ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG

INSTITUT FÜR INFORMATIK

Arbeitsgruppe Autonome Intelligente Systeme

Prof. Dr. Wolfram Burgard

Techniken für die Outdoor-Navigation mit mobilen Robotern

Diplomarbeit

Patrick Pfaff

September 2004 – März 2005

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe und sämtliche Stellen, die wörtlich oder sinngemäß aus veröffentlichter oder unveröffentlichter Literatur entnommen wurden, als solche kenntlich gemacht habe. Außerdem erkläre ich, dass die Arbeit nicht – auch nicht auszugsweise – bereits für eine andere Prüfung angefertigt wurde.

(Patrick Pfaff)
Freiburg, den 14. März 2005

Danksagung

Mein Dank richtet sich an all die Menschen, die mich während der Erstellung meiner Diplomarbeit freundlich unterstützt haben. Ganz besonders danken möchte ich Herrn Prof. Dr. Wolfram Burgard für die sehr gute Betreuung, seinen Rat und seine Hilfe. Des Weiteren danke ich den Mitarbeitern der Arbeitsgruppe Autonome Intelligente Systeme – Rudolph Triebel, Dirk Hähnel und Cyrill Stachniss–, die mir hilfreich zur Seite standen und stets für eine äußerst angenehme Arbeitsatmosphäre sorgten. Außerdem möchte ich Kristian Kerting für die zahlreichen Anregungen und Diskussionen danken. Herzlicher Dank gebührt ebenfalls meiner Freundin Sandra und meinen Eltern für das Verständnis, das sie mir in dieser Zeit und während des gesamten Studiums entgegengebracht haben.

INHALTSVERZEICHNIS

KAPITEL 1

Einleitung / Aufgabenstellung

Die Navigation mit autonomen Robotern im Outdoor-Bereich ist eine große Herausforderung. Nicht umsonst ist für das erste Fahrzeug, das erfolgreich die Strecke zwischen Los Angeles und Las Vegas zurücklegt, ein Preisgeld von 2.000.000 Dollar ausgesetzt. Zu Recht trägt dieser Wettbewerb den Namen "Grand Challenge" und bisher ist keinem gelungen, diese Herausforderung zu bewältigen. Beim letzten Versuch im Jahre 2004 mussten die meisten Teams ihre Hoffnungen bereits nach wenigen Metern im Wüstensand begraben. Ein Beispiel dafür zeigt Abbilddung 1.1. Das beste Team schaffte es, gerade einmal 6 von 140 Meilen zurückzulegen.

Abbildung 1.1: Darpa Grand Challenge 2004

Ein weiteres Beispiel für die Schwierigkeit, unbekanntes Terrain zu befahren, sind die bisherigen Marsexpeditionen der NASA, bei denen mit wechselndem Erfolg fast immer Outdoor-Roboter zum Einsatz kamen. Abbildung 1.2 zeigt einen der lediglich halbautonom operierenden Roboter.

Abbildung 1.2: NASA Mars Rover

1.1 Zielsetzungen

Das Ziel der vorliegenden Arbeit ist es, ein Softwaresystem zu schaffen, das autonome Navigation im Outdoor-Bereich ermöglicht. Erhebungskarten (Elevation Maps) sind hierbei eine populäre Methode, unebene Umgebungen von mobilen Robotern zu repräsentieren. Die Elevation Maps speichern in jeder Zelle einen Höhenwert und modellieren dadurch eine Oberflächendarstellung, in der sich ein mobiler Roboter bewegen kann. Nutzt man eine solche $2\frac{1}{2}$D-Darstellung, um mit einem mobilen Roboter am Boden zu operieren, ergeben sich gerade bei vertikalen und überhängenden Objekten einige Probleme. In dieser Arbeit, die auf dem Ansatz der Elevation Maps aufbaut, werden sich vor allem folgenden Kernfragen herauskristallisieren:

- Wie kann man Elevation Maps von Outdoor-Umgebungen lernen?
- Wie kann man Elevation Maps auf spezielle Gegebenheiten in diesen Umgebungen anpassen? Dazu zählen in urbanen Gegenden beispiels-

weise Brücken, hohe Gebäude und Gebiete, die auf mehreren übereinanderliegenden Ebenen befahren werden können.

- Wie können Elevation Maps für Roboter- und Scanlokalisierung verwendet werden?
- Wie kann man in Elevation Maps Pfade planen und navigieren?

1.2 Verwandte Arbeiten

Bezugnehmend auf die Kernfragen aus Abschnitt 1.1 möchten wir nun einige frühere Projekte zu diesem Thema betrachten. Das Problem, dreidimensionale Repräsentationen zu lernen, wurde in der Vergangenheit bereits ausführlich behandelt.

Eine der populärsten Repräsentationen sind die Rohdaten, also Mengen von 3D-Punkten, oder triangulierte Netze [21, 1, 27, 13]. Diese Umgebungsmodelle sind äußerst genau und haben den Vorteil, dass sie mühelos texturiert werden können. Ihr Nachteil ist wiederum, dass sie sehr hohe Speicheranforderungen stellen, die linear zur Punktmenge ansteigen. Eine Alternative stellen dreidimensionale Gitter [16] oder baumbasierende Strukturen [24] dar, deren Komplexität lediglich linear zur Größe der Umgebung sind. Wir werden uns deshalb in dieser Arbeit vor allem mit Gitterstrukturen befassen und eine effektive Umgebungsrepräsentation auf der Basis von Erhebungskarten (Elevation Maps) erarbeiten. Elevation Maps stellen eine Art $2\frac{1}{2}$-dimensionale Repräsentation dar und vermeiden dadurch die Komplexität des vollständigen dreidimensionalen Raumes. Mit diesen Elevation Maps haben sich bereits einige Wissenschaftler beschäftigt. Bares et al. [2] und Hebert et al., [10] benutzen diese Darstellung, um die Umgebung eines 6-beinigen Marsroboters zu repräsentieren. Parra et al. [20] repräsentieren den Boden als Elevation Map und benutzen Stereo Vision, um Objekte auf dem Boden zu erkennen und zu tracken. Singh und Kelly [26] erstellen Elevation Maps aus Laser-Entfernungsmessungen (Laserscans) und nutzen sie, um einen Geländewagen zu navigieren. Ye und Borenstein [30] stellen ebenfalls einen Algorithmus zur Akquisition von Rohdaten in Elevation Maps vor. Sie verwenden ein Fahrzeug, das mit einem gekippten Laser-Range-Finder ausgestattet ist. Sie stellen darüber hinaus Filteralgorithmen vor, mit denen das Sensorrauschen und die aus der Bewegung resultierenden Fehler bereinigt werden können. Lacroix [12] berechnet Elevation Maps aus Bildern einer Stereokamera. Lacroix verwendet ein zweidimensionales Gitter und speichert in jeder dieser Zellen die durchschnittliche Höhe. Hygounenc et al. [11] erstellen Elevation

Maps mit einem autonomen Blimp. Sie verwenden dabei 3D Bildverarbeitungslgorithmen. Sie stellen einem Algorithmus vor, der es ihnen ermöglicht, Landmarken zu finden und zu verfolgen, um dadurch einzelne lokale Elevation Maps zu registrieren. Olson [19] beschreibt einen probabilistischen Lokalisierungsalgorithmus für Weltraumfahrzeuge, die zur Modellierung der Umgebung Elevation Maps verwenden.

Der Beitrag dieser Arbeit im Vergleich zu diesen Techniken ist unter Verschiedenen Aspekten zu betrachten. Als erstes werden wir den Ansatz der Elevation Map um eine Zellklassifikation erweitern. In dieser *Extended Elevation Map* werden wir die Zellen der Map in

- Zellen mit vertikalen Objekten,
- Zellen mit horizontalen Objekten und
- Zellen mit großen vertikalen Lücken unterteilen.

Diese Klassifizierung wird sich als sehr wichtig erweisen, wenn es darum geht, mit einem autonomen Fahrzeug in Stadtgebieten zu navigieren. Zusätzlich werden wir zeigen, wie sich durch diese Klassifizierung die Effizienz des Scan-Matchings zweier Elevation Maps steigern lässt.

1.3 Aufbau dieser Arbeit

Zu Beginn werden wir uns in Kapitel 2 mit der Wahl der Umgebungsrepräsentation beschäftigen und die Elevation Map in ihrer erweiterten Form vorstellen. In Kapitel 3 zeigen wir, wie diese Elevation Maps zum Kartenbau, zur Lokalisierung und zur Navigation angewandt werden können. In Kapitel 4 werden wir die Effizienz der gewählten Datenstrukturen und der darauf angewandten Algorithmen nachweisen. Kapitel 5 fasst die gewonnen Erkenntnisse nochmals zusammen. Im Anhang, Kapitel 6, finden sich Informationen zur Hardware des verwendeten Robotersystems.

KAPITEL 2

Repräsentation von Outdoor-Umgebungen

Es gibt verschiedene Möglichkeiten, 3D-Punktmengen von Outdoor-Umgebungen zu repräsentieren. In dieser Arbeit werden wir uns auf die üblichen Gitterrepräsentationen in zwei und drei Dimensionen beschränken. Da eines der Hauptprobleme bei der Outdoor-Navigation in Echtzeit die Wahl einer adäquaten Datenstruktur darstellt, werden wir zuerst diese Gitterrepräsentationen einführen und miteinander vergleichen. Dieser Vergleich wird uns danach direkt zu der von uns gewählten Datenstruktur führen.

