindernis.

$$p(l) = \begin{cases} p_s(l \mid \alpha_v) & l \in L^f \\ \sum\limits_{l_i \in L^k \cup L^n} p_s(l_i \mid \alpha_v) & l \in L^k \\ 0 & l \in L^n \end{cases} \tag{4.32}$$

Schließlich berechnet sich der erwartete Informationsgewinn $E\{I(p)\}$ eines Zielpunktes p als die Summe der erwarteten Informationsgewinne aller ausgesandter Laserstrahlen $r \in R$:

$$E\{I(p)\} = \sum_{r \in R} E\{I(r)\}. \tag{4.33}$$

Zusammenfassend kann festgehalten werden, dass der Informationsgewinn den Rückgang der Unsicherheit in der bereits bekannten Karte berücksichtigt, sowie den Informationsgewinn durch zu erwartende neue Patches. Grundsätzlich sind entfernte Messungen unsicherer als nahe Messungen. Bei der temporären Aktualisierung bereits bekannter Patches geht dies durch die distanzabhängige Varianz der eingefügten

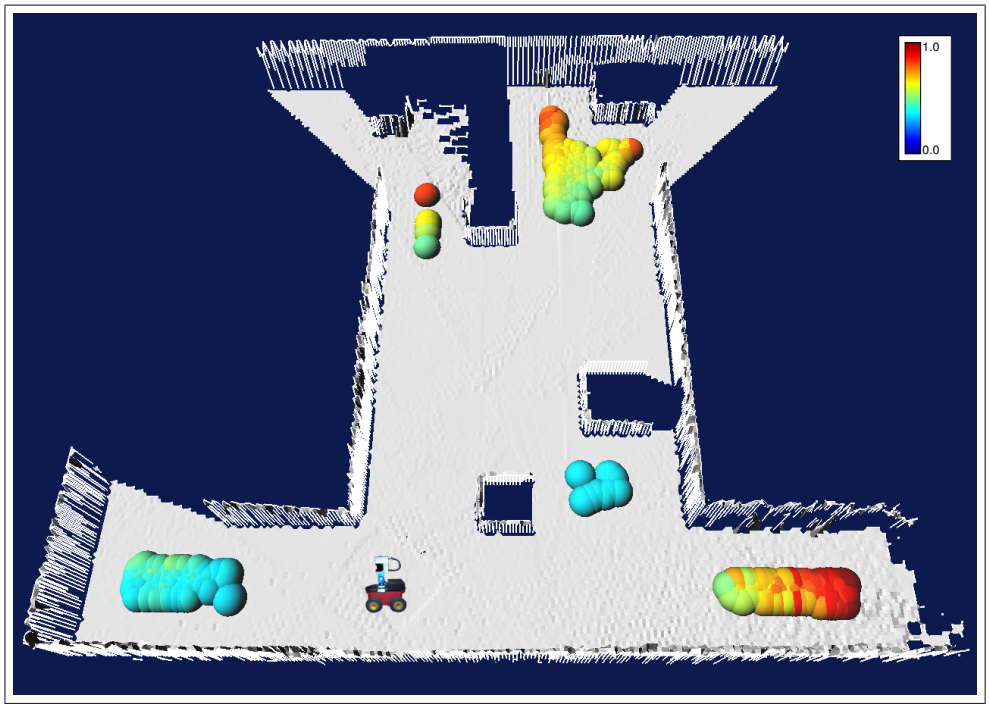


(a) $-17°$ **(b)** $17°$

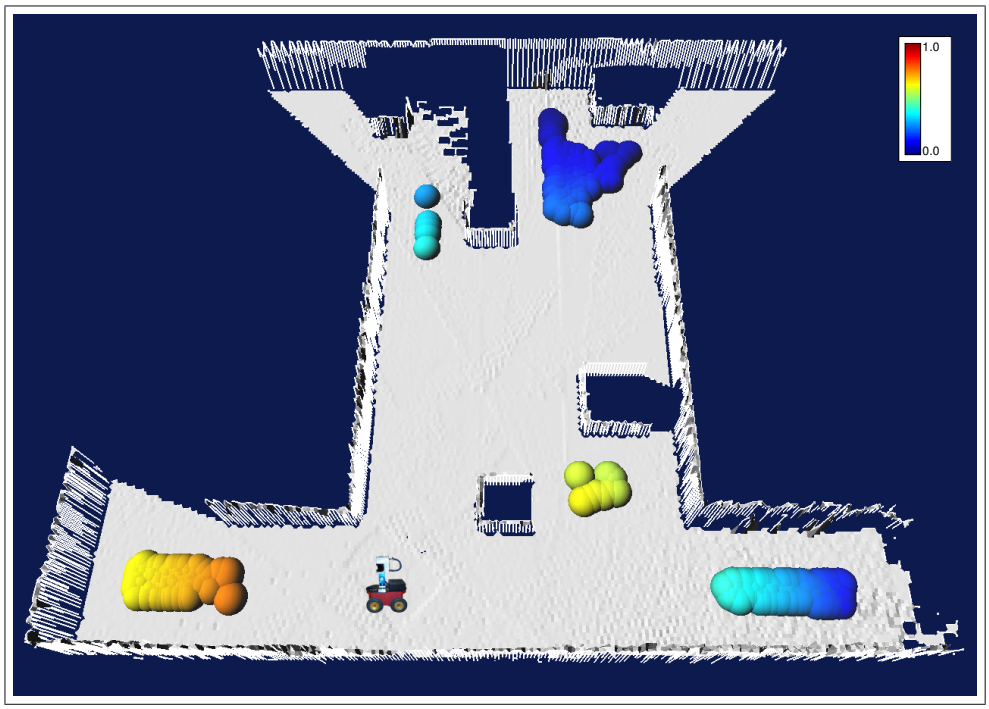
(c) $-10°$ **(d)** $10°$

(e) $-3°$ **(f)** $3°$

Abbildung 4.13: Wahrscheinlichkeitsverteilungen über die Distanz zu einem getroffenen Hindernis für verschiedene vertikale Winkel (positive Winkel zeigen nach oben, negative nach unten). Bei den nach oben gerichteten Strahlen ist eine uniforme Verteilung zu erkennen, die seltene und unvorhersehbare Ereignisse wie das Messen einer Hauswand in verschiedenen Distanzen widerspiegeln. Bei den nach unten gerichteten Strahlen ist die dominierende Komponente der gemessene Abstand zum Boden, der einer Gaußverteilung folgt und mit zunehmenden Abstand (geringere Neigung des Laserstrahls) an Varianz gewinnt. Der Übersichtlichkeit wegen, wurde die Wahrscheinlichkeit für das Ereignis auf kein Hindernis zu stoßen, nicht eingetragen. Zu beachten ist, dass positive und negative Winkel in unterschiedlichen Skalen dargestellt sind.



(a) Relativer, erwarteter Informationsgewinn.



(b) Relative Fahrtkosten.

Abbildung 4.14: Potenzielle Zielpunkte und deren Bewertung nach relativen, erwarteten Informationsgewinn (oben) und relativen Fahrtkosten (unten).

Messung mit ein, die durch die eindimensionale Kalman-Filterung zu einem geringeren Entropieunterschied und damit geringeren Informationsgewinn führt. Bei den zu erwartenden neuen Patches berücksichtigen wir dies durch die distanzabhängigen Entropie $H_f(p \mid m)$, die proportional zur Distanz zunimmt.

In Abb. 4.14 ist die Bewertung von potenziellen Zielpunkten nach Informationsgewinn und nach Fahrtkosten dargestellt. In Abb. 4.14a ist zu erkennen, dass die Zielpunkte mit hohem erwarteten Informationsgewinn meist am Rand der Karte liegen. Im gleichen Bild sind unten rechts auch Diskretisierungseffekte zu erkennen, die der diskreten Wahl der horizontalen und vertikalen Winkel der Laserstrahlen während einer simulierten Messung geschuldet sind. Die Wand wurde an der Stelle bisher nur aus großer Entfernung vermessen und weist in unregelmäßigen Abständen Löcher auf, sodass schon kleine Unterschiede in der Position des Zielpunkte in dieser Region einen großen Unterschied in der Anzahl der Strahlen bewirkt, die bei einer simulierten Messung durch die Wand in freies Gebiet reichen.

4.7 Überlappung und Zwischenziele

Die durch einen 3D-Scan gewonnene lokale Karte muss in die globale Karte eingefügt werden. Dazu müssen die egozentrischen Koordinaten der Patches in der lokalen Karte in das allozentrische Koordinatensystem der globalen Karte überführt werden. Diese Koordinatentransformation erfordert die Kenntnis der Position an der die lokale Karte aufgenommen wurde. Diese Information ist zwar durch die Selbstlokalisierung des Roboters gegeben, sie kann jedoch aufgrund ungenauer Odometriedaten und verrauschter Sensorendaten nie als fehlerfrei angesehen werden. Die Schätzung der Position der lokalen Karte wird deshalb durch ein Scan-Matching-Verfahren verbessert. Für MLS-Karten wird der ICP-Algorithmus (*iterative closest point*) verwendet. Das Verfahren versucht die lokale Karte in die vorherige lokale Karte einzupassen. Dafür ist es notwendig, dass die lokalen Karten eine ausreichende Überlappung besitzen, da nur so sichergestellt ist, dass während des Scan-Matchings ausreichend Datenpunkte zwischen den Karten assoziiert werden können. Der Roboter kann daher nicht direkt zum Zielpunkt fahren, da die Überlappung der an diesem Zielpunkt aufgenommenen und der letzten lokalen Karte zu gering oder im Extremfall gar nicht vorhanden sein könnte. Dies ist beispielsweise der Fall, wenn zwei aufeinanderfolgende 3D-Scans in unterschiedlichen Zimmern durchgeführt wurden, oder der Roboter zwischen zwei Messungen um die Ecke eines Hauses fährt. Wir werden deshalb auf dem Weg zum Zielpunkt Zwischenpunkte bestimmen, an denen zusätzliche 3D-Scans durchgeführt werden sollen.

Sei nun ein Zielpunkt p_n ausgewählt und ein Pfad $\langle p_1, \ldots, p_n \rangle$ zu diesem Ziel-

punkt geplant. Wir werden für jeden Patch p_i des Pfades abschätzen, wie groß die Überlappung

$$o(p_i) = \left| m_t^L \cap m_{t+1}^L \right| \tag{4.34}$$

einer an dieser Stelle aufgenommenen lokalen Karte m_{t+1}^L mit der zuletzt aufgenommenen lokalen Karte m_t^L wäre. Wir kennen die Karte m_{t+1}^L noch nicht, sind jedoch auch nur an der Schnittmenge dieser Karte mit der vorherigen lokalen Karte interessiert. Wir müssen daher nur bestimmen, welche der bereits bekannten Patches auch in der neuen lokalen Karte enthalten wären und müssen nicht über neu hinzugekommene Patches nachdenken.[2] Die Menge dieser Patches werden wir durch eine in der globalen Karte m_t^G durchgeführten Raycasting-Operation abschätzen. Die geschätzte Überlappung

$$\hat{o}(p_i) = \frac{r(p_i, m_t^L)}{r} \tag{4.35}$$

für den Patch p_i ist über den Anteil der Strahlen $r(p_i, m_t^L)$, die ein Patch in m_t^L treffen, zur Gesamtzahl r aller während der Raycasting-Operation an der Position von p_i ausgesandten Strahlen definiert.

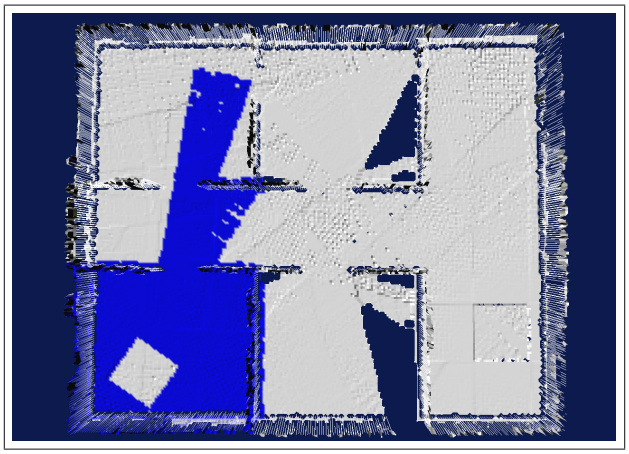
Wir sind daran interessiert, dass der Roboter dem Pfad möglichst weit folgt, bevor er einen zusätzlichen 3D-Scan durchführt. Andererseits soll die Überlappung einen gewissen Grenzwert ϑ_o nicht unterschreiten, um ein erfolgreiches Scan-Matching zu gewährleisten. Im Regelfall nimmt die Überlappung mit zunehmender Distanz jedoch ab. Wir definieren den anzufahrenden Zwischenpunkt daher als denjenigen Patch $p_z \in \langle p_1, \ldots, p_n \rangle$ mit dem höchsten Index

$$z = \arg\max_i \{ i \mid \hat{o}(p_i) > \vartheta_o \wedge (1 \leq i \leq n) \} \tag{4.36}$$

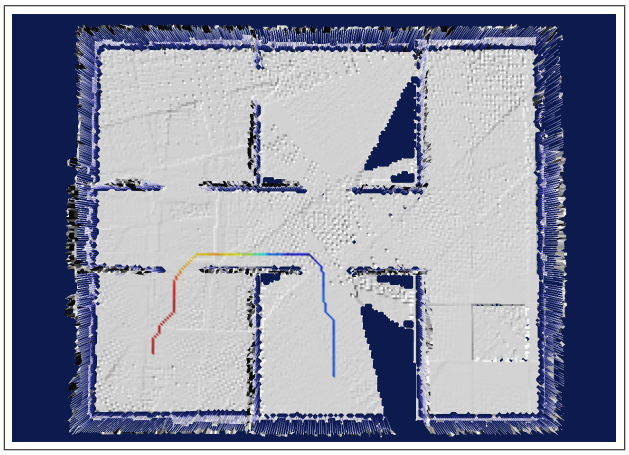
von allen Patches des Pfads, dessen Überlappung den geforderten Grenzwert ϑ_o überschreitet. Im unwahrscheinlichen Fall, dass jeder Patch des Pfades den Grenzwert unterschreitet, werden wir eine Mindestdistanz fahren, um sicherzustellen, dass der Roboter das Ziel letztendlich doch erreichen wird. Ebenso kann es sinnvoll sein, dass der maximale Index nach oben beschränkt wird.

Abb. 4.15 verdeutlicht die Wahl eines Zwischenpunktes. Fährt der Roboter durch eine Tür, fällt die Überlappung hinter der Tür drastisch ab und es wird deshalb ein Zwischenpunkt nahe dem Türdurchgang gewählt. Bei der Messung innerhalb des

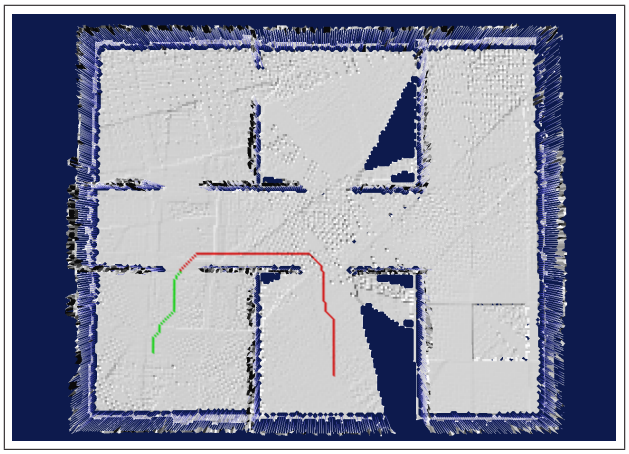
[2] Es können jedoch neu hinzugekommene Patches die Überlappung dadurch beeinflussen, dass sie Patches aus der vorherigen Karte verdecken. Diese Möglichkeit klammern wir hier bewusst aus, da solche Ereignisse kaum vorhergesagt werden können und die Abschätzung der Überlappung unverhältnismäßig erschweren würden.



(a) Lage der lokalen Karte innerhalb der globalen Karte.



(b) Überlappung eines jeden einzelnen Pfadpunktes.



(c) Erlaubte (grüne) und nicht erlaubte (rote) Pfadpunkte.

Abbildung 4.15: Wahl eines geeigneten Zwischenziels auf dem Pfad.

Zimmers liegen die gemessenen Patches fast ausschließlich innerhalb des Zimmers. Bei der Messung am Zwischenpunkt nahe dem Türdurchgang liegen die gemessenen Patches etwa hälftig innerhalb und außerhalb des Zimmers. Dies stellt einerseits eine ausreichende Überlappung mit der lokalen Karte der Messung innerhalb des Zimmers sicher, während andererseits für eine nachfolgende Messung auf dem Korridor bereits genügend Patches außerhalb des Zimmers liegen um auch hierfür eine Überlappung zu garantieren.

4.8 Terminierung

Die Exploration der Umgebung endet, sobald die ermittelte Menge der Zielpunkte leer ist, oder der erwartete Informationsgewinn aller in ihr enthaltenen Zielpunkte einen Grenzwert unterschreitet.

Damit wurden alle Konzepte der Explorationsstrategie vorgestellt. Im folgenden Kapitel gehen wir kurz auf Implementierungs-Details ein und stellen im darauffolgenden Kapitel Ergebnisse der durchgeführten Experimente vor.

Kapitel 5

Implementierungs-Details

Die Explorationsstrategie und die Navigation wurden als Module in das noch unveröffentlichte System „Morpheus“ integriert, welches Funktionen für die dreidimensionale Kartierung mit MLS-Karten bereitstellt. Wir geben zunächst einen Überblick des Zusammenspiels der relevanten Module in Morpheus und gehen dann auf die Integration der neu hinzugefügten Module ein.

5.1 Morpheus

Morpheus stellt verschiedene Module für die Kartierung von dreidimensionalen Umgebungen bereit. Einen Überblick des Zusammenspiels der Module wird in Abb. 5.1 gegeben. Die Kommunikation zwischen den Modulen findet über die IPC-Architektur von Carmen [1] statt. Dadurch können rechenintensive Module auf verschiedene Rechner verteilt werden. Bei den durchgeführten Experimenten war eine Aufteilung auf mehrere Rechner jedoch nicht notwendig.

Der verwendete Roboter ist mit einer Pan-Tilt-Einheit in einem Laser-Range-Scanner ausgestattet. Auf diese Einheiten kann über drei Module zugegriffen werden, die von der konkreten Hardware abstrahieren. Dadurch ist es auf einfache Weise möglich das System zunächst in einer Simulationsumgebung zu testen, und später ohne größere Modifikation direkt auf einem echten Roboter einzusetzen. Die verwendete Simulationsumgebung Gazebo [3] erlaubt es, eine dreidimensionale Welt aufzubauen, die über eine realistische Festkörperphysik verfügt. Genau wie bei einem echten Roboter werden Laser- und Odometriedaten erzeugt und Steuerkommandos ausgeführt.

Der Laser sendet 2D-Punktwolken an den Point-Cloud-Server. Über die Pan-Tilt-Einheit kann der Laser um einen horizontalen und vertikalen Winkel geschwenkt werden, was bei einem 3D-Scan für die Erstellung von 3D-Punktwolken genutzt wird, indem vom Point-Cloud-Server mehrere 2D-Punktwolken gesammelt werden

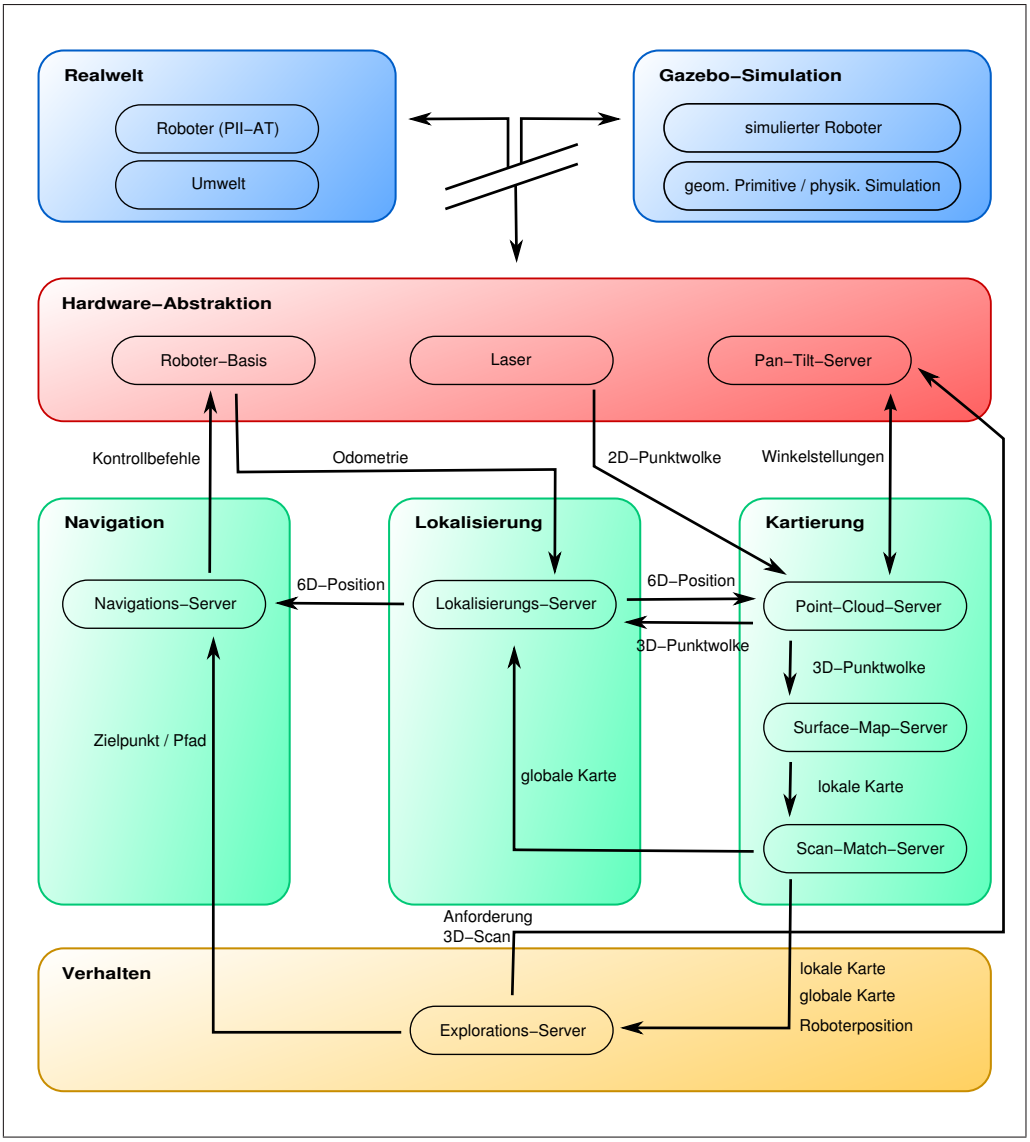


Abbildung 5.1: Überblick des Zusammenspiels verschiedener Module in Morpheus. Im Rahmen der Diplomarbeit wurden der Navigations- und Explorationsserver hinzugefügt.

und jeweils um den Winkel der Pan-Tilt-Einheit gedreht werden. Die Punktwolke ist relativ zur Pan-Tilt-Einheit gegeben und kann über die Positionsschätzung der Lokalisierung in globale Koordinaten überführt werden. Die in Morpheus implementierte Lokalisierung verwendet eine 6D-Monte-Carlo-Methode [27]. Die Lokalisierung erhält Odometrie-Daten von der Roboter-Basis und 3D-Punktwolken vom Point-Cloud-Server. Aus der 3D-Punktwolke eines 3D-Scans wird vom Surface-Map-Server eine lokale MLS-Karte erstellt. Die Position der lokale Karte wird schließlich vom Scan-Match-Server korrigiert, indem sie unter Verwendung des ICP-Verfahrens in die letzte lokale Karte eingepasst wird. Im Scan-Match-Server wird auch das Loop-Closing durchgeführt. Nachdem die Position der lokalen Karte festgelegt ist, wird eine neue globale Karte erzeugt.

5.2 Navigationsmodul

Die Navigation erhält vom Explorations-Server den Zielpunkt p_n zusammen mit einem Pfad $p = \langle p_1, \ldots, p_n \rangle$, der durch den Dijkstra-Algorithmus bestimmt wurde und alle Patches von der aktuellen Roboterposition bis zum Ziel enthält. Der detaillierte Pfad, wird mittels des Split-And-Merge-Verfahrens vereinfacht. Ziel dabei ist es, den Pfad durch Liniensegmente verbundene Wegpunkte $q = \langle q_1, \ldots, q_m \rangle$ anzunähern. Die Wegpunkte werden dann nach und nach abgefahren, bis das Ziel erreicht ist.

Über den Lokalisierungs-Server erhalten wir immer eine aktuelle Positionsschätzung $(x, y, z, \phi, \vartheta, \psi)^T$. Ist die Winkeldifferenz zwischen dem Gierwinkel ψ des Roboters und der Richtung zum nächsten Wegpunkt q_i klein genug, fährt der Roboter mit einer konstanten Geschwindigkeit los und korrigiert während der Fahrt stetig den Winkel. Ist die Winkeldifferenz zu groß, wird der Roboter angehalten und auf der Stelle gedreht, bis die Differenz wieder klein genug ist. Da der Pioneer 2-AT nicht auf der Stelle drehen kann, wird die Drehung auf der Stelle durch kurze Kurvensegmente erreicht, die abwechselnd vor- und rückwärts abgefahren werden, bis der gewünschte Winkel erreicht ist.