2.1 3D-Punktmengen / Rohdaten

Abbildung 2.1 zeigt eine typische Punktmenge, die man bei einem 3D-Scan mit einem Laser-Range-Finder in einer Outdoor-Umgebung erhält. Die hier dargestellte Punktmenge besteht aus ca. 200.000 Punkten. Eine Karte eines größeren Gebietes von ca 100×100 Metern, die aus ca 50-100 solcher Scans besteht, erreicht demnach bis zu 20.000.000 Punkte(siehe auch Experiment in Kapitel 4.1). Man ist also gezwungen eine extrem große Datenmenge zu verarbeiten. Ein weiteres Problem dieser Daten ist, dass in gewissen Gebieten, wie direkt vor dem Roboter, große Punkthäufungen in anderen Regionen hingegen geringe Punkthäufungen auftreten. Direkt vor dem Roboter gibt es sehr viele Messungen, die sich kaum unterscheiden. In größeren Entfernungen

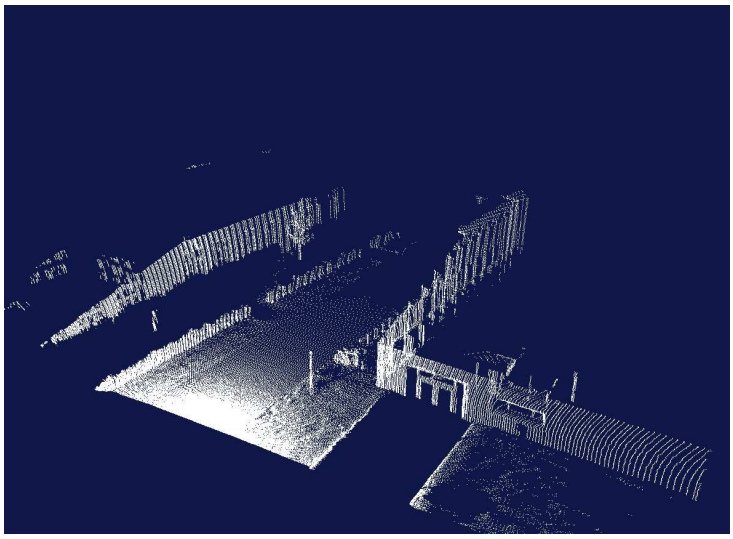

Abbildung 2.1: Typische Punktmenge aus einem Laserscan

ergibt sich genau das umgekehrte Problem. Man benötigt also eine Heuristik zum Bestimmen, welche Punkte wichtig oder unwichtig sind, um die Datenmenge sinnvoll zu reduzieren.

Das Hauptproblem dieser Form der rohen Daten ist allerdings, dass durch eine Punktmenge alleine nur wenige Informationen repräsentiert werden, die für die Navigation mit mobilen Robotern wichtig sind. Eine gängige herangehensweise, um diese Informationen speichern zu können, ist es, die Welt in Gitterzellen einzuteilen. Die Eigenschaften der einzelnen Zellen lassen sich aus den Lasermessungen ableiten. Dafür gibt es prinzipiell zwei verschiedene Ansätze: das Endpunktmodell und das Strahlenmodell. Was man genauer darunter versteht und wo die Vor- und Nachteile liegen, sollte aus den folgenden Abschnitten klar werden.

2.2 2D-Grid

Möchte man die Welt durch ein zweidimensionales Gitter (2D-Grid) darstellen, muss man die Lasermessungen nicht als 3D-Punkte sonder als Länge der Laserstrahlen interpretieren. Das heißt, man verfolgt jeden Strahl auf seinem Weg durch das Gitter und berechnet daraus eine Belegtheitswahrscheinlichkeit für jede tangierte Zelle. Dies erfolgt im Allgemeinen über einen inkrementellen Ray-Casting Algorithmus, wie ihn Abbildung 2.2 zeigt.

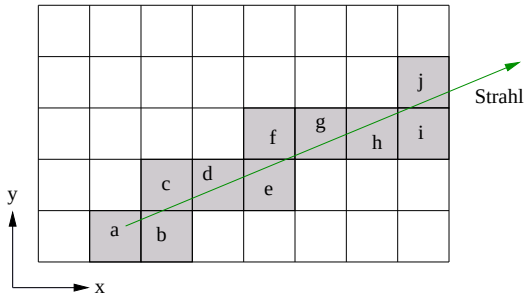

Abbildung 2.2: Inkrementelles Ray Casting in 2D

Hat man alle Belegtheitswahrscheinlichkeiten berechnet, ist die Pfadplanung und Navigation problemlos möglich. Allerdings ist diese Umgebungsrepräsentation lediglich in Indoor-Szenarien praktikabel, weil man davon ausgeht, dass sich der Roboter immer in der Gitterebene bewegt. Das ist in einer Outdoor-Umgebung jedoch nur selten der Fall. In Wirklichkeit findet der Roboter nämlich Umgebungen mit Steigungen, Gefällen, Hügeln und Senken vor. Die Hügel und Steigungen werden als nicht traversierbar klassifiziert, da alle Zellen, in denen ein Strahl in einem Punkt endet, der oberhalb der Gitterebene liegt, als "belegt" angenommen werden. Die Senken und Gefälle hingegen werden in einem 2D-Grid als traversierbar klassifiziert, da in der Gitterebene kein Hinderniss festgestellt werden kann. Diese Datenrepräsentation ist also für Outdoor-Daten ungeeignet.

2.3 3D-Grid

Wesentlich geeigneter ist ein 3D-Grid. Die Berechnung der Belegtheitswahscheinlichkeiten entspricht im wesentlichen der 2D-Version. Das Hauptproblem dieser Darstellung ist allerdings der hohe Speicher -und Berechnungsaufwand. Den Speicheraufwand könnte man durch Baumstrukturen reduzieren, indem man die Zahl der Zellen drastisch verringert. Dadurch würde sich der Aufwand für das 3D-Ray-Casting allerdings nocheinmal erhöhen.

Ideal wäre also eine Darstellung mit den Eigenschaften eines 3D-Grids verbunden mit einem Berechnungsaufwand der von einer Darstellung im Endpunktmodell ausgeht. D.h., dass man nicht die Strahlen durch das Gitter verfolgt, sondern die Lasermessungen als Punkte auffasst, die man direkt in das Gitter einfügen kann. Somit könte auf die aufwendigen Ray-Casting-Algorithmen verzichtet werden. Aus diesem Grund verwenden wir in dieser Arbeit die Elevation Map, die wir im nächsten Abschnitt vorstellen werden.

2.4 Elevation Map $2\frac{1}{2}$D-Grid

Eine Elevation Map basiert auf dem Endpunktmodell und stellt eine Art Histogramm dar: jede Gitterzelle erhält eine Höhe, die sich aus den Scanpunkten, die in dieser Zelle liegen, berechnen lässt. Der Berechnungsaufwand ist somit linear zur Anzahl der Scanpunkte. Da unter Umständen Scanpunkte mit sehr unterschiedlichen "Höhen" in eine Zelle fallen, muss man sich eine Darstellung überlegen, die alle Punkte angemessen repräsentiert. Dazu kann man den Mittelwert der "Höhen" μ und die dazu gehörende Varianz σ berechnen. Abbildung 2.3 zeigt die Definition der einzelnen Zellen.

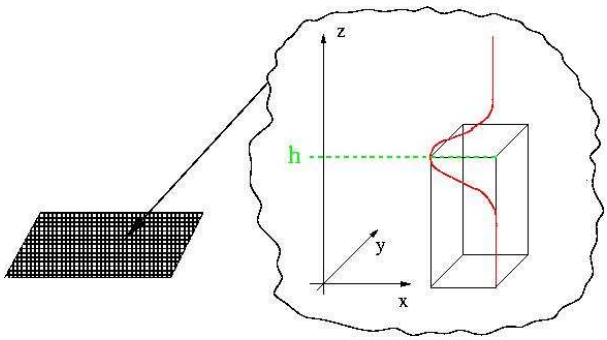

Abbildung 2.3: Eine Grid Zelle

Diese beiden Werte, μ und σ, können mit Hilfe eines Kalman Filters [15] zu jedem Zeitpunkt t aktualisiert werden. Das hat den Vorteil, dass zum einen mehrere Messungen in eine Elevation Map integriert werden können und zum anderen die Varianz einer Messung in die Elevation Map mit eingeht. Wir gehen davon aus, dass sich die Varianz der Lasermessungen proportional zur gemessenen Entfernung d verhält, wie Abbildung 2.4 zeigt.

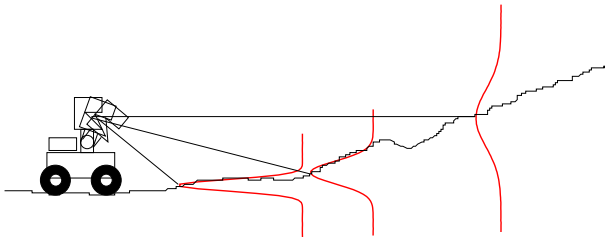

Abbildung 2.4: Integration neuer Messwerte

Die Zellparameter nach dem Einfügen von $t-1$ Punkten in die Zelle werden mit $\mu_{1:t-1}$ und $\sigma_{1:t-1}$ bezeichnet. Der Vektor $P_t(x_t, y_t, z_t)$ ist der nächste in die Gridzelle einzufügende Punkt mit $\mu_t = z_t$ und $\sigma_t = \xi \cdot d$, wobei die Konstante ξ aus der Spezifikation des Lasers resultiert. Die aktualisierten Zellparameter berechnen sich dann wie folgt:

$$\mu_{1:t} = \frac{\sigma_t^2 \mu_{1:t-1} + \sigma_{1:t-1}^2 \mu_{1:t}}{\sigma_t^2 + \sigma_{1:t-1}^2} \tag{2.1}$$

$$\frac{1}{\sigma_{1:t}^2} = \frac{1}{\sigma_t^2} + \frac{1}{\sigma_{t-1}^2} \tag{2.2}$$

Diese Darstellung hat einige Vorteile:

- Im Vergleich zum 3D-Grid benötigt diese Darstellung wesentlich weniger Speicher und die Berechnung aus einer Punktmenge ist mit linearem Aufwand verbunden.
- Der Zugriff auf jede Zelle kann in Konstantzeit erfolgen.
- Die Berechnung der Traversierbarkeit ergibt sich aus den Zellparametern μ und σ.
- Die Pfadplanung entspricht der Pfadplanung in 2D.
- Durch die Kalman Filterung wird das Sensormodell berücksichtigt und
- die Möglichkeit geschaffen, mehrere Scans in eine Elevation Map zu integrieren.

Allerdings ergeben sich aus dem Zwang, für jede Zelle eine Höhe schätzen zu müssen, auch einige Nachteile:

- Es ist nicht möglich, vertikale Objekte wie beispielsweise Wände zu erkennen. Man erhält lediglich einen Mittelwert der gemessenen Höhen.
- Dieser Mittelwert ist, da die Daten mit einem Roboter vom Boden aus akquiriert wurden, vom Standpunkt des Roboters abhängig.
- Jede Zelle hat nur eine Höhe. Was passiert also, wenn mehrschichtige Zellen vorliegen? Solche Zellen liegen beispielsweise bei Brückendurchfahrten und befahrbaren Rampen vor. In diesen Fällen ist die Höhe,

die sich aus dem Mittelwert μ der z-Werte aller Zellpunkte ergibt, keine optimale Darstellung. Abbildung 2.5 zeigt diesen Fall anhand einer Punktmenge und der dazugehörenden Elevation Map. Durch die Wahl von μ als Höhe der Gitterzellen erscheint die Brücke als großes, unumgängliches Hindernis in der Elevation Map.