5.3 Explorationsmodul

Der Explorations-Server erhält vom Scan-Match-Server die globale und lokale Karte, die auch die Position des Roboters innerhalb der Karte enthält. Es wird über das in Kapitel 4 vorgestellte Verfahren ein Zielpunkt (oder ein Zwischenziel) bestimmt. Dem Navigations-Server wird dann die anzufahrende Position und der geplante Pfad gesendet. Sobald der Navigations-Server meldet, dass das Ziel erreicht wurde, wird ein 3D-Scan durchgeführt. Vom Scan-Match-Server erhalten wir kurz danach eine

neue lokale und globale Karte und es wird ein neues Ziel bestimmt. Falls der letzte anzufahrende Punkt nur ein Zwischenziel war, wird in der neuen globalen Karte ein neuer Pfad zum Ziel geplant und eventuell weitere Zwischenziele bestimmt. Kann das Ziel nicht mehr erreicht werden, wird ein neuer Zielpunkt (und eventuell ein Zwischenziel) bestimmt. Ergeben sich keine neuen Zielpunkte, ist die Exploration abgeschlossen.

Kapitel 6

Experimente

In diesem Kapitel wird die vorgestellte Explorationsstrategie evaluiert und die Fähigkeit demonstriert, verschiedenste Umgebungen explorieren zu können. Es werden sowohl Simulationsexperimente, als auch ein Experiment mit einem echten Roboter durchgeführt. Wir vermitteln zunächst einen Eindruck eines typischen Verlaufs einer Exploration und untersuchen im darauffolgenden Kapitel den Einfluss des Parameters α, der bei der Auswahl des Zielpunkts für die Abwägung zwischen dem erwarteten Informationsgewinn und den Fahrtkosten verantwortlich ist (vgl. Kapitel 4.6). In Kapitel 6.3 wird demonstriert, dass die vorgestellte Strategie auch in der Lage, ist mehrstöckige Gebäude zu explorieren und damit die Möglichkeiten, die sich durch die Verwendung von MLS-Karten ergeben, auch voll ausgeschöpft werden können. In Kapitel 6.4 präsentieren wir ein Experiment, das mit einem echten Roboter auf dem Campus der Universität Freiburg durchgeführt wurde.

6.1 Beispiel eines Explorationsverlaufs

In diesem Kapitel soll ein Eindruck über den typischen Verlauf einer Exploration vermittelt werden. Die simulierte Umgebung besteht aus vier Zimmer, die alle über einen Korridor zugänglich sind, welcher in einem Foyer mündet. Zwei der Zimmer sind zusätzlich direkt miteinander verbunden. Für die Explorationsstrategie wurde $\alpha = 0.5$ gewählt, so dass Fahrtkosten und erwarteter Informationsgewinn gleichwertig die Auswahl der Zielpunkte beeinflussen.

In Abb. 6.1 ist die Startposition des Roboters als grüner Würfel dargestellt, sowie die ausgewählten Zielpunkte als rote Kugeln. Der Übersichtlichkeit wegen sind die Zwischenziele, an denen ebenfalls 3D-Scans durchgeführt wurden, nicht eingezeichnet. Der erste ausgewählte Zielpunkt (Abb. 6.1a) führt den Roboter auf den Korridor. Mit dem Erreichen des zweiten Ziels (Abb. 6.1b) sind die ersten beiden

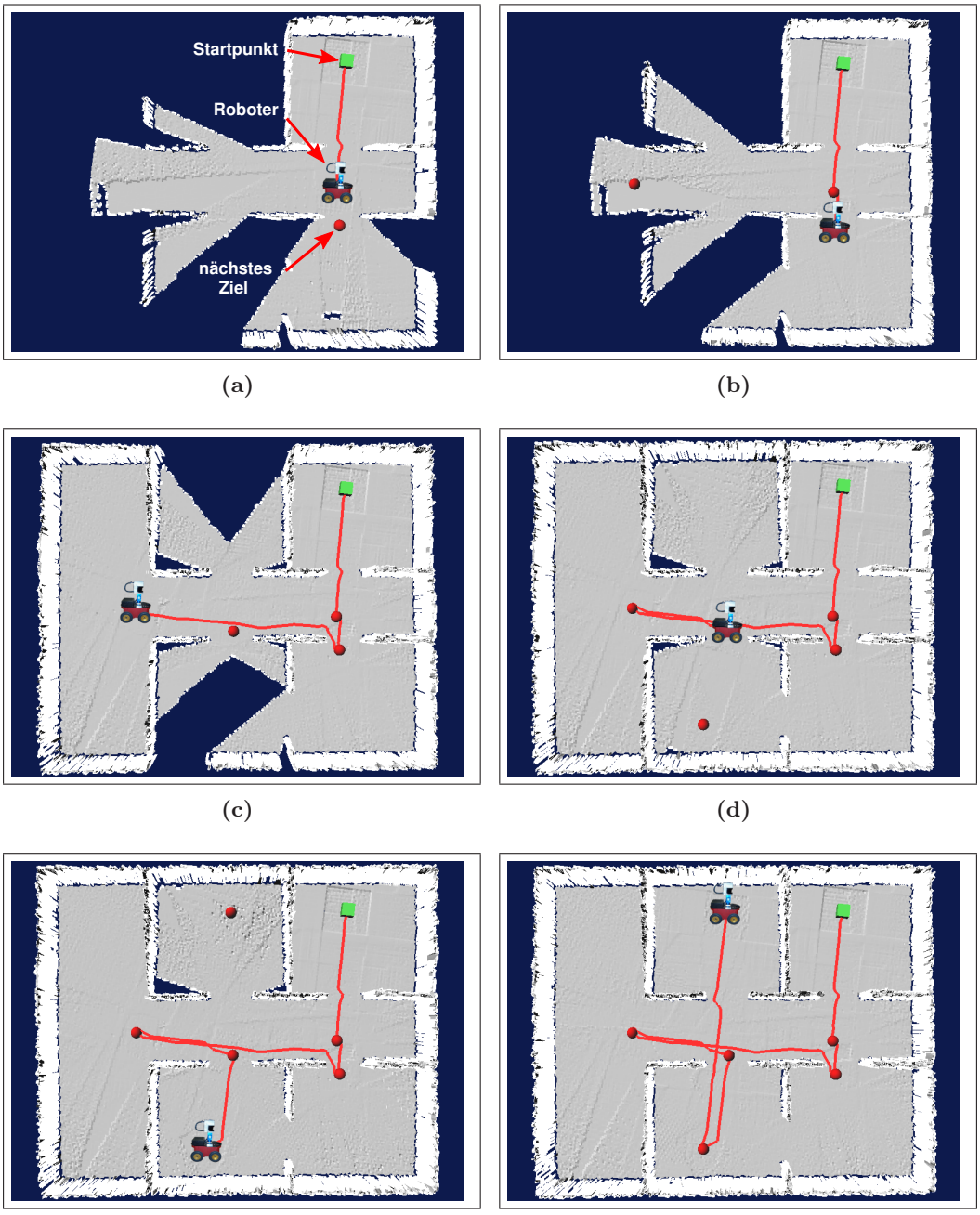
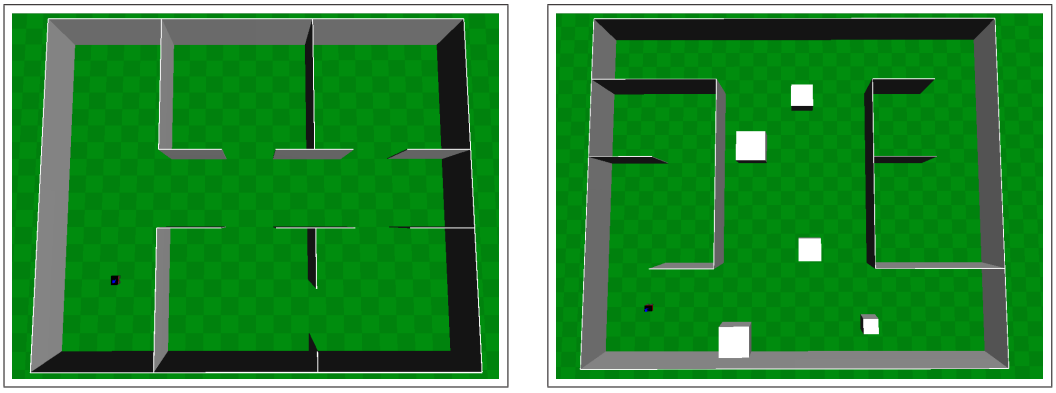


Abbildung 6.1: Beispiel eines Explorationsverlaufs in einer simulierten Umgebung.



(a) **(b)**

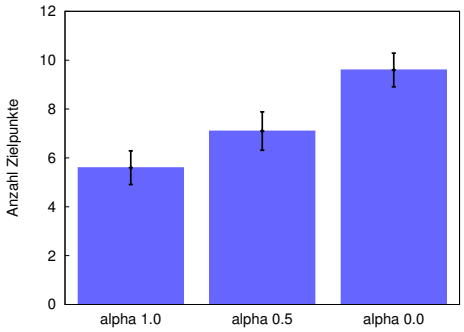
Abbildung 6.2: Die zwei für die Experimente verwendeten simulierten Umgebungen. Als Größenvergleich ist der Roboter jeweils unten links zu sehen. Ein Quadrat der Bodentextur entspricht einem Quadratmeter. Die weißen Würfel in Umgebung 2 sollen durch Abschattungen die Exploration schwieriger gestalten.

Zimmer exploriert. Es werden nun Zielpunkte vor dem Durchgang zum angrenzenden Zimmer, sowie Zielpunkte im Korridor generiert. Ausgewählt wird ein Zielpunkt am Ende des Korridors, der einen guten Einblick in das noch fast unexplorierte Foyer erlaubt. Dort angekommen (Abb. 6.1c) führt der nächste Zielpunkt den Roboter an den Eingang des unteren, linken Zimmers. Nach dem 3D-Scan an dieser Stelle (Abb. 6.1d) ist dieses und auch das letzte Zimmer bis auf kleine Bereiche in den Ecken exploriert. Es werden in beiden Zimmern Zielpunkte erzeugt und der Roboter entscheidet sich für das Zimmer, vor dem er gerade steht (Abb. 6.1e). Schließlich wird der letzte ausgewählte Zielpunkt im gegenüberliegenden Zimmer angefahren und die Exploration der Umgebung ist damit abgeschlossen (Abb. 6.1f).

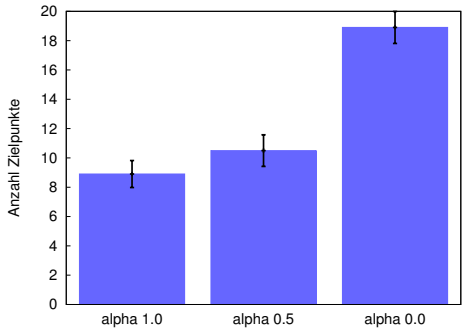
6.2 Evaluation verschiedener Explorationsstrategien

Um den Einfluss des Parameters α quantitativ einschätzen zu können, wurden in zwei unterschiedlichen Umgebung (vgl. Abb. 6.2) zehn zufällige Start-Positionen erzeugt. Für jede dieser Positionen wurde die Exploration mit den Werten $\alpha \in \{1.0, 0.5, 0.0\}$ durchgeführt, sodass also insgesamt 60 Einzelexperimente stattfanden. Diese Werte entsprechen einer Strategie, die rein auf die Maximierung des Informationsgewinns bedacht ist ($\alpha = 1.0$), einer Strategie, die nur die Fahrtkosten berücksichtigt ($\alpha = 0.0$), sowie einer ausgewogenen Strategie ($\alpha = 0.5$). Wir wollen diese drei Strategien im Folgenden als I-, F-, und A-Strategie bezeichnen.

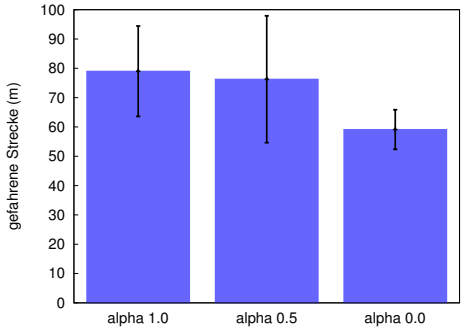
Abb. 6.3 zeigt die durchschnittliche Anzahl an ausgewählten Zielpunkten, die Ge-



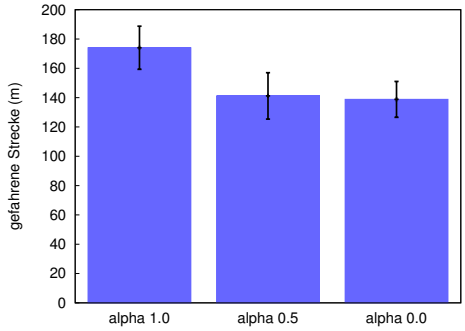
(a) Zielpunkte in Umgebung 1.



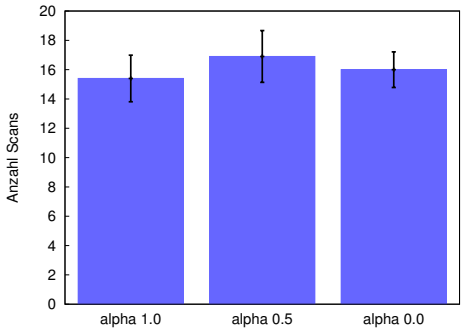
(b) Zielpunkte in Umgebung 2.