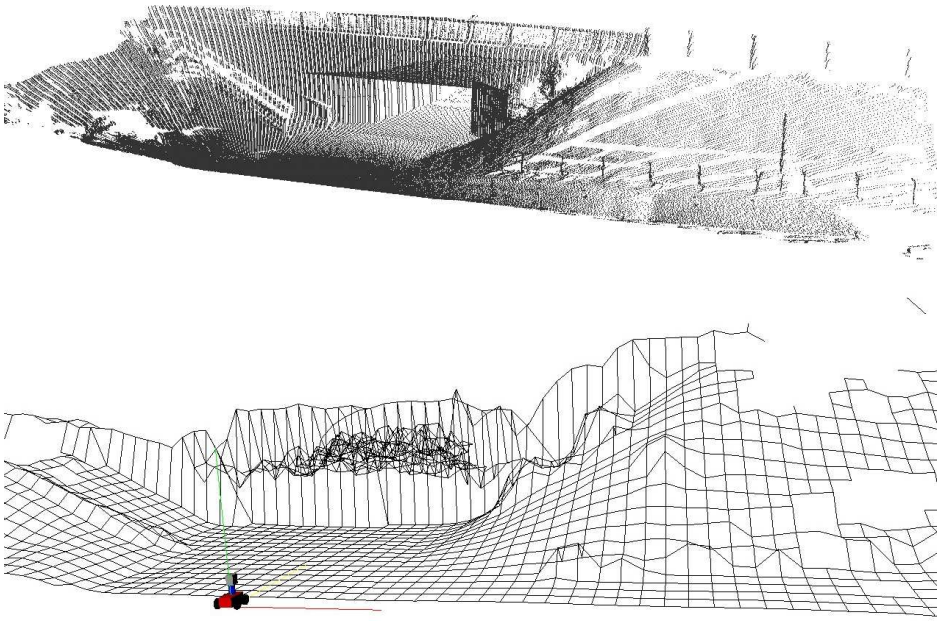

Abbildung 2.5: Probleme bei Brückendurchfahrten

2.5 Bestimmung traversierbarer Zellen in Elevation Maps

Diese Nachteile lassen den Schluss zu, dass es notwendig ist, die Elevation Maps zu erweitern. Um die *Extended Elevation Maps* einführen zu können, ist es notwendig, den Begriff der traversierbarkeit einzelner Zellen zu klären. Dies werden wir in diesem Abschnitt vornehmen.

Um sicheres Navigieren zu ermöglichen, ist es wichtig, festzustellen, welche Zellen traversierbar sind und welche nicht. Zur Detektion der Traversierbarkeit kann man die beiden Zellparameter μ und σ verwenden. Dafür wird aus

jeder Zelle C_i ein Punkt P_i berechnet. Mit Hilfe von n Punkten $P_1, P_2, ..., P_n$, die aus n benachbarten Zellen extrahiert werden, wird eine eindeutige Ebene E_j mit der Normalen $\mathbf{N_j}$ berechnet, welche die Summe aller Abstände zu den Punkten $P_1, P_2, ..., P_n$ minimiert. Im Beispiel, das in den Abbildungen 2.6 und 2.7 zu sehen ist, wurde $n = 4$ gesetzt. D.h., es werden immer vier Nachbarzellen gleichzeitig betrachtet und auf Traversierbarkeit geprüft.

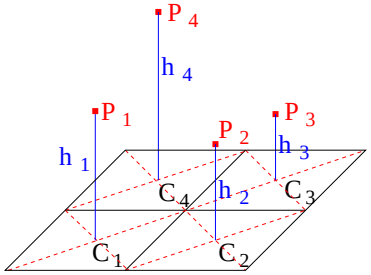

Abbildung 2.6: Punktextraktion für Ebenenfitting

Die Zellen C_1, C_2, C_3, C_4 werden als *traversierbar* klassifiziert, falls der z-Wert der Normalen $\mathbf{N_j}$ einen Schwellwert übersteigt. Dies entspricht dem Ansatz, die Neigung der Ebene auf einen Winkel α zu begrenzen. Für diesen Winkel gilt: $\cos \alpha = N_j \cdot e_3$, wobei e_3 den Einheitsvektor $(0, 0, 1)$ symbolisiert.

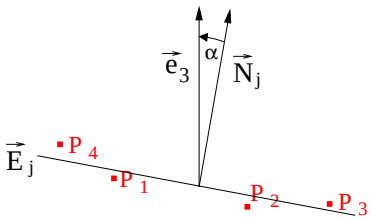

Abbildung 2.7: In Bildebene projiziertes Ebenenfitting

2.5.1 μ- und σ-Detektion

Im Allgemeinen gibt es zwei Ursachen für Nichttraversierbarkeit:

- Hindernisse wie beispielsweise Mauern, Fahrzeuge, Bäume etc. oder
- schlecht passierbarer und rauher Untergrund wie beispielsweise Wiesen, Kies, Geröll etc.

Daher liegt es nahe, zwei voneinander unabhängige Traversierbarkeitsberechnungen vorzunehmen. Die μ-Detektion und die σ-Detektion. Bei der μ-Detektion verwendet man zur Berechnung der Punkte P_i, die zum Ebenefitting herangezogen werden, als z-Koordinate den Zellparameter μ. Es ist direkt einsehbar, dass auf diese Art und Weise feste Hindernisse wie z.B. Mauern detektiert werden können. Dieser Fall wird in Abbildung 2.8 gezeigt. Traversierbare Zellen werden durch die grüne Farbe gekennzeichnet.

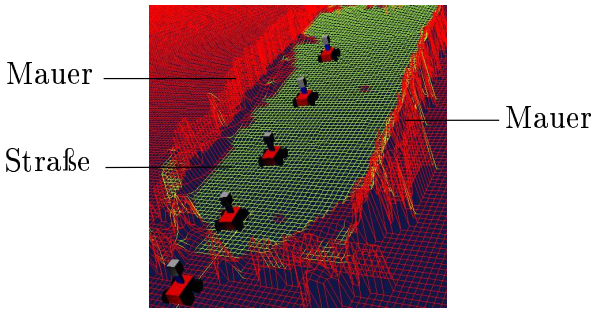

Abbildung 2.8: μ-Detektion

Bei der σ-Detektion verwendet man den Zellparameter σ als z-Koordinate für die Bestimmung der Punkte P_i. Dies ermöglicht beispielsweise die Unterscheidung von Gras und asphaltierter Straße (siehe Abbildung 2.9). Grün kennzeichnet hier ebenfalls Traversierbarkeit.

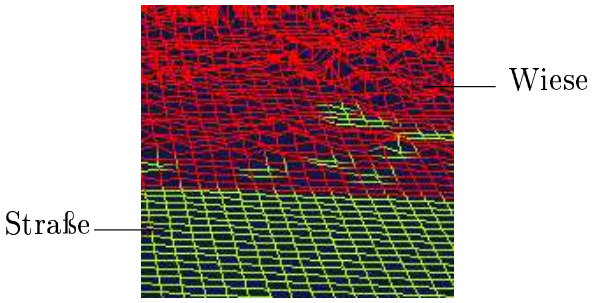

Abbildung 2.9: σ-Detektion

Eine Zelle wird genau dann als traversierbar klassifiziert, wenn sie in beiden Tests als traversierbar erkannt wird. Es gibt allerdings theoretische Szenarien mit großen Flächen von nicht traversierbaren Zellen, die ein konstant hohes σ und μ besitzen und deshalb fälschlicherweise als traversiebar klassifiziert werden. In unseren praktischen Experimenten konnten wir diesen Fall jedoch nicht beobachten.

2.6 Extended Elevation Maps

Wie bereits angemerkt, hat die Darstellung der realen Welt in einer Elevation Map einige Nachteile. Deshalb ist es notwendig, die Elevation Maps zu erweitern, was auch den Namen erklärt. Diese Erweiterung umfasst im Wesentlichen eine Klassifizierung der nicht traversierbaren Zellen. Betrachtet man die Berechnung der traversierbaren Zellen, so wird klar, dass bei den nicht traversierbaren Zellen eine Fehlklassifikation vorkommen kann. Besonders einleuchtend wird dieser Umstand, wenn man Abbildung 2.5 betrachtet. Diese Abbildung zeigt eine Unterführung, die fälschlicherweise als "nicht traversierbar" erkannt wird. Wie dieses Problem umgangen werden kann, wird im Folgenden beschrieben.

2.6.1 Unterteilung der nicht traversierbaren Zellen

Die nicht travesierbaren Zellen teilen wir in vier Klassen ein:

- Zellen mit vertikalen Objekten,
- Zellen mit einer vertikalen Lücke,
- Zellen, die von Oben gesehen wurden und
- Bodenzellen.

Die ersten drei Zelltypen werden wir genauer betrachten und erklären, wie man diese Einteilung erreichen kann. Die vierte Klasse dient lediglich dazu, die weniger aussagekräftigen Zellen von der Menge der nicht traversierbaren Zellen abzutrennen, was im Prinzip einer Kantenextraktion entspricht. Wir nehmen an, nicht traversierbare Bodenzellen sind genau diejenigen, die ihre Nachbarzellen im Schnitt um nicht mehr als einen Schwellwert δ überragen. In Experimenten haben sich Schwellwerte zwischen 0,15 und 0,2 Metern bewährt. Bei diesen Zellen kann darüberhinaus die Traversierbarkeitsklassifikation unsicher sein, was daher rührt, dass die Dichte der Messpunkte quadratisch zur Entfernung zum Roboter abnimmt. D.h., es sind in größerer Entfernung zum Roboter sehr häufig nicht ausreichend Zellen belegt, so dass durch die μ- und σ-Detektion keine zuverlässige Aussage über die Traversierbarkeit getroffen werden kann.

Zellen mit vertikalen Objekten

Bewegt man sich mit einem Roboter entlang des Bodens einer urbanen Umgebung, müssen einige Dinge beachtet werden. Abbildung 2.10 zeigt eine typische Szene in einer solchen Umgebung. Auf der linken Seite sind Bäume zu sehen, etwas weiter rechts eine Wand und im Vordergrund eine Straße.