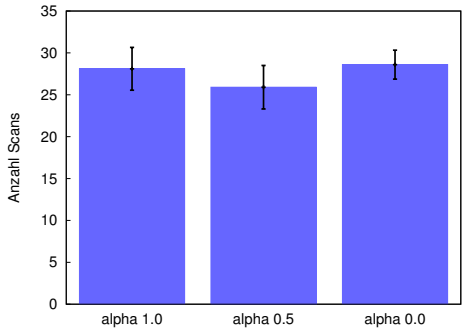
(c) Gefahrene Strecke in Umgebung 1.



(d) Gefahrene Strecke in Umgebung 2.



(e) 3D-Scans in Umgebung 1.



(f) 3D-Scans in Umgebung 2.

Abbildung 6.3: Verschiedene untersuchte Werte, aufgeschlüsselt nach verwendeter Strategie und Umgebung. Die Fehlerbalken geben die 95%-Konfidenzintervalle an.

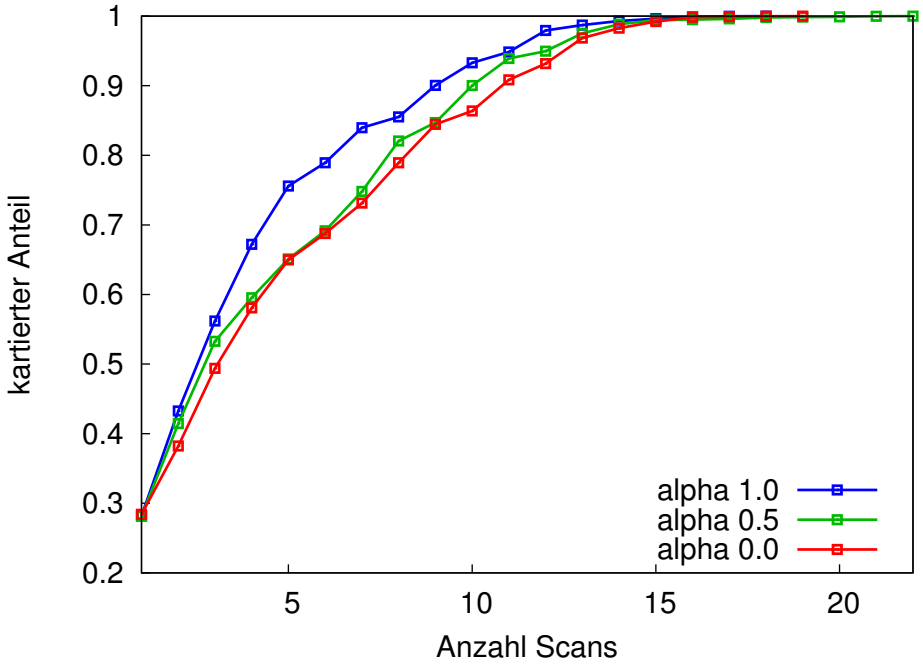
samtzahl der durchgeführten 3D-Scans (an Zielpunkten und Zwischenzielen), sowie die gefahrene Strecke. Die Fehlerbalken zeigen das 95%-Konfidenzintervall an (für eine normalverteilte Grundgesamtheit mit unbekannter Varianz, vgl. [7]).

Für beide Umgebungen zeigt sich, dass eine Strategie umso mehr Zielpunkte auswählt, je stärker sie die Fahrtkosten berücksichtigt (Abb. 6.3a, 6.3b). Das ist darin begründet, dass bei der F-Strategie die ausgewählten Zielpunkte besonders nahe liegen und somit die meisten der durchgeführten Scans auf Zielpunkte und nicht etwa auf Zwischenziele entfallen. Bei der I-Strategie müssen weniger Zielpunkte ausgewählt werden, da diese weiter entfernt liegen und bis zum Erreichen dieses Punktes bereits ein Teil der Umgebung durch Scans an Zwischenzielen exploriert wurde. Die A-Strategie liegt zwischen diesen Extremen.

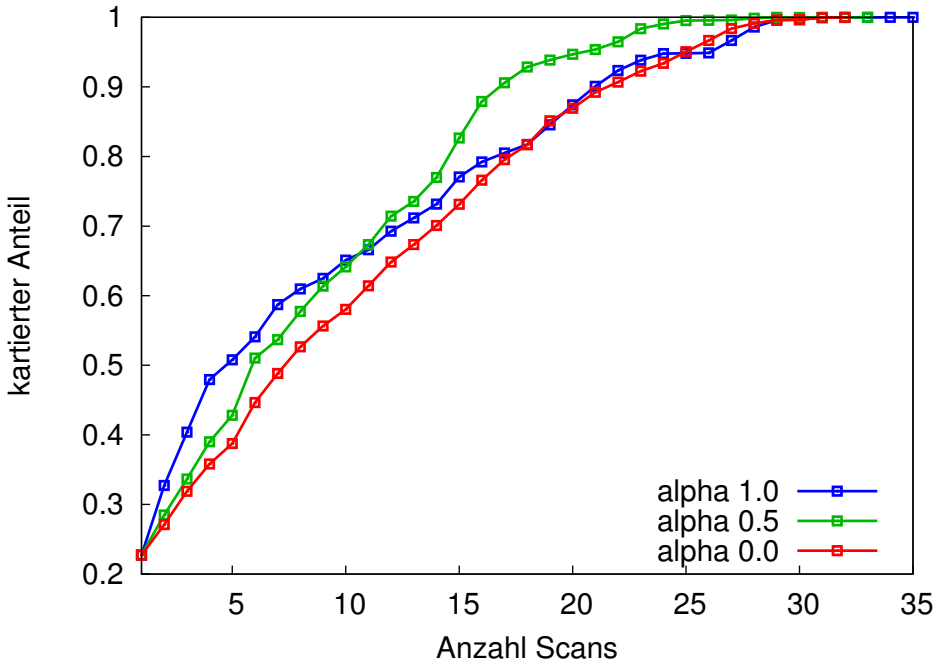
Vergleicht man die gefahrene Strecke (Abb. 6.3c, 6.3d), erzielt die F-Strategie erwartungsgemäß die geringsten Werte. Die I-Strategie dagegen ignoriert die Fahrtkosten vollständig und erkauft sich dieses Verhalten mit deutlich längeren Explorationswegen. Die A-Strategie liegt in Umgebung 1 näher an der I-Strategie, in Umgebung 2 näher an der F-Strategie. Mittelt man über beide Umgebungen liegt die A-Strategie auch hier wieder zwischen den Extremen. Die Fehlerbalken sind besonders in Umgebung 1 recht hoch. Hierbei ist jedoch zu beachten, dass die notwendige Pfadlänge für die Exploration zum Teil auch von der Startposition abhängt, was ein Grund für die Variationen innerhalb der jeweiligen Strategien sein kann.

Betrachtet man die Gesamtzahl der durchgeführten Scans, die auch Scans an Zwischenzielen beinhaltet, finden sich praktisch keine signifikanten Unterschiede zwischen den einzelnen Strategien (Abb. 6.3e, 6.3f). Dieses auf den ersten Blick erstaunliche Ergebnis ist jedoch eine Konsequenz der Unterschiede in den gefahrenen Strecken der einzelnen Strategien. Dies lässt sich am besten an den extremen Strategien I und F verdeutlichen. Die F-Strategie erzeugt zwar kurze Explorationswege, muss jedoch an jedem „nächstbesten“ Zielpunkt eine Messung vornehmen, obwohl ein nur etwas weiter gelegener Zielpunkt vielleicht mehr Informationsgewinn verspräche und einen zusätzlichen Scan überflüssig machen würde. Die I-Strategie dagegen kann nahegelegene Zielpunkte mit wenig Informationsgewinn ignorieren, erzeugt jedoch längere Explorationswege. Diese gegenläufige Faktoren scheinen sich aufzuheben und führen, bezüglich der Anzahl durchgeführter Scans, zu vernachlässigbaren Unterschieden zwischen den Strategien (vgl. dazu auch Abb. 6.5).

Zwar fällt die Anzahl der notwendigen Scans für die drei Strategien ungefähr gleich aus, jedoch können hinsichtlich der Effektivität mit der die Umgebung erkundet wird Unterschiede zwischen den einzelnen Scans bestehen. Um dies zu überprüfen wurde in Abb. 6.4 der Anteil der kartierten Umgebung über die Anzahl der Scans aufgetragen. Für Umgebung 1 (Abb. 6.4a) zeigt sich, dass die I-Strategie die Umgebung

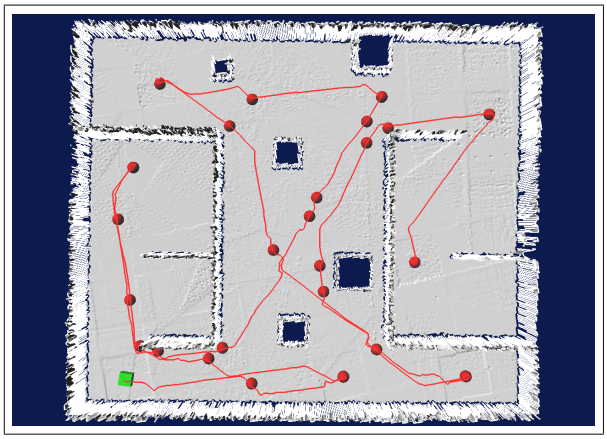


(a)

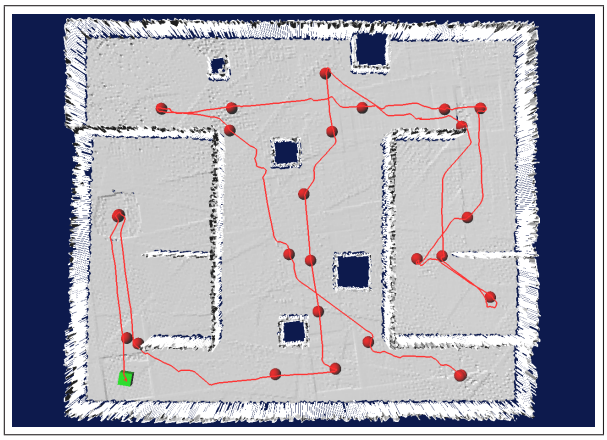


(b)

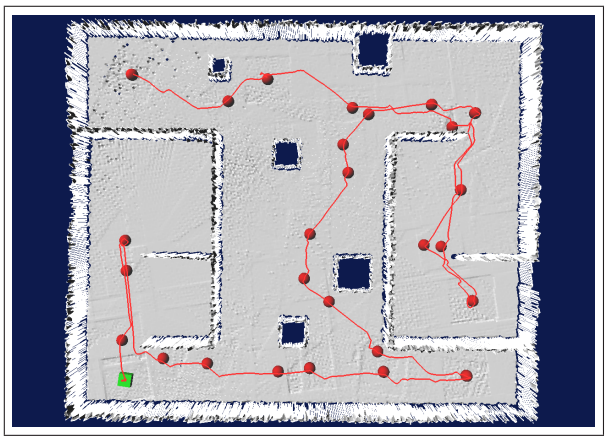
Abbildung 6.4: Anteil der kartierten Umgebung, aufgetragen über die Anzahl der durchgeführten Scans.



(a) I-Strategie ($\alpha = 1.0$)



(b) A-Strategie ($\alpha = 0.5$)



(c) F-Strategie ($\alpha = 0.0$)

Abbildung 6.5: Vergleich der Explorationswege verschiedener Strategien in Umgebung 2. Die roten Kugeln befinden sich an Positionen, an denen ein 3D-Scan durchgeführt wurde – dies beinhaltet Scans an Zielen und Zwischenzielen.

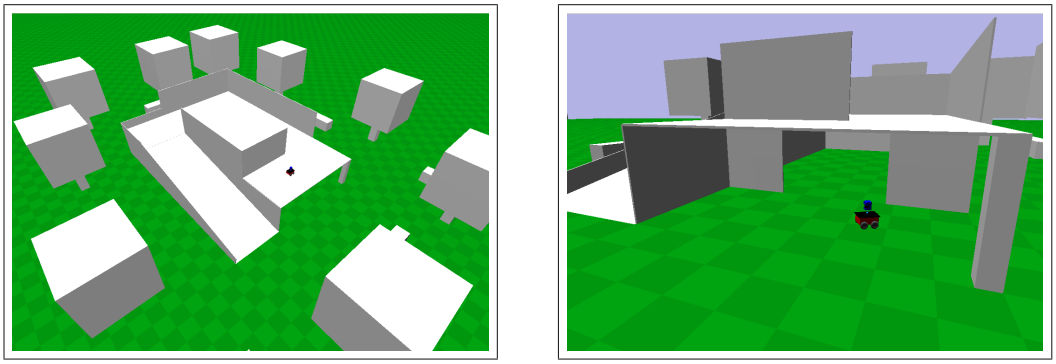
am schnellsten und die F-Strategie am langsamsten kartiert. Die A-Strategie liegt wieder zwischen diesen Strategien. Während die I-Strategie in Umgebung 1 also gut abschneidet, zeigt sich für Umgebung 2 (Abb. 6.4b) ein anderes Bild. Nach dem zehnten Scan fällt die I-Strategie (blaue Linie) hinter die A-Strategie zurück und verläuft dann ähnlich wie die F-Strategie. Der Grund hierfür ist, dass die I-Strategie in Umgebung 2 den Roboter öfter über bereits kartiertes Gebiet führt, und dieser dann auch in diesem Gebiet Zwischenmessungen durchführen muss. Diese Messungen tragen nicht dazu bei, neue Bereiche der Karte zu erkunden, sondern dienen lediglich dazu, die Überlappung mit der letzten lokalen Karte zu garantieren. Um dies zu illustrieren werden in Abb. 6.5 beispielhaft die Explorationswege der drei Strategien für die gleiche Startposition in Umgebung 2 gezeigt. Deutlich ist zu erkennen, dass je stärker die Fahrtkosten beachtet werden, desto seltener zu bereits besuchten, aber nicht vollständig explorierten Gebieten zurückgekehrt werden muss.