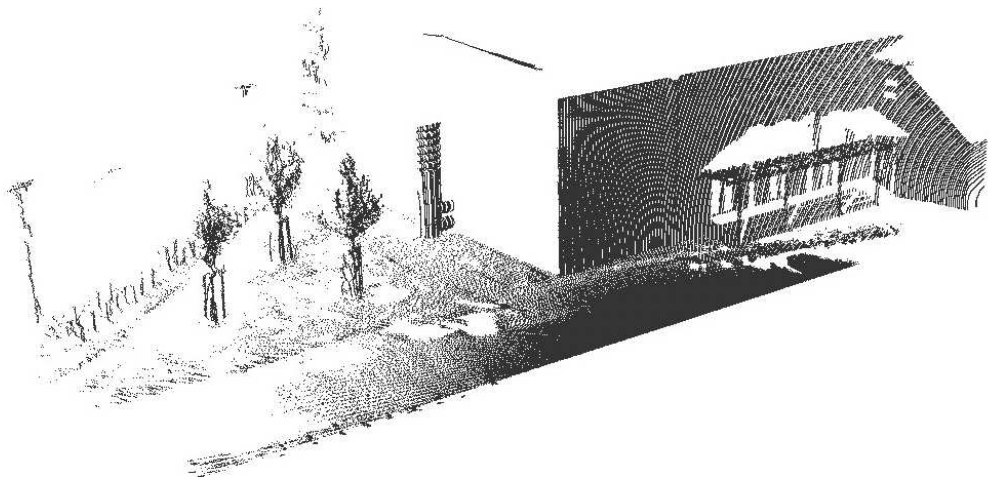

Abbildung 2.10: Szene einer urbanen Umgebung

Aufgrund der Datenakquisition wie sie in Kapitel 6.1.2 beschrieben wird, besteht das Problem, dass die maximale Messhöhe abhängig von der Entfernung zum gemessenen Objekt ist. Dies liegt in erster Linie daran, dass durch die Geometrie des Roboters der vertikale Scanwinkel (tilt) bei der Akquisition auf ca. 40 Grad begrenzt ist. Zusätzlich ist man auch selbst daran interessiert, diesen Winkel beschränkt zu halten, da es bei der Navigation mehr auf Punkte am Boden als auf Punkte in großer Höhe ankommt. Als Nebenprodukt kann man durch das Weglassen dieser hohen Punkte außerdem Scanzeit und Speicheraufwand sparen.

In Abbildung 2.11 sieht man, wie sich die Messentfernung direkt auf die Elevation Map auswirkt. Die horizontale Erhebung im rechten Teil der Abbildung ist die Wand aus Abbildung 2.10 und 2.12. Die Wand erscheint in der oberen Abbildung deutlich höher als in der unteren, die aus geringerer Entfernung aufgenommen wurde.

Desweiteren kann man anhand dieser Abbildung noch ein anderes Problem deutlich erkennen. Durch die Diskretisierung der Messpunkte in quadratische Gitterzellen werden Wände, die nicht parallel zu den Gitterstrukturen verlaufen, als zackige Gebilde dargestellt.

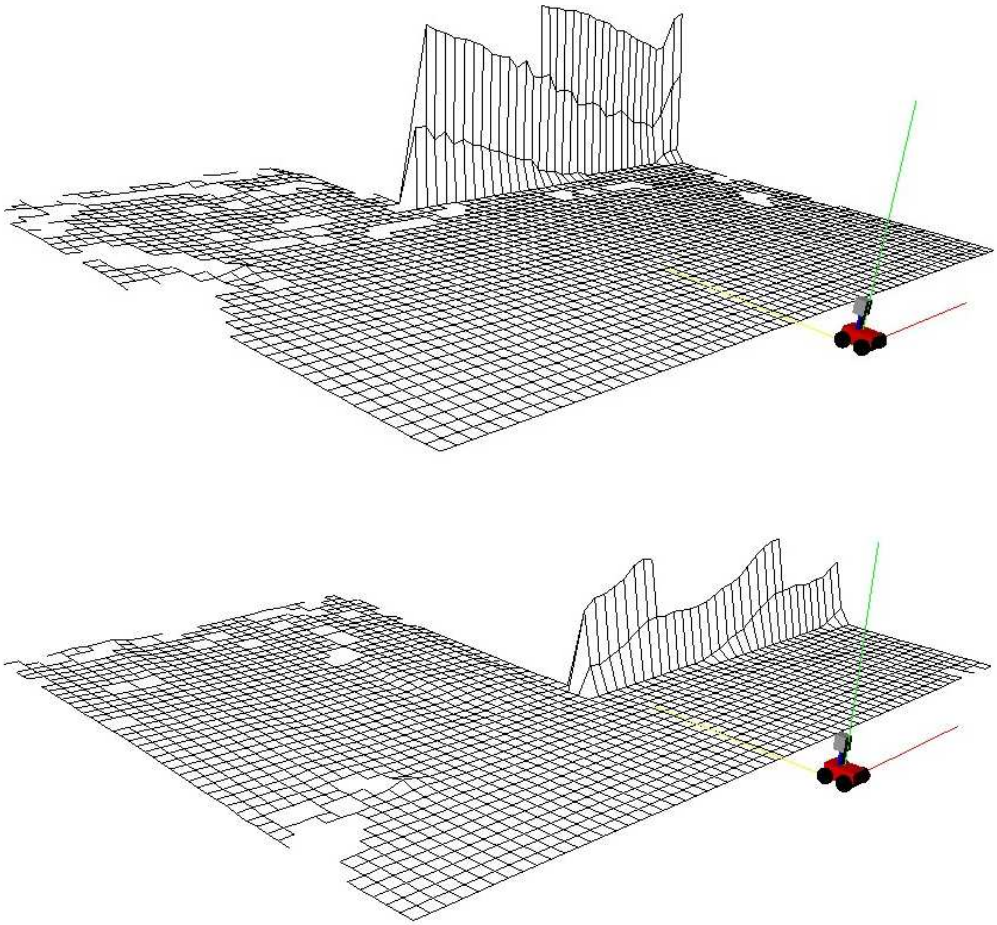

Abbildung 2.11: Eine Wand aus zwei verschiedenen Enfernungen

Der Diskretisierungsfehler und das Problem, keine eindeutige Höhe bestimmen zu können, haben auf die Navigation selbst keine große Auswirkung, da die Höhe eines Hindernisses keine Rolle spielt, sofern es als Hindernis erkannt wird. Allerdings treten dadurch bedingt Probleme beim Scan-Matching auf. Deshalb drängt man die unsichere Höhenmessung in den Hintergrund und klassifiziert die Zellen, die vertikale Objekte enthalten. Abbildung 2.12 zeigt die selbe Szene wie Abbildung 2.10. Die Punkte aus Zellen mit vertikalen Objekten wurden rot eingefärbt.

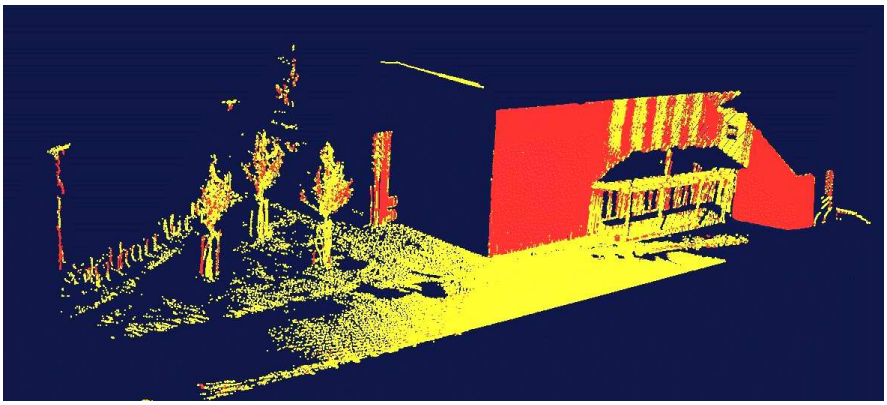

Abbildung 2.12: Zellen mit vertikalen Objekten rot eingefärbt

Zur Erkennung der Zellen mit vertikalen Objekten gibt es verschiedene heuristische Verfahren. Beispielsweise kann man die Varianz der Elevation Map nutzen und ab einem bestimmten Schwellwert auf ein vertikales Objekt schließen. Dieser Schwellwert sollte aus oben beschriebenen Gründen in Abhängigkeit zur Messdistanz und der Ausrichtung des Roboters gewählt werden. Eine andere Möglichkeit ist es, durch die Punkte, die in eine Zelle fallen, Ebenen zu legen und aufgrund der Ausrichtung dieser Ebenen vertikale Strukturen zu detektieren. In der Praxis arbeiten beide Verfahren relativ zuverlässig, wobei das ebenenbasierte Verfahren einen höheren Berechnungsaufwand erfordert, aber auch sehr kleine vertikale Objekte sicher erkennen kann. Dies ist beim varianzbasierten Verfahren nicht immer der Fall, was allerdings bei stark welligem und unebenem Gelände seine Vorteile hat. So werden Hügel und steile Böschungen in der Regel nicht als vertikale Objekte erkannt, was beim Ebenenbasierten Verfahren hingegen auftreten kann. Auf dem Campusgelände, auf dem sich sehr viele Gebäude befinden, hat sich das Ebenenverfahren als sehr praktikabel erwiesen, weshalb es hier auch verwendet wird. Steht jedoch die Erkennung von vertikalen Objekten in schwerem Gelände im Vordergrund, sollte man die Varianzinformation der Gitterzellen nutzten.

Zellen mit einer vertikalen Lücke

Zellen mit einer vertikalen Lücke stellen bei der Navigation im Gelände und vor allem bei der Navigation in urbanen Gebieten ein ernsthaftes Problem dar, wie aus Abbildung 2.5 direkt ersichtlich wird. Diese Zellen werden dadurch detektiert, dass man alle Punkte in einer Zelle nach ihrer z-Koordinate sortiert. Danach sucht man nach einem Zwischenraum, indem man die Distanz zwischen allen Nachbarpunkten in z-Richtung mit einem Schwellwert vergleicht. Dieser Schwellwert sollte mindestens der Höhe des Roboters entsprechen. Man speichert dann optimaler Weise immer die Höhe der unteren Grenze einer Lücke ab. In Abbildung 2.13 sieht man das Endpunktmodell aus Abbildung 2.10, indem die Zellen mit einer vertikalen Lücke blau eingefärbt wurden.