Die „beste“ Strategie gibt es nicht. Je nachdem welche Faktoren zu berücksichtigen sind, muss der Parameter α entsprechend gewählt werden. Ist die Berechnung eines neuen Zielpunkts teuer, scheint die I-Strategie angebracht. Soll auf kurze Explorationswege geachtet werden, ist man mit der F-Strategie besser aufgestellt. Die A-Strategie hat sich dagegen als robuster Mittelweg erwiesen, vor allem, wenn man den Verlauf in Abb. 6.4b bedenkt, in der sie bessere Ergebnisse liefert, als die Extrem-Strategien.

6.3 Exploration eines simulierten, mehrstöckigen Gebäudes

In diesem Simulationsexperiment wird der Innen- und Außenbereich eines Gebäudes exploriert. Das Gebäude verfügt über zwei Stockwerke, die jeweils aus einem Raum mit einer Grundfläche von 12×8 Metern bestehen. Im Obergeschoss befindet sich zudem ein ungesicherter Balkon mit einer Fläche von 4×8 Metern. Über eine seitlich angebrachte Rampe mit einer Steigung von $20,8\%$ ist das Obergeschoss mit dem Außenbereich verbunden. Der Eingang zum Erdgeschoss befindet sich auf der gleichen Hausseite wie der überhängende Balkon. Das Haus ist von einigen Quadern umgeben, die Bäume und Büsche darstellen sollen und die Lokalisierung erleichtern (Abb. 6.6).

Durch das Experiment soll demonstriert werden, dass der Roboter unter Verwendung der hier vorgestellten Strategie in der Lage ist, übereinanderliegende Ebenen zu explorieren und damit die Möglichkeiten, welche die Umgebungsrepräsentation durch MLS-Karten bietet, auch voll ausnutzen kann. Zudem soll gezeigt werden, dass der Roboter auch negative Hindernisse, wie sie der Abgrund am ungesicherten Balkon



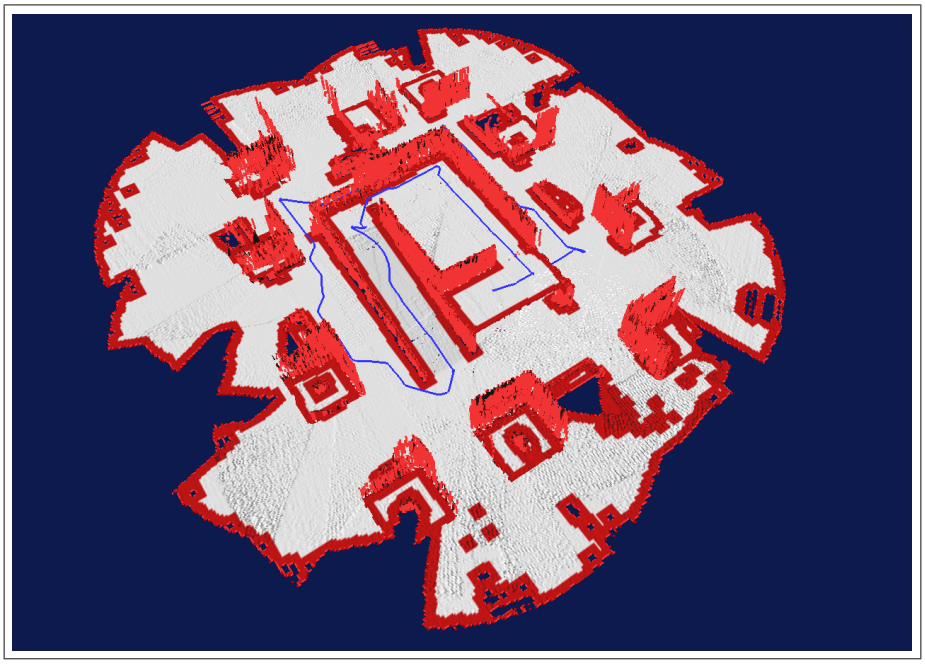
(a) **(b)**

Abbildung 6.6: (a) Überblick über die simulierte Umgebung. **(b)** Nahaufnahme des Eingangs zum Raum im Erdgeschoss. Als Größenvergleich ist in beiden Bildern der Roboter zu sehen (Modell eines ActivMedia Pioneer 2-AT).

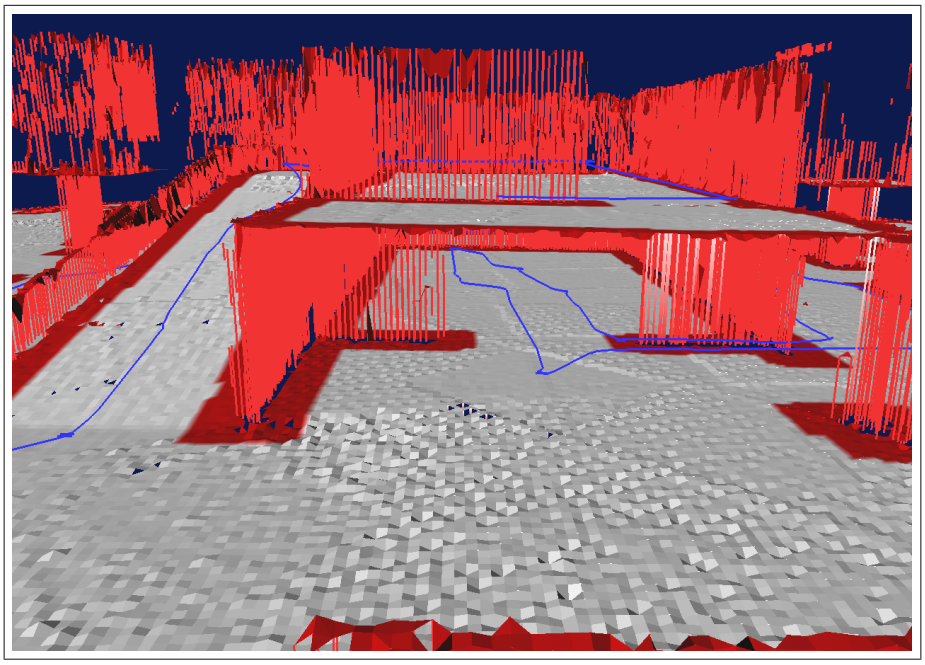
darstellt, erkennt.

Um die Exploration auf die Umgebung des Hauses zu beschränken, werden potenzielle Zielpunkte nur innerhalb eines rechteckigen, das Haus umfassenden Bereich generiert. Für dieses Experiment setzen wir $\alpha = 0.5$. Der Roboter startete die Exploration auf dem ungesicherten Balkon, durchquerte das Zimmer im Obergeschoss und fuhr die Rampe runter. Am Boden des Außenbereichs angekommen, wurde das Haus einmal umrundet, bevor er sich dem Innenbereich im Erdgeschoss zuwandte. Nach der Exploration des Zimmers im Erdgeschoss wurde noch ein Zielpunkt an der bereits explorierten Rückseite angefahren, der bei der ersten Vorbeifahrt wohl zu wenig Informationsgewinn versprach. Mit Erreichen dieses Zielpunktes war die Exploration abgeschlossen. Es wurden 18 Zielpunkte bestimmt, 29 3D-Scans durchgeführt und dabei eine Strecke von 212 m zurückgelegt. Die erstellte Karte besteht aus ca. 185 000 Patches. In Abb. 6.7 ist die erstellte Traversierbarkeitskarte nach Erreichen des letzten Zielpunkts dargestellt. Im unteren Bild ist ein Detailausschnitt am Eingang des Gebäudes dargestellt, in dem man erkennt, dass der Roboter auch in das Untergeschoss eingefahren ist und dieses komplett exploriert hat. Dabei wurden auch mehrere „dreidimensionale“ Schleifen zwischen lokalen Karten im Ober- und Untergeschoss geschlossen.

Die Umgebung enthält typische Elemente, die unter Verwendung von zweidimensionalen Umgebungsmodellen, aber auch bei einfachen dreidimensionalen Modellen, etwa Höhenkarten, Schwierigkeiten bereiten. Unter Verwendung einer zweidimensionalen Karte könnte der Roboter bereits am ungesicherten Balkon abstürzen. Unter Verwendung von Höhenkarten ist es möglich den Balkon als negatives Hindernis zu



(a) Erstellte Karte nach Erreichen des letzten Zielpunkts.



(b) Detailansicht am Eingang im Erdgeschoss.

Abbildung 6.7: Die erstellte Karte mit eingezeichneter Traversierbarkeit (weiß traversierbar, rot Hindernis) nach Erreichen des letzten Zielpunkts. Die blaue Linie zeigt den vom Roboter genommenen Weg und verläuft ca. 0.2 m über dem Boden.

erkennen, jedoch unterstützen diese Karten nicht die Exploration übereinanderliegender Ebenen.

6.4 Experiment mit einem echten Roboter

Das Verfahren wurde mit einem echten Roboter auf dem Campus der Universität Freiburg getestet. Wir verwendeten einen ActivMedia Pioneer 2-AT mit Pan-Tilt-Einheit und einem SICK-Laserscanner (für Details siehe Anhang B, S. 79).

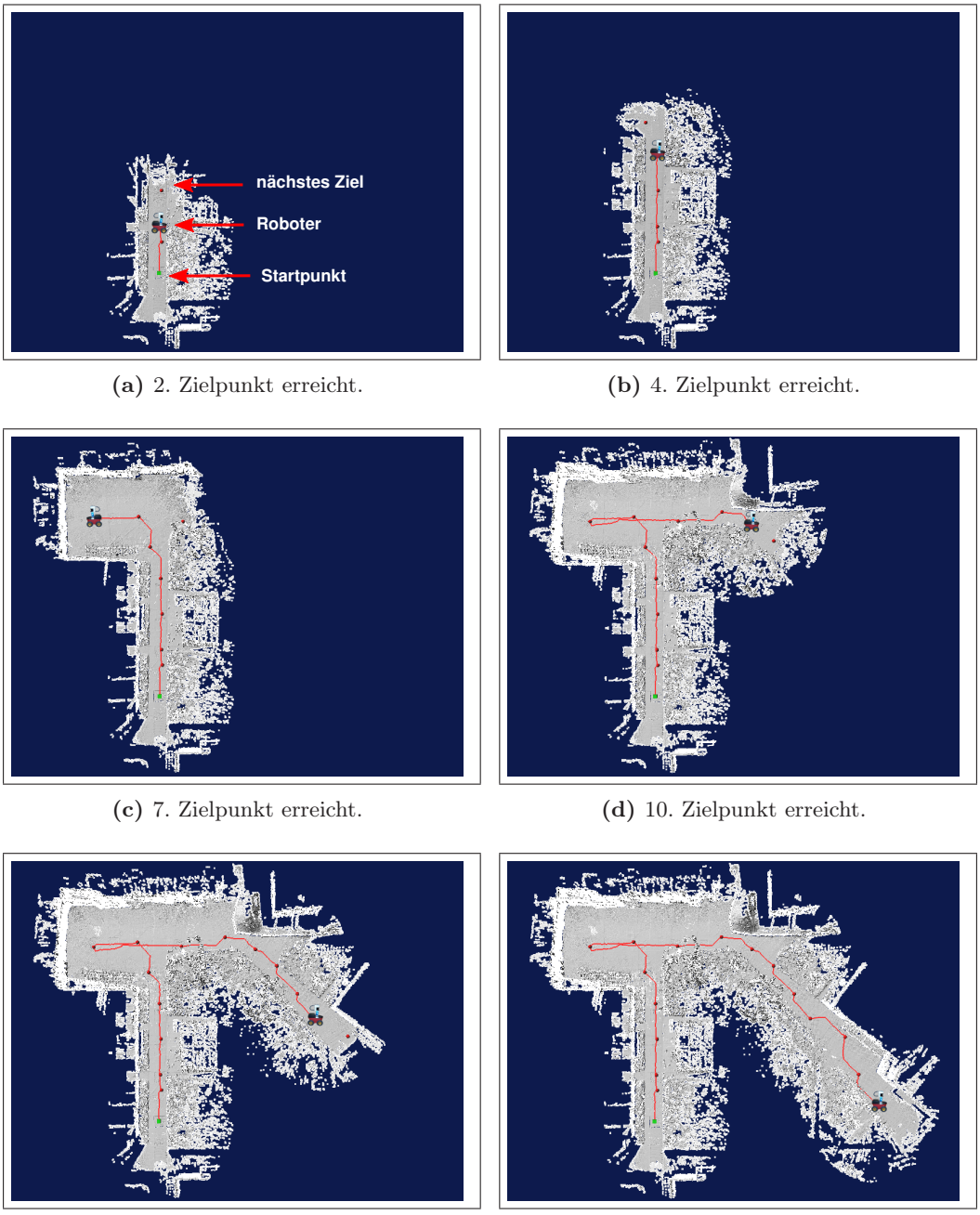
Der Explorationsverlauf ist in Abb. 6.8 dargestellt. Um der Exploration eine initiale Richtung vorzugeben, wurde die Lage der potenziellen Zielpunkte auf die Halbebene vor der Startposition des Roboters eingeschränkt. Über den Parameter $\alpha = 0.5$ wurde eine ausgewogene Strategie gewählt. Der Roboter folgte zunächst einem Weg, der links von einer Hauswand und rechts von einer Grasfläche und einem Gewächshaus begrenzt ist. Das Gras ist recht hoch und stellte aufgrund der Unebenheiten für den Roboter ein Hindernis dar. Am Ende der Hauswand angekommen entschied sich der Roboter links in einen Platz einzufahren, der an drei Seiten von Häuser begrenzt ist. Nach wenigen Scans war der Platz erkundet und der Roboter wandte sich den hinter ihm liegenden Zielpunkten zu, um die Exploration der weiteren Umgebung aufzunehmen. Dabei fuhr er nahe genug an früheren Scan-Positionen vorbei, sodass zwei Schleifen (passiv) geschlossen werden konnten. An einer Weggabelung angekommen (Abb. 6.8d und Abb. 6.9b) entschied er sich für den rechten Weg und fuhr bis zur Grenze der Halbebene. Dort angekommen wurde das Experiment beendet.

Der Roboter fuhr vollständig autonom eine Strecke von 186 m, bestimmte dabei 18 Zielpunkte und führte 26 Scans durch. Die letzte Karte enthält knapp 410 000 Patches. In Abb. 6.10 ist eine perspektivische Ansicht der erstellten Traversierbarkeitskarte zu sehen.

6.5 Fazit

Durch die in diesem Kapitel beschriebenen Experimente wurde gezeigt, dass der Roboter in der Lage ist, vollständig autonom eine unbekannte, dreidimensionale Umgebung zu erkunden – sowohl in der Simulation, als auch unter Realweltbedingungen. Es wurden negative Hindernisse erkannt und vermieden, Gebäude auf mehreren Ebenen exploriert und dabei Schleifen über Stockwerksgrenzen hinweg geschlossen. Zudem haben wir die Eigenschaften verschiedener Strategien durch die Variation des Parameters α untersucht.

Während der Durchführung der Simulationsexperimente und der Experimente mit dem echten Roboter hat sich aber auch gezeigt, dass die Registrierung neuer Informa-



(a) 2. Zielpunkt erreicht.

(b) 4. Zielpunkt erreicht.

(c) 7. Zielpunkt erreicht.

(d) 10. Zielpunkt erreicht.

(e) 13. Zielpunkt erreicht.

(f) 16. Zielpunkt erreicht.

Abbildung 6.8: Verlauf der Exploration auf dem Campus der Universität Freiburg.

(a) Perspektive kurz vor dem Startpunkt.

(b) Perspektive kurz vor der Weggabelung.

Abbildung 6.9: Fotos der explorierten Strecke des Campus.

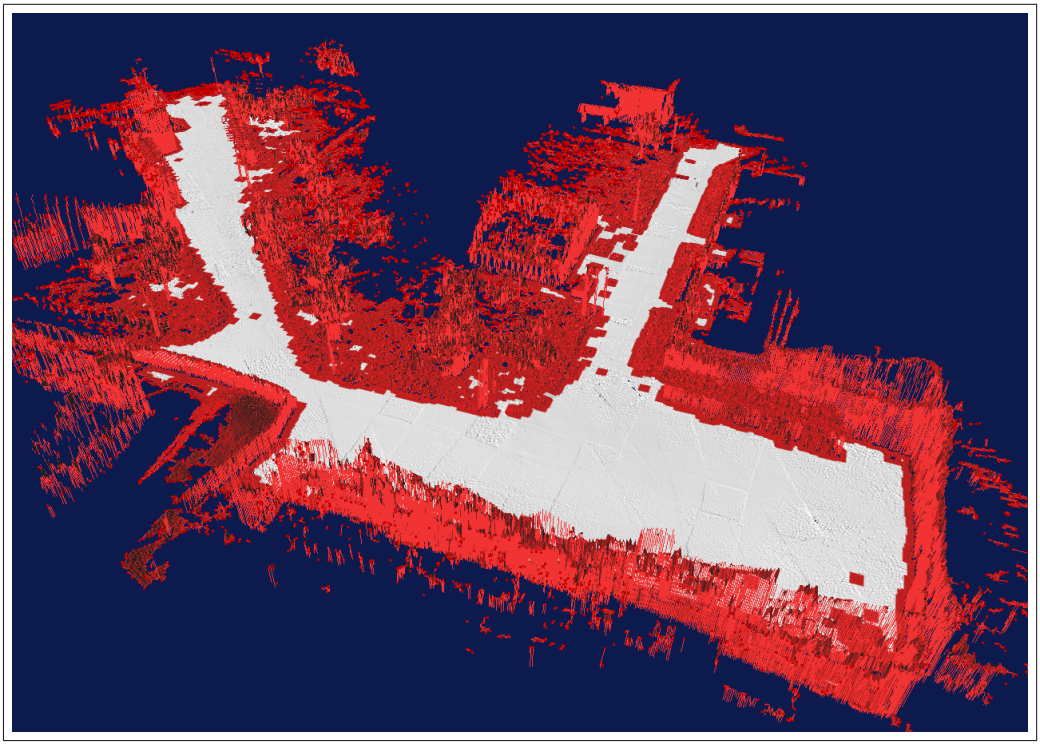


Abbildung 6.10: Perspektivische Ansicht der erstellten Traversierbarkeitskarte des Campus.

tionen in einem Online-Verfahren, wie sie die Exploration darstellt, eine schwierige Aufgabe ist. Die Lokalisierung muss mit einer unvollständigen Karte zurechtkommen, was besonders am Rand der Karte schwierig ist, da der Laserscanner hier während der Navigation bereits Informationen über neue Hindernisse liefert, diese aber noch nicht in der Karte verzeichnet sind. Zudem können inkrementelle Fehler des Scan-Matching-Verfahrens erst nach dem Schließen einer Schleife aufgelöst werden. Schon kleine Fehler in der Schätzung der Winkel einer lokalen Karte können jedoch dazu führen, dass bei der Integration der lokalen Karte in die globale Karte größere Bereiche nicht mehr traversierbar sind.

Kapitel 7

Zusammenfassung

In der vorliegenden Diplomarbeit wurde ein Verfahren vorgestellt, das einem mobilen Roboter die autonome Exploration einer anfangs unbekannten Umgebung ermöglicht. Dabei wurde die häufig getroffene Annahme einer zweidimensionalen Umgebung fallen gelassen und die dreidimensionale Struktur explizit berücksichtigt. Als Umgebungsrepräsentation wurden MLS-Karten (*multi-level surface maps*) verwendet [45], wodurch eine kompakte, dreidimensionale Modellierung der Umgebung erreicht wurde. Eine zentrale Fragestellung bei der Exploration ist die Auswahl des nächsten Zielpunkts, bei der wir die Fahrtkosten zu diesem Punkt und den erwarteten Informationsgewinn eines an dieser Stelle durchgeführten 3D-Scans berücksichtigten. Bei der Bestimmung von möglichen Zielpunkten, den Fahrtkosten, und des erwarteten Informationsgewinns wurde berücksichtigt, dass der Roboter in einer dreidimensionalen Umgebung agiert. Bei den Fahrtkosten wurde deshalb nicht nur die Distanz, sondern auch die Neigung und Rauheit des Geländes beachtet. Dazu wurde eine neue Traversierbarkeitsanalyse für MLS-Karten vorgestellt, die eine Ebene in die Nachbarschaft der Patches einpasst und im Gegensatz zu 2D-Ansätzen auch in der Lage ist, negative Hindernisse zu erkennen. Bei der Generierung von möglichen Zielpunkten und der Berechnung des erwarteten Informationsgewinns tragen wir der dreidimensionalen Struktur durch eine 3D-Raycasting-Methode Rechnung.

Das vorgestellte Verfahren wurde in das Softwaresystem Morpheus integriert und Experimente mit der Simulationsumgebung Gazebo sowie mit einem echten Roboter durchgeführt. Es wurden verschiedene Variationen des Explorationsverhalten evaluiert, um den Einfluss der Abwägung zwischen erwartetem Informationsgewinn und Fahrtkosten zu untersuchen. Wir demonstrierten zudem, dass das Verfahren in der Lage ist, übereinanderliegende Ebenen zu explorieren, und somit die Möglichkeiten, welche die Verwendung von MLS-Karten bietet, voll genutzt werden können. Durch Experimente mit einem echten Roboter wurde gezeigt, dass das System auch unter

Realweltbedingungen funktioniert.

7.1 Ausblick

Die aktuelle Implementierung geht davon aus, dass der Roboter in einer statischen Umgebung agiert. Eine Erweiterung des Verfahrens auf dynamische Umgebungen, würde die autonome Navigation betreffen, die dynamische Objekte erkennen und diesen ausweichen müsste. Zudem müssten dynamische Objekte während eines 3D-Laserscans ausgefiltert werden oder der Laserscan einfach wiederholt werden.

Eine Schwierigkeit bei der Entwicklung des Verfahrens war die Tatsache, dass MLS-Karten die Bestimmung von nicht exploriertem Gebiet nicht direkt unterstützen. Die Patches in MLS-Karten modellieren nur von Hindernissen belegte Bereiche. Ob die Bereiche über oder unter den Patches frei von Hindernissen sind oder einfach noch nicht von einem Laserstrahl erfasst wurden, kann nicht entschieden werden. Die Einführung von *free space patches*, die von Laserstrahlen hindernisfrei durchlaufenen Bereichen entsprechen, würde diese Ambiguität aufheben. Gerade bei der Berechnung des erwarteten Informationsgewinnes wären solche Patches hilfreich. Jedoch würde man sich die Einführung solcher Patches mit einem erhöhten Speicherbedarf erkaufen und gerade die kompakte Repräsentation ist einer der Vorteile von MLS-Karten.

Die Bewertung der Zielpunkte könnte weitere Kriterien berücksichtigen. Neben dem Informationsgewinn in der Karte wäre der Informationsgewinn bezüglich der Lokalisierung interessant. Dazu müsste man entlang des geplanten Pfades zu einem potenziellen Zielpunkt Lasermessungen simulieren und Odometrieinformationen generieren, um die Partikel des Partikelfilters vorwärts zu propagieren. Für die Verteilung der Partikel im 6D-Konfigurationsraum ließe sich ein Maß der Unsicherheit definieren und über den Unterschied vor und nach Erreichen des fraglichen Zielpunktes wäre dann der Informationsgewinn oder -verlust definiert. Diese Idee wurde bereits teilweise implementiert. Als jedoch abzusehen war, dass dies die Auswahl von Zielpunkten deutlich verlangsamt, wurde im Hinblick auf die für die Diplomarbeit noch durchzuführenden, zeitintensiven Experimente das Vorhaben zunächst zurückgestellt. Die Implementierung der Raycasting-Methode bietet jedoch noch Potenzial für Optimierungen. Auch könnte man die Auswahl von Zielpunkte in mehrere Schritte aufteilen. Der erste Schritt würde nur die nach dem bisherigen Verfahren am besten bewerteten Zielpunkte behalten. Erst diese engere Auswahl würde man dann in einem zweiten Schritt zusätzlich bezüglich des Informationsgewinns des Partikelfilters bewerten. Diese zweistufige Auswahl könnte keine optimale Auswahl bezüglich aller Kriterien garantieren, da im ersten Schritt eventuell gute Zielpunkte verfrüht verworfen werden, sie stellt jedoch vermutlich eine gute Heuristik dar.

Ein weiteres Bewertungskriterium könnte der erwartete Energieverbrauch entlang eines Pfades sein. Zwar hängt dieser vor allem von der gefahrenen Distanz und der Steigung und den Unebenheiten des Geländes ab, die bereits als Fahrtkosten berücksichtigt werden – jedoch entspricht der relative Einfluss dieser Faktoren auf den Energieverbrauch wohl nicht der hier verwendeten Abbildung auf die Pfadkosten. Eine Abbildung dieser Faktoren wäre experimentell zu bestimmen. Ein geringer Energieverbrauch ist besonders für nicht radgetriebene Roboter, wie humanoide Roboter oder andere Laufroboter, von Bedeutung. Zusätzlich zur Distanz und Unebenheiten des Geländes könnten auch notwendige Drehungen auf einem geplanten Pfad beachtet werden, da diese ebenfalls einen erhöhten Energieverbrauch bedeuten und zudem die Odometrie verschlechtern. Unter Einbeziehung des angesprochenen Informationsgewinns der Lokalisierung würden Drehungen jedoch bereits implizit durch das Bewegungsmodell berücksichtigt werden.