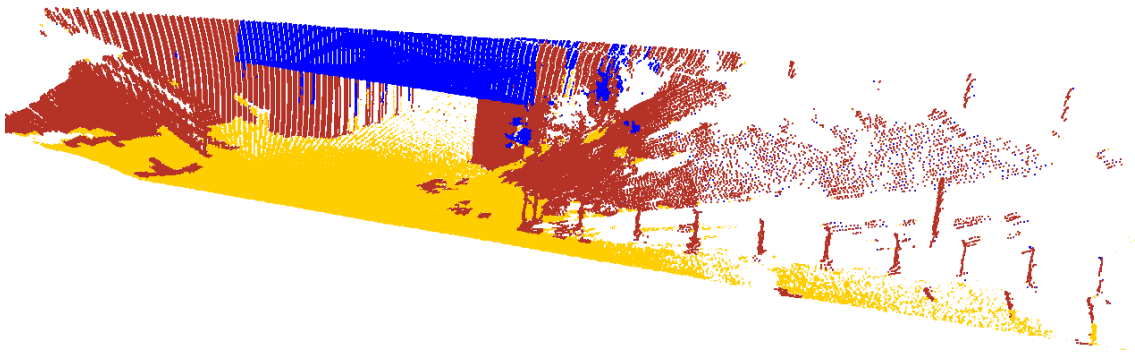

Abbildung 2.13: Zellen mit einer vertikalen Lücke

Falls man in einer so klassifizierten Zelle mehrere Lücken detektiert, muss man entscheiden, durch welche Lücke der Roboter eventuell hindurchfahren soll. Deshalb vergleicht man die Lücken anhand ihrer Zellparameter μ und σ. Genauer gesagt, berechnet man für jede der n von oben nach unten gefundenen Lücken L_i die Parameter μ_i und σ_i mit Hilfe der Punkte unterhalb dieser Lücke L_i und oberhalb der Lücke L_{i+1}. Man wählt danach diejenige Lücke L_i als relevante Lücke, fur die $\mu_i - \mu_{tr}$ minimal ist. Hier ist μ_{tr} der Zellparameter der nächstliegenden traversierbaren Zelle. μ_i und σ_i sind dann die Zellparameter dieser Zelle.

Abbildung 2.14 zeigt die Zellparameter μ_1 und μ_2 einer Zelle mit einer vertikalen Lücke. Die Punkte dieser Zelle sind rot markiert. In diesem Fall ist $\mu_{tr} = \mu_2$ was auch Abbildung 2.15 zeigt.

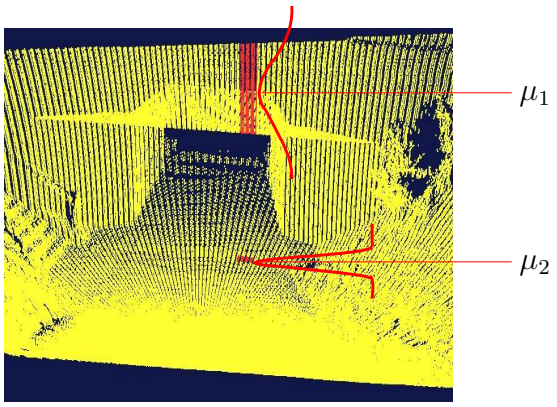

Abbildung 2.14: Zellparamter einer Zelle mit vertikalen Lücken

Abbildung 2.15 zeigt nun im Vergleich zu Abbildung 2.5, wie sich die Erweiterung der Elevation Map um die Klassifizierung der Zellen bemerkbar macht. In dieser extended Elevation Map ist der Weg unter der Brücke frei passierbar. Würde man die Brücke danach auf ihrer Oberseite befahren wollen, müsste man oben lediglich einen neuen Scan aufnehmen. Der Zellparameter der nächstgelegenen traversierbaren Zelle des aktuellen Scans wäre dann μ_{tr}. Somit wäre die Brücke auch von oben her traversierbar. Die Erweiterung der Elevation Map liefert also eine Möglichkeit, mit mehreren traversierbaren Ebenen in einer Zelle umgehen zu können.

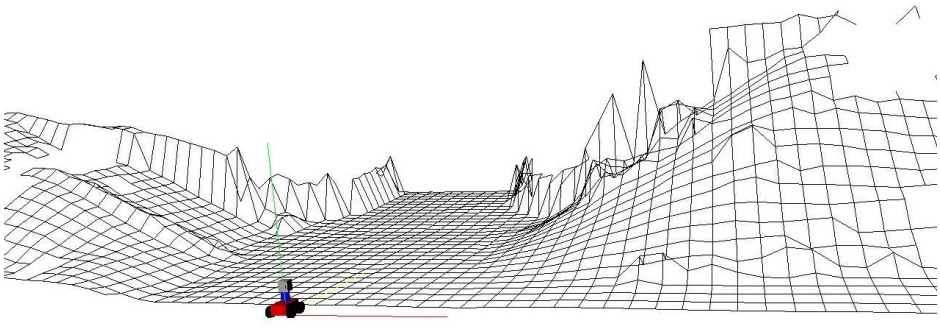

Abbildung 2.15: Zellen mit einer vertikalen Lücke

Zellen die von oben gesehen wurden

Diese Zellklasse unterscheidet sich derart von den anderen beiden, dass dies die einzigen nicht traversierbaren Zellen sind, bei denen die Höhenschätzung durch den Kalman Filter prinzipiell unabhängig vom Standpunkt des Roboters ist. Alle Punkte in dieser Zelle liegen unterhalb des Augpunktes[1] des Laser-Range-Finders. Sie sind also jederzeit vom Roboter aus sichtbar. Die Messungen dieser Zellen liefern immer, wenn man von Abschattungsproblemen und Sensorrauschen absieht, vergleichbare Zellparameter μ und σ.

[1] Ursprung des Laser-Koordinatensystems

KAPITEL 3

Registrierung, Kartenbau und Navigation

Um eine Karte eines unbekannten Gebietes zu erstellen, benötigt man in der Regel mehrere von einander unabhängige 3D-Scans. Um aus mehreren 3D-Scans ein komplettes Weltmodell zu erstellen, ist es notwendig zu wissen, welche Strecke der Sensor, der das 3D-Modell aufgenommen hat, zwischen den Messungen zurückgelegt hat. Die Odometrie am Fahrwerk eines Roboters ist hierfür zu ungenau und kann lediglich als Ausgangspunkt für genauere Matching-Algorithmen dienen. Im folgenden werden wir den ICP-Algorithmus vorstellen, den wir zum genaueren Matching verwendet haben.

3.1 ICP-Algorithmus

Besl und McKay [3, 9] entwickelten 1992 einen Algorithmus, der das nichtlineare Matching-Problem auf ein iteratives Punkt-Matching-Problem reduziert. Der ICP-Algorithmus berechnet die Transformationsparameter, um eine Punktmenge $P = \{p_1, ..., p_n\}$ auf ein vorgegebenes Modell $X = \{x_1, ..x_n\}$ abzubilden. Das Modell X kann aus Punkten, Liniensegmenten, parametrischen Kurven, impliziten Kurven, Dreiecken, parametrischen Dreiecken, parametrischen Oberflächen oder impliziten Oberflächen bestehen. Durch die Suche nach dem nächsten Nachbarn wird in jedem Iterationsschritt eine Paarung berechnet. Der ICP-Algorithmus hat somit den Vorteil, dass keine korre-

spondierenden Merkmale aus beiden Punktmengen segmentiert werden müssen und dadurch ein Vorverarbeitungschritt und eine mögliche Fehlerquelle wegfällt. Der Algorithmus versucht dann eine relative Position der beiden Punktmengen derart zu suchen, dass die Summe der quadratischen Fehler zwischen den Korrespondenzpaaren minimiert wird. Dieser Iterationsprozess wird solange fortgesetzt, bis die Verringerung des Fehlers unter eine Schranke fällt.

Seien nun zwei Punktmengen gegeben, die bestehende Punktmenge $X = \{x_1, ..x_n\}$ und die zu registrierende Punktmenge $P = \{p_1, ..., p_n\}$, wobei x_i und p_i zueinander korrespondierende Punkte sind. Gesucht ist die Rotation R und eine Translation t, die folgende Kostenfunktion minimiert:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2 \tag{3.1}$$

Der Schwerpunkt μ_p der aufgenommenen Punktmenge P und der Schwerpunkt μ_x der bestehenden Punktmenge X sind gegeben durch:

$$\mu_p = \frac{1}{N_p} \sum_{i=1}^{N_p} p_i \quad \text{und} \quad \mu_x = \frac{1}{N_x} \sum_{i=1}^{N_x} x_i$$

Um die Transformation zu berechnen, werden von allen Punkten der beiden Mengen X und P die entsprechenden Schwerpunkte μ_x und μ_p subtrahiert. Dies führt zu den $X' = \{x_i - \mu_x\} = \{x'_i\}$ und $P' = \{p_i - \mu_p\} = \{p'_i\}$.

$$\begin{aligned}
E(R,t) &= \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2 \\
&= \frac{1}{N_p} \sum_{i=1}^{N_p} ||(x'_i + \mu_x) - R(p'_i + \mu_p) - t||^2 \\
&= \frac{1}{N_p} \sum_{i=1}^{N_p} ||x'_i - Rp'_i - \underbrace{(t - \mu_x + R\mu_p)}_{=:t'}||^2 \\
&= \frac{1}{N_p} \left(\sum_{i=1}^{N_p} ||x'_i - Rp'_i||^2 - 2t' \sum_{i=1}^{N_p} (x'_i - Rp'_i) + \sum_{i=1}^{N_p} ||t'||^2 \right) \qquad (3.2)
\end{aligned}$$

Um die Summe zu minimieren, muss jeder einzelne Summand minimiert werden. Aus $\sum_{i=1}^{N_p} x'_i = 0$ und $\sum_{i=1}^{N_p} p'_i = 0$ folgt, dass der zweite Summand null

ist. Der dritte Summand hat ein Minimum für $t' = 0 \Leftrightarrow t = \mu_x + R\mu_p$. Daher folgt für die neue Kostenfunktion:

$$E(R,t) = \sum_{i=1}^{N_p} ||x'_i - Rp'_i||^2. \tag{3.3}$$

Da Rotationen die Länge von Vektoren nicht verändern, gilt die Gleichung $||Rp'_i||^2 = ||p'_i||^2$. Man kann also die Gleichung (3.3) folgendermaßen erweitern.

$$E(R,t) = \sum_{i=1}^{N_p} ||x'_i||^2 - 2\sum_{i=1}^{N_p} (x'_i - Rp'_i) + \sum_{i=1}^{N_p} ||p'_i||^2, \tag{3.4}$$

In dieser Gleichung ist lediglich der zweite Term rotationsabhängig. Um Gleichung 3.4 zu minimieren, muss man also den folgenden Ausdruck maximieren:

$$\sum_{i=1}^{N_p} (x'_i - Rp'_i) \tag{3.5}$$

An dieser Stelle ist es sinnvoll für die Repräsentation der Rotation Quaternionen zu benutzen. Ein Quaternion kann man sich entweder als Vektor mit vier Komponenten vorstellen, bestehend aus einem skalar und einem gewöhnlichen Vektor, oder als Komplexe Zahl mit drei Imaginärteilen. Quaternionen werden hier durch Punkte über den Symbolen gekennzeichnet. Aus der Rotation von komplexen Zahlen erhalten wir mit

$$\dot{q} = q_0 + iq_x + jq_y + kq_z$$

ein Quaternion mit dem Realteil q_o und drei Imaginärteilen q_x, q_y und q_z. Falls $q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1$, $q_0 \geq 0$ spricht man von einem Einheitsquaternion. Multiplikationen von Quaternionen sind als Multiplikationen ihrer Komponenten definiert, wobei gilt:

$$\begin{array}{lll} i^2 = -1, & j^2 = -1, & k^2 = -1 \\ ij = k, & jk = i, & ki = j \end{array}$$

und

$$ji = -k, \quad kj = -i \quad ik = -j$$

Falls

$$\dot{r} = r_0 + ir_x + jr_y + kr_z,$$

erhält man

$$\begin{aligned}\dot{r}\dot{q} &= (r_0q_0 - r_xq_x - r_yq_y - r_zq_z) \\ &\quad +i(r_0q_x + r_xq_0 + r_yq_z - r_zq_y) \\ &\quad +j(r_0q_y - r_xq_z + r_yq_0 + r_zq_x) \\ &\quad +k(r_0q_z + r_xq_y - r_yq_x + r_zq_0)\end{aligned}$$

Das Produkt $\dot{q}\dot{r}$ hat eine ähnliche Form, wobei sechs der Zeichen vertauscht sind, was problemlos nachvollzogen werden kann. Allgemein gilt $\dot{r}\dot{q} \neq \dot{q}\dot{r}$. Das Produkt von zwei Quaternionen kann daher ebenso als Produkt einer 4×4 Matrix mit einem vierdimensionalen Vektor geschrieben werden. Man kann dabei entweder das erste oder das zweite Quaternion wie folgt als 4×4 Matrix notieren:

$$\dot{r}\dot{q} = \underbrace{\begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & -r_z & r_y \\ r_y & r_z & r_0 & -r_x \\ r_z & -r_y & r_x & r_0 \end{bmatrix}}_{=:Q} \dot{q}$$

oder

$$\dot{q}\dot{r} = \underbrace{\begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{bmatrix}}_{=:\bar{Q}} \dot{q}$$

Das Skalarprodukt zweier Quaternionen ist die Summe der Produkte der korrespondierenden Komponenten:

$$\dot{p} \cdot \dot{q} = p_0q_0 + p_xq_x + p_yq_y + p_zq_z$$

Das Normquadrat eines Quaternions ist das Skalarprodukt des Quaternions mit sich selbst:

$$||\dot{q}||^2 = \dot{q} \cdot \dot{q} \; .$$

Das konjugierte Quaternion erhält man durch die Negation des Imaginärteils:

$$\dot{q}^* = q_0 - iq_x - jq_y - kq_z$$

Man kann zeigen, dass die Suche nach der Rotationsmatrix in Gleichung 3.5 der Suche nach einem Einheitsquaternion $\dot{q}$ entspricht, das den folgenden Term maximiert:

$$\sum_{i=1}^{N_p} \dot{x'}_i \cdot (\dot{q}\dot{p'}_i\dot{q}^*) = \sum_{i=1}^{N_p} (\dot{q}\dot{x'}_i) \cdot (\dot{p'}_i\dot{q})$$

Sei $x'_i = (x'_{x,i}, y'_{x,i}, z'_{x,i})^T$ und $p'_i = (x'_{p,i}, y'_{p,i}, z'_{p,i})^T$; dann gilt

$$\dot{q}\dot{x'_i} = \begin{bmatrix} 0 & -x'_{x,i} & -y'_{x,i} & -z'_{x,i} \\ x'_{x,i} & 0 & z'_{x,i} & -y'_{x,i} \\ y'_{x,i} & -z'_{x,i} & 0 & x'_{x,i} \\ z'_{x,i} & y'_{x,i} & -x'_{x,i} & 0 \end{bmatrix} = \bar{Q}_x\dot{q}$$

wobei

$$\dot{x'_i}\dot{q} = \begin{bmatrix} 0 & -x'_{p,i} & -y'_{p,i} & -z'_{p,i} \\ x'_{p,i} & 0 & -z'_{p,i} & y'_{p,i} \\ y'_{p,i} & z'_{p,i} & 0 & -x'_{p,i} \\ z'_{p,i} & -y'_{p,i} & x'_{p,i} & 0 \end{bmatrix} = Q_p\dot{q}$$

Die zu maximierende Summe kann nun in folgender Form geschrieben werden:

$$\begin{aligned} \sum_{i=1}^{N_p} (\dot{q}\dot{x'}_i) \cdot (\dot{p'}_i\dot{q}) &= \sum_{i=1}^{N_p} (\bar{Q}_x\dot{q}) \cdot (Q_p\dot{q}) \\ &= \sum_{i=1}^{N_p} (\dot{q}^T\bar{Q}_x^TQ_p\dot{q}) \\ &= \dot{q}^T \underbrace{\left(\sum_{i=1}^{N_p} \bar{Q}_x^TQ_p \right)}_{=N} \dot{q} \end{aligned}$$

An dieser Stelle ist es naheliegend die 3×3 Matrix M einzuführen mit:

$$M = \sum_{i=1}^{N_p} (x'_i - p'^T_i) = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix}$$

wobei gilt: $S_{xx} = \sum_{i=1}^{N_p} x'_{x,i}x'_{p,i}$, $S_{xy} = \sum_{i=1}^{N_p} x'_{x,i}y'_{p,i}$, , usw. Dann gilt

$$N = \begin{bmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}+S_{zy} & S_{zx}+S_{xz} & S_{xy}+S_{yx} \\ S_{yz}+S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}+S_{xz} & S_{xy}+S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}+S_{yx} & S_{yz}+S_{zy} & S_{zx}+S_{xz} & -S_{xx}-S_{yy}+S_{zz} \end{bmatrix}$$

Man kann zeigen - worauf ich an dieser Stelle verzichten möchte - , dass genau das Einheitsquaternion, das $\dot{q}^T N \dot{q}$ maximiert, dem größten Eigenvektor der Matrix N entspricht. Aus diesem Quaternion kann man mit dem folgenden die Rotationsmatrix berechnen:

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_3 q_0) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & q_0 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.6}$$

Kennt man die Rotation, ergibt sich für die Translation:

$$t = \mu_x + R\mu_p \tag{3.7}$$

Der ICP Algorithmus berechnet iterativ das Minimum von Gleichung 3.1. In jedem Iterationsschritt, wählt der Algorithmus zu jedem Punkt der zu registrierenden Punktmenge $P = \{p_i\}$ einen nächsten Nachbarn aus der bestehenden Punktmenge $X = \{x_i\}$ und berechnet die Transformation, welche die Gleichung 3.1 in Bezug auf Gleichung 3.6 und 3.7 minimiert.

In der Praxis konvergiert der ICP Algorithmus allerdings häufig zu lokalen Minima. Um dies zu verhindern, sind einige Zusatzheuristiken notwendig, was allgemein als Trimmed-ICP-Algorithmus [5] bekannt ist. Bei jeder Version des ICP-Algorithmus ist es prinzipiell von Vorteil, initiale Transformationsparameter zu besitzen. Dadurch wird das Matchen schneller und die Gefahr, in lokalen Minima festzuhängen, geringer.

Auf einige der Möglichkeiten, wie das Verhalten des ICP Algorithmus verbessert werden kann, werden wir in den Abschnitten 3.2 und 3.3 eingehen.

3.2 Effiziente Versionen des ICP-Algorithmus

"Der ICP-Algorithmus führt immer zu perfektem Matching zweier Punktmengen." Im Prinzip ist diese Aussage richtig. Man muss allerdings zwei erhebliche Einschränkungen hinzufügen:

- Der ICP-Algorithmus findet nur dann das globale Minimum, falls eine optimale Nächste-Nachbarn-Suche zur Bestimmung der korrespondierenden Punktepaare vorrausgeht, und damit ausgeschlossen werden kann, dass keine "falschen" Paare gebildet werden. Mit anderen Worten: alle Punkte, die für den ICP-Algorithmus verwendet werden, sind sowohl in der bestehenden Punktmenge $X = \{x_i\}$ als auch in der zu registrierenden Punktmenge $P = \{p_i\}$ enthalten.

- Führt man den ICP-Algorithmus in seiner ursprünglichen Form aus, benötigt man viel Zeit zur Bestimmung der Punktkorrespondenzen, was dazu führt, dass der Algorithmus für große Punktmengen, die in der Regel vorliegen, nicht online verwendbar ist.

Rusinkiewicz und Levoy [23] haben aus diesem Grund einige effizientere Varianten des ICP-Algorithmus beschrieben und miteinander verglichen. Rusinkiewicz und Levoy analysieren fünf Ansatzmöglichkeiten, den Algorithmus hinsichtlich Konvergenz und Laufzeit effizienter zu machen: die Auswahl der Punkte, das Matching der Punkte, die Gewichtung der Paare, das Zurückweisen von Paaren und die Wahl der Metrik zur Minimierung. Hierbei werden wir besonderes Augenmerk auf die ersten vier Ansätze legen, da von den bei Rusinkiewicz und Levoy [23] aufgelisteten Metriken für diese Arbeit höchstens die Wahl einer Farbmetrik relevant wäre, sofern man den Laser-Range-Finder mit einer Kamera kalibrieren würde. Zusätzlich möchten wir in Abschnitt 3.2.5 noch kurz die Idee des Trimmed ICP-Algorithmus [5] betrachten.

3.2.1 Auswahl der Punkte

Der Standard-ICP-Algorithmus von Besl und McKay [3, 9] benutzt immer alle vorhandenen Punkte. Wie man sich nur unschwer ausdenken kann, wird diese Herangehensweise bei großen Punktmengen äußerst ineffizient. Andere Verfahren zur Auswahl der Punkte sind:

- Einmaliges zufälliges Auswählen aller möglichen Punkte [28].
- Wiederholtes zufälliges Auswählen von Punkten bei jedem Iterationsschritt [14]
- Verwendet man Farbmetriken zur Minimierung, so kann man Punkte mit hohem Intensitätsgradienten bevorzugt auswählen [29].
- Bei der Auswahl der Punkte, zu denen korrespondierende Punkte gesucht werden, kann man entweder Punkte aus einer der beiden Mengen wählen oder aus beiden [8].

3.2.2 Matching der Punkte

Dieser Teil des ICP-Algorithmus ist einer der wichtigsten, da er sehr großen Einfluss auf die Konvergenz bezüglich Qualität und Geschwindigkeit hat.

Das kommt daher, dass die Qualität des ICP-Algorithmus von Besl und McKay[3, 9] erheblich davon abhängt, wie die korrespondierenden Punktepaare ausgewählt werden. Als Vorraussetztung für die mathematische Herleitung des Algorithmus wird von der optimalen Wahl ausgegangen, die den quadratischen Fehler minimiert.