Zwar findet bei der aktuellen Implementierung *loop closing* statt, es werden jedoch keine Schleifen *aktiv* geschlossen [41]. Der genommene Weg und die Positionen an denen lokale Karten aufgenommen wurden sind jedoch bekannt und ein aktives Schließen von Schleifen könnte deshalb leicht implementiert werden. Schwieriger ist es jedoch, den Informationsgewinn einer solchen Aktion abzuschätzen, was notwendig wäre, wenn man diese Aktionen auf konsistente Weise in das bestehende System der Zielpunktbewertung aufnehmen will. Alternativ könnte man sich auf einfache Heuristiken verlassen, wie z. B. immer dann eine Schleife zu schließen, wenn ein solcher Punkt nahe der aktuellen Position, oder nahe eines geplanten Pfades liegt.

Anhang A

Algorithmen und Definitionen

A.1 Dijkstra-Algorithmus

Die in Alg. 3 dargestellte Implementierung des Dijkstra-Algorithmus [34] berechnet die kürzesten Pfade in einem Graphen $G = (V, E)$ von einem Startpunkt $s \in V$ zu allen erreichbaren Knoten in V. Die gerichteten Kanten $E \subseteq V^2$ sind durch eine Kostenfunktion $c(v, v') : E \to \mathbb{R}_{\geq 0}$ gewichtet. In jedem Knoten $v \in V$ werden die Pfadkosten $pfadkosten(v) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ des kürzesten bisher gefundenen Pfades $p = \langle s, \ldots, v', v \rangle$ vom Startknoten s zu diesem Knoten gespeichert, sowie dessen Vorgänger $vorgänger(v) = v' \in V \cup \{\emptyset\}$. Es existieren im Allgemeinen mehrere optimale Pfade von s zu einem Knoten, weshalb Knoten auch mehrere Vorgänger haben können. Für die hier angedachte Verwendung zur Pfadplanung in MLS-Karten spielt es keine Rolle, für welchen Vorgänger (und somit für welchen optimalen Pfad) wir uns entscheiden – wir speichern daher nur einen Vorgänger.

Anfangs werden die Pfadkosten und Vorgänger aller Knoten auf ∞, bzw. $\emptyset$ gesetzt (Zeilen 1 bis 4). In den Zeilen 5 und 6 wird das Startelement initialisiert, das Pfadkosten 0 zu sich selbst hat und sein eigener Vorgänger ist. In einer Prioritätswarteschlange (*priority queue*, PQ, [34]) halten wir alle Knoten, die noch expandiert werden müssen. Knoten werden absteigend nach Wegkosten einsortiert. Diese Menge entspricht einer „Wellenfront“ von Knoten, die durch den Graph läuft. Anfangs enthält die PQ nur den Startknoten selbst (Zeile 8). Danach wird wiederholt der Knoten v mit den bisher kürzesten Pfadkosten aus der Prioritätswarteschlange entfernt (Zeile 10) und geprüft, ob seine direkten Nachbarn von diesem Knoten aus mit geringeren als den bisher bekannten Pfadkosten erreicht werden können (Zeile 12). Diese Kosten setzen sich zusammen aus den optimalen Kosten bis zum Knoten v plus den direkten Kosten von v zu n. Falls ein Nachbar dadurch günstiger erreicht werden kann, ist v Vorgänger von n auf dessen optimalen Pfad und die Kosten des

optimalen Pfades zu n können ersetzt und somit reduziert werden.

Ist die Prioritätswarteschlange leer, dann wurden alle Pfadkosten und Vorgänger bestimmt. Knoten mit Pfadkosten ∞ sind nicht von s aus erreichbar. Der optimale Pfad $\langle s, \ldots, v \rangle$ zu einem beliebigen, erreichbaren Knoten v kann durch Ablaufen der Vorgänger bestimmt werden. Würde man die Vorgänger nicht speichern, wäre die Bestimmung des Pfades auch durch einen Gradientenabstieg in den Pfadkosten möglich.

Algorithmus 3 Implementierung des Dijkstra-Algorithmus.

Eingabe: Graph (V, E), Startknoten s, Kostenfunktion c.

Ausgabe: Die Pfadkosten von s zu jedem Knoten $v \in V$, sowie implizit die optimalen Pfade, sofern diese existieren.

```
1:  for all v ∈ V do
2:    pfadkosten(v) ← ∞
3:    vorgänger(v) ← ∅
4:  end for
5:  pfadkosten(s) ← 0
6:  vorgänger(s) ← s
7:  initialisiere die Priority Queue PQ
8:  füge s mit Priorität 0 in PQ ein
9:  while PQ nicht leer do
10:   extrahiere den Knoten v mit niedrigster Priorität aus PQ
11:   for all n für die eine Kante e(v, n) existiert do
12:     if pfadkosten(n) > pfadkosten(v) + c(v, n) then
13:       pfadkosten(n) ← pfadkosten(v) + c(v, n)
14:       vorgänger(n) ← v
15:       if n bereits in PQ gespeichert then
16:         ersetze die Priorität von n in PQ durch pfadkosten(n)
17:       else
18:         füge n mit Priorität pfadkosten(n) in PQ ein
19:       end if
20:     end if
21:   end for
22: end while
```

A.2 Entropie

Die Entropie spiegelt die Unsicherheit einer Wahrscheinlichkeitsverteilung wider. Für eine Wahrscheinlichkeitsverteilung p über dem Ereignisraum X ist sie definiert als

$$H_p(X) = -E\{\log_2 p(x)\}. \tag{A.1}$$

Für eine diskrete Wahrscheinlichkeitsverteilung erhalten wir somit

$$H_p(X) = -\sum_{x \in X} p(x) \log_2 p(x). \tag{A.2}$$

und für eine kontinuierliche Verteilung

$$H_p(X) = -\int_{x \in X} p(x) \log_2 p(x) dx. \tag{A.3}$$

Im diskreten Fall gibt $-\log_2 p(x)$ an, wieviel Bits von einem optimalen Kodierer benötigt werden, um x zu enkodieren, wenn $p(x)$ die Wahrscheinlichkeit von x ist [42]. Eine wichtige kontinuierliche Verteilung ist die d-dimensionalen Gauß-Verteilung $\mathcal{N}(x; \mu, \Sigma)$ mit Kovarianz Σ. Für diese Verteilung ergibt sich die Entropie als

$$H_{\mathcal{N}} = \frac{d}{2}\left(1 + \log\left(2\pi\right)\right) + \frac{1}{2}\log\left(\det\left(\Sigma\right)\right) \tag{A.4}$$

und damit gilt für den Spezialfall einer eindimensionalen Gauß-Verteilung

$$H_{\mathcal{N}} = \frac{1}{2}\log_2\left(2\pi e\sigma^2\right). \tag{A.5}$$

Verwenden wir hierbei statt des 2er-Logarithmus den natürlichen Logarithmus, erhalten wir als Einheit nicht Bits, sondern Nats [11].

A.3 Kalman-Filter

Der Kalman-Filter [25, 42] kann als eine Implemetierung des Bayes-Filter aufgefasst werden, wobei bestimmte zusätzliche Annahmen über das Systemverhalten getroffen werden.

Bayes-Filter [42] dienen dazu, rekursiv die Zustände $x_{1:t}$ eines Systems zu verschiedenen Zeitpunkten $1 : t$ zu schätzen. Das System befindet sich zu jedem Zeitpunkt in genau einem Zustand $x \in X$ und wird über eine Kontrollaktion $u \in U$ in einen anderen Zustand versetzt oder in diesem belassen. Der Zustand x_t zum Zeitpunkt t hängt nur vom vorherigen Zustand x_{t-1} und einer Kontrollaktion u_t ab. Dabei ist der

Algorithmus 4 Bayes-Filter (nach [42]).

Eingabe: Zustandsschätung $bel(x_{t-1})$ zum Zeitpunkt $t-1$, Kontrollaktion u_t, Messung z_t

Ausgabe: Zustandsschätung $bel(x_t)$ zum Zeitpunkt t

1: **for all** x_t **do**
2: $\quad \overline{bel}(x_t) \leftarrow \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
3: $\quad bel(x_t) \leftarrow \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
4: **end for**

Zustand des Systems nur indirekt über Beobachtungen $z \in Z$ in Erfahrung zu bringen. Zustandsübergänge und Beobachtungen sind nichtdeterministisch und werden über Wahrscheinlichkeitsverteilungen modelliert. Die Übergangswahrscheinlichkeit

$$p(x_t \mid x_{t-1}, u_t) \tag{A.6}$$

gibt an, wie wahrscheinlich es ist, dass sich das System in Zustand x_t befindet, wenn es sich zuvor im Zustand x_{t-1} befand und die Kontrollaktion u_t ausgeführt wurde. Entsprechend gibt die Beobachtungswahrscheinlichkeit

$$p(z_t \mid x_t) \tag{A.7}$$

an, wie wahrscheinlich es ist, die Beobachtung (Messung) z_t zu erhalten, wenn sich das System in Zustand x_t befindet. Die Schätzung $bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$ für den aktuellen Zustand x_t des Systems ist wiederum eine Wahrscheinlichkeitsverteilung über alle Zustände und kann mittels der Wahrscheinlichkeiten (A.6) und (A.7), sowie der aktuellen Beobachtung z_t, der aktuellen Kontrollaktion u_t und der Schätzung $bel(x_{t-1})$ für den letzten Zustand rekursiv wie in Alg. 4 dargestellt berechnet werden (dabei ist η ein Normalisierungsfaktor).

Der Kalman-Filter nimmt an, dass die Schätzung des aktuellen Zustands $bel(x_t)$ normalverteilt ist und damit durch Mittelwert und Kovarianz (μ_t, Σ_t) repräsentiert werden kann. Entsprechend muss auch die initiale Schätzung $bel(x_0)$ normalverteilt sein. Zudem wird gefordert, dass Zustandsübergänge des Systems

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{A.8}$$

und Beobachtungen

$$z_t = C_t x_t + \delta_t \tag{A.9}$$

durch eine lineare Funktion mit additiven, gaußverteilten Fehlern ε_t (mit Kovarianz R) und δ_t (mit Kovarianz Q) beschreibbar sind. In Alg. 5 ist der Kalman-Filter

Algorithmus 5 Kalman-Filter (nach [42]).

Eingabe: Zustandsschätzung $(\mu_{t-1}, \Sigma_{t-1})$ zum Zeitpunkt $t-1$, Kontrollaktion u_t, Messung z_t

Ausgabe: Zustandsschätzung (μ_t, Σ_t) zum Zeitpunkt t

1: $\overline{\mu}_t \leftarrow A_t \mu_{t-1} + B_t u_t$
2: $\overline{\Sigma}_t \leftarrow A_t \Sigma_{t-1} A_t^T + R_t$
3: $K_t \leftarrow \overline{\Sigma}_t C_t^T \left(C_t \overline{\Sigma}_t C_t^T + Q_t \right)^{-1}$
4: $\mu_t \leftarrow \overline{\mu}_t + K_t \left(z_t - C_t \overline{\mu}_t \right)$
5: $\Sigma_t \leftarrow \left(I - K_t C_t \right) \overline{\Sigma}_t$

dargestellt, der analog zum Bayes-Filter aus der Zustandsschätzung $bel(x_{t-1})$, Kontrollaktion u_t und Messung z_t die Zustandsschätzung $bel(x_t)$ berechnet. In Zeilen 1 und 2 wird die Vorhersage $\overline{bel}(x_t) = (\overline{\mu_t}, \overline{\Sigma_t})$ berechnet. Zeile 3 berechnet den sogenannten Kalman-Gain, der beeinflusst, wie stark die Vorhersage von der Beobachtung beeinflusst wird. Die resultierende neue Schätzung $bel(x_t) = (\mu_t, \Sigma_t)$ des Systemzustands wird schließlich in Zeilen 4 und 5 berechnet.

Anhang B

Technische Daten des Roboters

B.1 Roboter „Herbert“

Der für das Experiment auf dem Universitätscampus eingesetzte Roboter „Herbert“ besteht aus einer Pioneer 2-AT Basis von ActivMedia, sowie einer darauf montierten Pan-Tilt-Einheit, die einen SICK Laserscanner trägt. Zur Erstellung von 3D-Punktwolken während eines 3D-Scans kann der Laserscanner horizontal geschwenkt und vertikal gekippt werden. Für die Lokalisierung während der Fahrt wird der Laser noch vorne ausgerichtet.

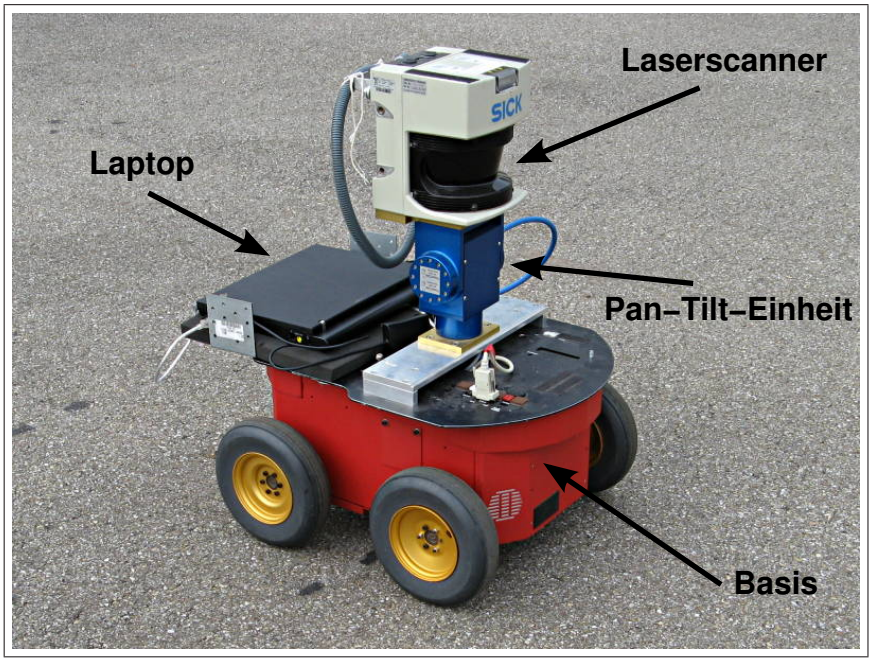


Abbildung B.1: Roboter „Herbert“.

B.2 Roboterbasis

Tabelle B.1: Technische Daten ActivMedia Pioneer 2-AT.

Abmessung (L × H × B)	50 cm × 49 cm × 26 cm
Gewicht	12 kg
Batterien	3 × 12 VDC
Max. Geschwindigkeit	0.7 $\frac{\mathrm{m}}{\mathrm{s}}$
Max. Steigung	40%
Max. Nutzlast	30 kg

Abbildung B.2: Roboterbasis ActivMedia Pioneer 2-AT.

B.3 Pan-Tilt-Einheit

Tabelle B.2: Technische Daten Amtec PowerCube Wrist PW 090.

	Pan	Tilt
Max. Winkelgeschwindigkeit	$149\ \frac{°}{s}$	$248\ \frac{°}{s}$
Lageerfassung	$894\ \frac{Inc}{°}$	$672\ \frac{Inc}{°}$
Auflösung	$4\ \frac{Bogensek.}{Inc}$	$5\ \frac{Bogensek.}{Inc}$
Genauigkeit der Positionierung	$\pm 0.02°$	$\pm 0.02°$
Spannung	24 ± 1 V	
Max. Strom	15 A	
Masse	3.4 kg	

Abbildung B.3: Pan-Tilt-Einheit Amtec PowerCube Wrist PW 090.

B.4 Laserscanner

Tabelle B.3: Technische Daten SICK LMS 291.

Abmessung (B × H × T)	155 mm × 185 mm × 156 mm
Masse	4.5 kg
Max. Reichweite	80 m
Messauflösung	10 mm
Max. Scanbereich	180°
Winkelauflösungen	0.25°/0.5°/1°
Spannung	24 VDC

Abbildung B.4: Laserscanner SICK LMS 291.

Abbildungsverzeichnis

Algorithmenverzeichnis

Literaturverzeichnis

[1] *Carmen – Robot Navigation Toolkit.* http://carmen.sourceforge.net/.

[2] *RoboCup.* http://www.robocup.org/.

[3] *The Player Project.* http://playerstage.sourceforge.net/.

[4] Amigoni, Francesco und Alessandro Gallo: *A multi-objective exploration strategy for mobile robots.* In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Seiten 3850–3855, Barcelona, Spanien, 2005. IEEE Press.

[5] Awerbuch, Baruch, Yossi Azar, Avrim Blum und Santosh Vempala: *New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen.* SIAM Journal on Computing (SICOMP), 28(1):254–262, 1998.

[6] Bares, J., M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons und W. Whittaker: *Ambler: An autonomous rover for planetary exploration.* IEEE Computer Magazine, 22(6):18–26, 1989.

[7] Bartsch, Hans-Jochen: *Taschenbuch mathematischer Formeln.* Fachbuchverlag Leipzig im Carl-Hanser-Verlag, 18. Auflage, 1999, ISBN 3-446-21048-2.

[8] Bresenham, Jack E.: *Algorithm for computer control of a digital plotter.* IBM Systems Journal, 4(1):25–30, 1965.

[9] Burgard, W., M. Moors, D. Fox, R. Simmons und S. Thrun: *Collaborative multi-robot exploration.* In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000.

[10] Burgard, W., M. Moors, C. Stachniss und F. Schneider: *Coordinated multi-robot exploration.* IEEE Transactions on Robotics, 21(3):376–378, 2005.

[11] Cover, Thomas M. und Joy A. Thomas: *Elements of Information Theory.* John Wiley & Sons, 1991, ISBN 0-471-06259-6.

[12] Ericson, Christer: *Real-time Collision Detection*. Morgan Kaufmann / Elsevier, 2005, ISBN 1-55860-732-3.

[13] Fong, T., M. Bualat, L. Edwards, L. Flückiger, C. Kunz, S. Y. Lee, E. Park, V. To, H. Utz, N. Ackner, N. Armstrong-Crews und J. Gannon: *Human-robot site survey and sampling for space exploration*. In: *Proc. of the AIAA Space Conf. (Space)*, San Jose, CA, USA, 2006.

[14] Fournier, Jonathan, Benoit Ricard und Denis Laurendeau: *Mapping and exploration of complex environments using persistent 3D model*. In: *Proc. of the Canadian Conf. on Computer and Robot Vision (CRV)*, Seiten 403–410, Montreal, Kanada, 2007.

[15] Freda, L. und G. Oriolo: *Frontier-based probabilistic strategies for sensor-based exploration*. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Seiten 3881–3887, Barcelona, Spanien, 2005. ISBN 0-7803-8914-X.

[16] Freund, Rudolf J. und William J. Wilson: *Regression Analysis – Statistical Modeling of a Response Variable*. Academic Press, 1998, ISBN 0-12-267475-8.

[17] Gerbaud, T., V. Polotski und P. Cohen: *Simultaneous exploration and 3D mapping of unstructured environments*. In: *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics (SMC)*, Band 6, Seiten 5333–5337, Den Haag, Niederlande, 2004. ISBN 0-7803-8566-7.

[18] Gerkey, B., S. Thrun und G. Gordon: *Parallel stochastic hill-climbing with small teams*. In: *Proc. of the 3rd Int. Workshop on Multi-Robot Systems*, Amsterdam, Niederlande, 2004.

[19] González-Baños, Héctor H. und Jean-Claude Latombe: *Navigation strategies for exploring indoor environments*. Int. Journal of Robotics Research (IJRR), 21(10-11):829–848, 2002.

[20] Grabowski, Robert, Pradeep Khosla und Howie Choset: *Autonomous exploration via regions of interest*. In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Band 2, Seiten 1691–1696, Las Vegas, NV, USA, 2003.

[21] Howard, Andrew, Lynne E. Parker und Gaurav S. Sukhatme: *Experiments with large heterogeneous mobile robot team: Exploration, mapping, deployment and detection*. Int. Journal of Robotics Research (IJRR), 25(5):431–447, 2006.

[22] Howard, Andrew, Gaurav S. Sukhatme und Maja J. Matarić: *Multirobot simultaneous localization and mapping using manifold representations.* Proceedings of the IEEE - Special Issue on Multi-robot Systems, 94(7):1360–1369, 2006.

[23] Jensfelt, P. und H. I. Christensen: *Active global localization for a mobile robot using multiple hypothesis tracking.* IEEE Trans. on Robotics and Automation, 17:748–760, 2001.

[24] Joho, Dominik, Cyrill Stachniss, Patrick Pfaff und Wolfram Burgard: *Autonomous exploration for 3D map learning.* In: *Fachgespräche Autonome Intelligente Systeme (AMS)*, Kaiserslautern, 2007.

[25] Kalman, R. E.: *A new approach to linear filtering and prediction problems.* Trans. ASME, Journal of Basic Engineering, 82:35–45, 1960.

[26] Kuhn, H. W.: *The Hungarian method for the assignment problem.* Naval Research Logistic Quarterly, 2(1):83–97, 1955.

[27] Kümmerle, Rainer, Rudolph Triebel, Patrick Pfaff und Wolfram Burgard: *Monte carlo localization in outdoor terrains using multi-level surface maps.* In: *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, Chamonix, Frankreich, 2007.

[28] Lacroix, S., A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb und R. Chatila: *Autonomous rover navigation on unknown terrains: Functions and integration.* Int. Journal of Robotics Research (IJRR), 21(10–11):917–942, 2002.

[29] Makarenko, Alexei A., Stefan B. Williams, Frederic Bourgault und Hugh F. Durrant-Whyte: *An experiment in integrated exploration.* In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Seiten 534–539, Lausanne, Schweiz, 2002.

[30] Mei, Yongguo, Yung-Hsiang Lu, C. S. George Lee und Y. Charlie Hu: *Energy-efficient mobile robot exploration.* In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Seiten 505–511, Orlando, FL, USA, 2006. ISBN 0-7803-9505-0.

[31] Moorehead, Steward J., Reid Simmons und William L. Whittaker: *Autonomous exploration using multiple sources of information.* In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Band 3, Seiten 3098–3103, Seoul, Südkorea, 2001.

[32] Newell, A. und H. A. Simon: *GPS, a program that simulates human thought.* In: Feigenbaum, Edward A. und Julian Feldman (Herausgeber): *Computers & Thought*, Seiten 279–293. MIT Press, 1995, ISBN 0-262-56092-5. Nachdruck.

[33] Nüchter, Andreas: *Autonome Exploration und Modellierung von 3D-Umgebungen.* Diplomarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn, 2002.

[34] Ottmann, Thomas und Peter Widmayer: *Algorithmen und Datenstrukturen.* Spektrum, Heidelberg, 4. Auflage, 2002, ISBN 3-8274-1029-0.

[35] Parra, Carlos, Rafael Murrieta-Cid, Michel Devy und Maurice Briot: *3-D modelling and robot localization from visual and range data in natural scenes.* In: *Proc. of the Int. Conf. on Computer Vision Systems (ICVS)*, Seiten 450–468, London, UK, 1999.

[36] Pfaff, Patrick und Wolfram Burgard: *An efficient extension of elevation maps for outdoor terrain mapping.* In: *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, Seiten 165–176, Port Douglas, QLD, Australien, 2005.

[37] Singh, S. und A. Kelly: *Robot planning in the space of feasible actions: two examples.* In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Band 4, Seiten 3309–3316, Minneapolis, MN, USA, 1996.

[38] Stachniss, Cyrill und Wolfram Burgard: *Exploring unknown environments with mobile robots using coverage maps.* In: *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Seiten 1127–1132, Acapulco, Mexiko, 2003.

[39] Stachniss, Cyrill und Wolfram Burgard: *Mapping and exploration with mobile robots using coverage maps.* In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Seiten 476–481, Las Vegas, NV, USA, 2003.

[40] Stachniss, Cyrill, Giorgio Grisetti und Wolfram Burgard: *Information gain-based exploration using rao-blackwellized particle filters.* In: *Proc. of Robotics: Science and Systems (RSS)*, Seiten 65–72, Cambridge, MA, USA, 2005.

[41] Stachniss, Cyrill, Dirk Hähnel, Wolfram Burgard und Giorgio Grisetti: *On actively closing loops in grid-based FastSLAM.* Advanced Robotics, 19(10):1059–1080, 2005.

[42] Thrun, Sebastian, Wolfram Burgard und Dieter Fox: *Probabilistic Robotics.* MIT Press, 2005.

[43] Tovey, C. und S. Koenig: *Improved analysis of greedy mapping.* In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Band 3, Seiten 3251–3257, Las Vegas, NV, USA, 2003.

[44] Triebel, Rudolph, Barbara Frank, Jörg Meyer und Wolfram Burgard: *First steps towards a robotic system for flexible volumetric mapping of indoor environments.* In: *Proc. of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[45] Triebel, Rudolph, Patrick Pfaff und Wolfram Burgard: *Multi-level surface maps for outdoor terrain mapping and loop closing.* In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Peking, China, 2006.

[46] Visser, Arnoud, Xingrui-Ji, Merlijn van Ittersum, Luis A. González Jaime und Laurenţiu A. Stancu: *Beyond frontier exploration.* In: *Proc. of the RoboCup Int. Symposium*, Atlanta, GA, USA, 2007.

[47] Whaite., P. und F. P. Ferrie: *Autonomous exploration: driven by uncertainty.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 19(3):193–205, 1997.

[48] Yamauchi, Brian: *A frontier-based approach for autonomous exploration.* In: *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1997.

[49] Yamauchi, Brian: *Frontier-based exploration using multiple robots.* In: *Proc. of the Second Int. Conf. on Autonomous Agents (AGENTS)*, Seiten 47–53, Minneapolis, MN, USA, 1998.