- Diese Berechnung kann beispielsweise durch die Verwendung eines k-d trees beschleunigt werden [25].
- Chen [4] zeigt, dass zum Bestimmen eines Korrespondenzpaares von einem Punkt aus in Richtung der Punktnormalen gesucht werden kann. Dies hat den Namen "normal shooting".
- Weiter ist es möglich die Punkte der einen Menge über den Augpunkt der anderen Punktemenge auf diese zu projizieren [7, 17]. Dies nennt man auch "reverse calibration".
- Diese obengenannten Varianten beschränken sich als auf eine bestimmte Metrik. Andere Metriken sind aber durchaus denkbar, wie zum Beispiel Farbmetriken oder Metriken, die auf dem Winkel zwischen zwei Normalenvektoren beruhen.

3.2.3 Gewichtung der Paare

Sind ersteinmal korrespondierende Punktepaare in ausreichender Zahl berechnet worden, ergibt sich selbstverständlich noch die Möglichkeit diese zu gewichten. Man kann dies im Großen und Ganzen auf vier verschiedene Art und Weisen tun:

- Konstante Gewichtung aller Punktepaare.
- Paaren weiter entfernter Punkte ein geringeres Gewicht

$$Weight = 1 - \frac{Dist(p_1, p_2)}{Dist_{max}} \tag{3.8}$$

  zuzuordnen entspricht dem Ansatz, nur Punkkorrespondenzen zu erlauben, deren Distanz unter einem Grenzwert liegt. Man hat den Vorteil, dass immer Korrespondenzen vorhanden sind, wenn auch mit geringerem Gewicht.

- Gewichtung bezüglich Normalenvektoren:

$$Weight = n_1 \cdot n_2 \tag{3.9}$$

  Außerdem kann man auch eine Gewichtung nach Farbwertdifferenzen vornehmen [8], worauf wir in der vorliegenden Arbeit aber nicht genauer eingehen möchte, da keine Kamera verwendet wurde.

- Zuletzt bleibt noch die Möglichkeit eine Gewichtung bezüglich des Sensorrauschens vorzunehmen. Bei einem SICK Laser-Range-Finder verhält sich der Fehler in etwa linear zur gemessenen Entfernung. Man muss also entfernte Messungen weniger gewichten als nahe. Experimentell hat sich eine Gewichtung nicht als vorteilhaft erwiesen, da durch die abnehmende Punktdichte die weiter entfernten Messungen durch eine geringere Gewichtung unterrepräsentiert werden.

3.2.4 Zurückweisung von Paaren

Da man bei der Bestimmung der Korrespondenzpunktepaare nicht selten erhebliche Fehler macht, liegt es auf der Hand, nicht jedes der gefundenen Punktepaare zu verwenden. Zur Modifikation des ICP-Algorithmus gibt es in diesem Bereich im wesentlichen fünf Ansätze:

- Beschränkung des Abstandes von korrespondierenden Punkten durch eine obere Schranke.

- Zurückweisen der schlechtesten n% der bestimmten Punktepaare [22].

- Zurückweisen der Punktepaare, deren Distanz größer als ein Vielfaches der Standardabweichung ist [14].

- Zurückweisen von Paaren, die nicht konsistent zu benachbarten Paaren sind. D.h. es werden zwei Paare (p_1, p_2) und (q_1, q_2) genau dann zurückgewiesen, falls:

$$d_{max} \leq |Dist(p_1, p_2) - Dist(q_1, q_2)| \tag{3.10}$$

  wobei d_{max} eine obere Schranke darstellt [6].

- Zurückweisen von Punkten am Rand der Punktmenge [28]

3.2.5 Trimmed ICP-Algorithmus

Der Trimmed ICP (TrICP) [5] ist eine Spezialvariante des ICP, die einige Vorteile besitzt: TrICP ist schnell, anwendbar bei Überlappungen der Punktmengen, die kleiner als 50% sind, robust gegenüber Messfehlern und hat einfach zu bestimmende Parameter. Falls eine Überlappung von 100% besteht, ist TrICP identisch zu ICP.

Es seien jetzt also zwei Punktmengen gegeben $X = \{x_1, ..x_n\}$ und $P = \{p_1, ..., p_m\}$, für die gilt : $m \neq n$. Die beiden Punktemengen sind also unterschiedlich groß. Also ergibt sich für die Zahl der Punkte, für die es einen Korrespondenzpunkt gibt: $n_0 = \zeta n$. Den Wert ζ nennt man auch die Überlappung (overlap). Kennt man diese Überlappung, ist das Vorgehen denkbar einfach und sehr effektiv:

1. Bestimmen des Overlap-Parameters ζ
2. Bestimmen der Korrespondierenden Paare
3. Sortieren der Paare bezüglich der Distanz und Wahl der ζ besten Paare
4. Minimieren des Fehlers für diese Paare.

Der Ansatz funktioniert allerdings nur, wenn ζ bekannt ist oder nicht zu sehr von der Realität abweicht[1]. Schon bei geringen Abweichungen zieht es oft schon starke Verluste in der Qualität des Matchings nach sich. Ist der Overlap unbekannt, kann man ihn dadurch berechnen, dass man die folgende Formel für ein $\lambda \geq 0$ minimiert:

$$\psi(\zeta) = \exp^{\zeta} \zeta^{-(1+\lambda)} \tag{3.11}$$

3.3 ICP für Extended Elevation Maps

Wenn es darum geht, effektiv und sicher zu navigieren, sind Elevation Maps eine sehr gute und elegante Möglichkeit zur Darstellung von Outdoor-Umgebungen. Bleibt die Frage, ob man Extended Elevation Maps effektiv auch zur Registrierung neuer Maps in einer bestehenden verwenden kann. Dabei sollte es möglich sein die Eigenschaften der Extended Elevation Maps zu nutzen, um den ICP-Algorithmus von Besl und McKay [3, 9] effizienter zu machen. Wie

[1] Persönliche Korrespondenz mit Rudolph Triebel

das funktioniert und welche Vor -und Nachteile sich aus den Elevation Maps ergeben, zeigen wir im folgenden Abschnitt.

Da man bei Elevation Maps nicht immer eine sichere Aussage über die tatsächliche Höhe in einem Punkt treffen kann, ist es sinnvoll, das 3D-Matching durch ein 2D-Matching mit anschliessender "3D-Korrektur" zu ersetzen. D.h. man geht beim Matchen durch den ICP an sich von einer ebenen 2D-Translation und Rotation aus und korrigiert danach die Transformationsparameter. Dies erreicht man dadurch, dass man das Wissen aus der bestehenden Elevation Map nutzt, um die dreidimensionale Ausrichtung der zu registrierenden Elevation Map zu berechen. Man "matched" also unter der Annahme, dass die dreidimensionale Ausrichtung des Roboters zu jeder Zeit bekannt ist. Durch das Matchen mit dem ICP wird die Position des Roboters in der bestehenden Elevation Map berechnet, von wo aus die nächste Elevation Map erstellt wurde. Man kann nun durch die Berechnung der Roboternormalen n_i und der Roboterhöhe h_i die Rotationsparameter R_i und t_i für die 3D-Korrektur bestimmen. Die Translation ergibt sich aus:

$$t_i = (h_1 - h_{i-1})n_{i-1} \tag{3.12}$$

Die Rotation um die Achse r_i mit dem Winkel φ_i ergibt sich folgendermaßen:

$$r_i = n_i \times n_{i-1} \tag{3.13}$$

$$\varphi_i = \arccos\left(n_i \cdot n_{i-1}\right) \tag{3.14}$$

3.3.1 Wahl der Punkte

Eine Elevation Map besteht oftmals aus 100.000 bis 200.000 Punkten. Deshalb ist die richtige Auswahl der Punkte für das Gelingen der Registrierung entscheidend.

Die bei Rusinkiewicz und Levoy [23] und durch den TrICP[5] aufgelisteten Ansatzpunkte zur Verbesserung des ICP sind bei den Elevation Maps selbstverständlich auch gegeben. Man könnte natürlich so vorgehen, dass man ca. 5.000 bis 10.000 Punkte zufällig auswählt und zu diesen Punkten korrespondierende Punkte sucht. Dadurch wird das Suchproblem der Nächsten-Nachbar-Suche allerdings bereits so groß, dass man bei vielen Iterationen mit einer erheblichen Laufzeit rechnen muss.

Besser ist daher, zur Wahl der Punkte die Eigenschaften der Extended Elevation Map zu nutzen. Man wählt zum Matching also nur die nichttraversierbaren, klassifizierten Zellen aus (die Farben in Klammern beziehen sich auf Abbildung 3.1):

- Zellen mit vertikalen Lücken,wie z.B. Fenster, Brücken oder Türrahmen(blau)
- Zellen mit vertikalen Objekten(rot)
- Zellen die von oben gesehen wurden(grün)

Man wählt daher zufällig nur wenige Punkte (typischer Weise im einstelligen Bereich) aus diesen Zellen der beiden Elevation Maps und erhält somit die Punkte für das 2D-Matching durch Ignorieren der dritten Dimension der zufällig gewählten Punkte. Diese Punktwahl hat den entscheidenden Vorteil, dass man nur Punkte zum Matchen verwendet, die bezüglich der Translation und Rotation relevant sind. Die Datenassoziation zwischen zwei Scans verbessert sich dadurch erheblich, was das Matching effizienter und robuster macht. Im Gegensatz zur Standardfehlerfunktion des ICP (3.1) schlagen wir die folgende zu minimierende Fehlerfunktion vor:

$$D = \underbrace{\sum_{n=1}^{N} \|p_n - Rp'_n - t\|}_{2D\ (vertikale\ \ Objekte)} + \underbrace{\sum_{m=1}^{M} \|q_n - Rq'_n - t\|}_{2D\ (vertikale\ \ Lücken)} + \underbrace{\sum_{k=1}^{K} \|r_k - Rr'_k - t\|}_{3D\ (von\ \ oben\ \ gesehene\ \ Objekte)} \quad (3.15)$$

In Abschnitt 4.2 werden wir zeigen, dass (3.15) zu besserer Registrierung führt als (3.1).

Abbildung 3.1 zeigt die Rohdaten einer 3D-Punktmenge auf der linken und die zufällig daraus gewählten Punkte auf der rechten, zu denen nun korrespondierende Punkte einer zweiten Punktmenge gesucht werden können.

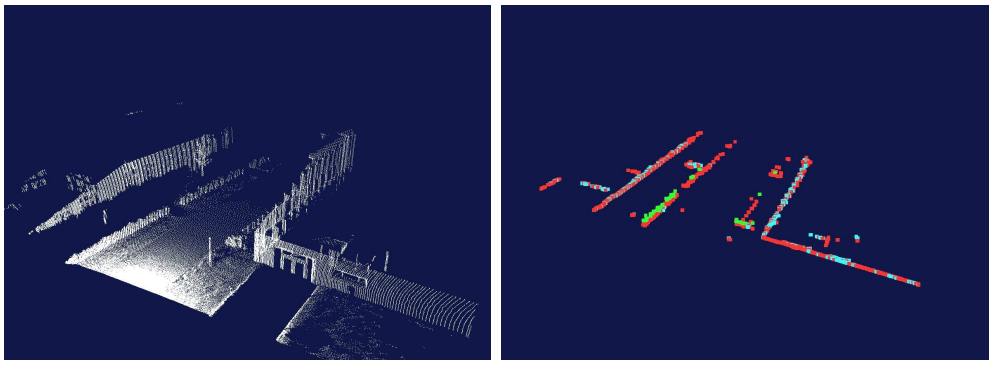

Abbildung 3.1: Rohdaten und Punkte zum Matchen

Im Vergleich zur normalen zufälligen Punktauswahl erhält man auf diese Weise weniger aber zum Matchen zweier Punktmengen ausdrucksstärkere Punkte. Die Zahl der Punkte ist allerdings von der Umgebung abhängig und kann bei einer Gittergröße von 30 × 40 Metern zwischen 0 und 5000 schwanken, wobei der schlechteste Fall von 0 Punkten in den Experimenten nicht aufgetreten ist.

3.3.2 Berechnung des Überlappungsbereichs

Die Berechnung des Überlappungsbereichs, die für den TrICP notwendig ist, ist durch die Wahl der Elevation Map als Datenstruktur überflüssig geworden. Da man sich auf Matching in 2D beschränkt, kann man für jede Gridzelle prüfen, ob die Punkte, die sie enthält, auch in der zu registrierenden Elevation Map enthalten sind, wie man aus Abbildung 3.2 direkt erkennen kann:

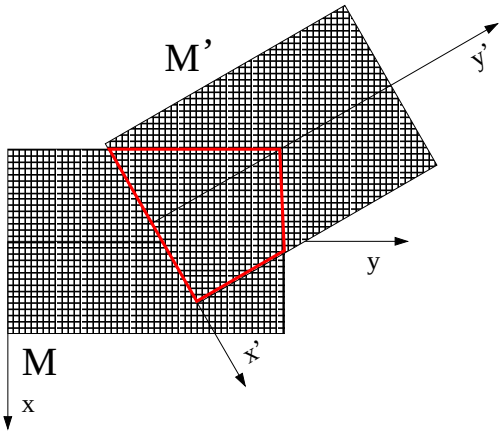

Abbildung 3.2: Overlap zweier Elevation Maps

Um für die Zellen der bereits registrierten Elevation Map M herrauszufinden, ob sie innerhalb der roten Begrenzung liegen, transformiert man die Koordinaten der Zellenmittelpunkte durch das Inverse der initialen, euklidischen Bewegung, die sich aus den Odometrieparametern des Roboters ergibt und erhält damit die Koordinaten der Zelle im Koordinatensystem der Elevation Map M'. Nun kann man für die so transformierten Punkte prüfen, ob sie innerhalb der Grenzen der zu registrierenden Elevation Map M' liegen. Für die zu registrierende Elevation Map M' verwendet man logischer Weise dann einfach direkt die Transformation, die durch die Odometrie gegeben ist, um eine Vorauswahl der Zellen vorzunehmen.

Man kann dies selbstverständlich vor jedem ICP-Schritt wiederholen, was aber im allgemeinen nicht notwendig ist, da sich die möglichen Zellen zwischen zwei Iterationsschritten meist nur sehr wenig ändern.

Sind die Transformationsparameter der Odometrie durch R_{od} und t_{od} gegeben, muss man zwei Koordinatentransformationen vornehmen,

$$\begin{pmatrix} x'_p \\ y'_p \end{pmatrix} = R_{od}^{-1} \left[\begin{pmatrix} x_p \\ y_p \end{pmatrix} - t_{od}^{-1} \right] \tag{3.16}$$

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = R_{od} \begin{pmatrix} x'_p \\ y'_p \end{pmatrix} + t_{od} \ , \tag{3.17}$$

wobei (x_p, y_p) und $\left(x'_p, y'_p\right)$ den selben Punkt P in den Koordinatensystemen der beiden Elevation Maps M und M' darstellen.

3.3.3 Matching der Punkte und Zurückweisen von Paaren

Für die Registrierung muss gelten, dass die Punkte eines Paares aus der selben Zellenklasse gewählt sein werden. Daher verwendet man bei der Nächste-Nachbarn-Suche drei k-d trees [25], um das Matching zu beschleunigen. Für jede Klasse von relevanten Punkten, baut man einen eigenen k-d Tree auf.

Das Zurückweisen von Paaren ist notwendig und gefährlich zugleich. Weist man keine Paare zurück, läuft man Gefahr, in ein lokales Minimum zu geraten. Weist man zu viele Paare zurück, beispielsweise die schlechtesten n%, riskiert man eine wenig aussagekräftige Kovarianzmatrix, weil man zu wenige Paare erhält. Macht man die Zurückweisung von der Distanz abhängig, muss man darauf achten, dass nicht zu viele Paare entfernt werden. Dies kann beispielsweise passieren, wenn der Anfangsfehler größer als die obere Schranke für die Distanz ist.

Weil sich aus den Elevation Maps direkt der Overlap ergibt, macht es wenig Sinn, eine Zurückweisungsstrategie wie beim TrICP zu wählen. Um große Odometriefehler ausgleichen zu können, muss die obere Schranke für die Distanz innerhalb eines Paares sehr hoch sein. Um ein genaues Matching zu erreichen, ist es notwendig diese obere Schranke gering zu halten. Deshalb liegt es auf der Hand eine dynamische Schranke δ für das Zurückweisen "schlechter Punktkorrespondenzen" zu konstruieren.

Zum Berechnen dieser Zurückweisungsschranke benötigt man einen Startwert δ_0, eine untere Schranke δ_{min} und einen konstante, garantierte Verringerung des zulässigen Fehlers α, die nach jedem Iterationsschritt subtrahiert wird. Die Abnahme des durchschnittlichen quadratischen Fehlers vom i-ten zum $(i-1)$-ten Iterationsschritt wird mit $\Delta_{err_i} = \psi \frac{err_{i-1}}{\Delta_{err_{i-1}} + err_{i-1}}$ berechnet. Zur Skalierung verwendet man eine Konstante ψ. Es ergibt sich somit für die Zurückweisungsschranke folgende rekursive Formel:

$$\delta_i = \begin{cases} \delta_{i-1}\Delta_{err_i} - \alpha & \text{falls } \delta_i \leq \delta_{min}, \\ \delta_{min} & \text{sonst} \end{cases} \tag{3.18}$$

Zu Beginn des iterativen Matchingprozesses ist die Obergrenze für den Fehler also höher als am Ende. Somit kann der Algorithmus auch große Registrierungsfehler ausgleichen. Liegt ein ehr geringer Anfangsfehler vor, so ändert sich der Fehler mit großem δ_i in jedem Iterationsschritt nicht sehr stark. Um trotzdem eine gute Konvergenz zu garantieren, benutzt man die Konstante α, um die Zurückweisungsschranke garantiert absenken zu können. Man kann diesen Ansatz in gewissem Sinne mit einem simulated annealing Ansatz vergleichen.

3.4 Inkrementelle Registrierung

Die inkrementelle Registrierung wird zum Bau großer Karten verwendet, die aus vielen Scans bestehen. Dabei startet der Roboter im Ursprung des Koordinatensystems mit dem ersten Scan. Für jeden weiteren Scan wird mit Hilfe des ICP die relative Position zum Vorgängerscan berechnet und die Transformationsparameter T_i abgespeichert. Die Roboterposition P_{r_i} beim i-ten Scan ist also:

$$P_{r_i} = T_i(P_{r_{i-1}}) \tag{3.19}$$

Um die inkrementelle Registrierung des i-ten Scans ausführen zu können, muss man vor der Registrierung die Translations- und Rotationsparameter des Scans im Roboterkoordinatensystem in das Koordinatensystem des (i-1)-ten Scans transformieren. R_{odo_i} bezeichnet die Rotationsmatrix, welche die globale Orientierung des i-ten Scans bezüglich des Odometriekoordinatenssytems des Roboters beschreibt. Die Größen t_i und R_i beschreiben die inkrementellen Transformationsparameter T_i:

$$t_i = R^{-1}_{odo_{i-1}} \left(t_{odo_i} - t_{odo_{i-1}} \right) \tag{3.20}$$

$$R_i = R^{-1}_{odoi-1} R_{odo_i} = R^{T}_{odoi-1} R_{odo_i} \tag{3.21}$$

Diese Transformationsparameter dienen beim Scan-Matching als initiale Konfiguration. Dieses Wissen ermöglicht die Registrierung eines Scans in seinem Vorgänger. Der inkrementelle Registrierungsalgorithmus lautet dann:

Algorithm 1 Inkremeteller Registrierungsalgorithmus

Input: n hintereinander aufgenommene 3D-Scans $S_1, ..., S_n$ mit Odometriemessung odo_{x_i}, odo_{y_i} und odo_{θ_i} für jeden Scan S_i

i = 1, $odocorr_x = odocorr_y = odocorr_\theta = 0$

for i $\leq$ n **do**

register(S_i, S_{i-1})

//berechne die Korrektur der Odometriedaten

$odocorr_{x_i}$:= xcorrection(S_i, S_{i-1})

$odocorr_{y_i}$:= ycorrection(S_i, S_{i-1})

$odocorr_{\theta_i}$:= θcorrection(S_i, S_{i-1})

//Aktualisiere die Summe aller Odometriekorrekturen

$odocorr_x := odocorr_x + odocorr_{x_i}$

$odocorr_y := odocorr_y + odocorr_{y_i}$

$odocorr_\theta := odocorr_\theta + odocorr_{\theta_i}$

//Aktualisiere Position des Scans:

$odo_{x_i} := odo_{x_i} + odocorr_x$

$odo_{y_i} := odo_{y_i} + odocorr_y$

$odo_{\theta_i} := odo_{\theta_i} + odocorr_\theta$

end for

Abbildung 3.3, 3.4 , 3.5 und 3.6 zeigen die Registrierung eines Scans in seinem Vorgängerscan nach 0, 5, 10 und 30 ICP-Schritten. Wie man sehen kann tritt nach dem 10-ten Iterationsschritt bereits keine sichtbare Verbesserung mehr auf. Die Abbildungen zeigen auf der linken Seite die zu registrierenden Punktmengen und auf der rechten Seite die für das Matching extrahierten Punkte der 3 relevanten Zellklassen. Abbildung 3.3 zeigt deutlich den Odometriefehler und damit die Notwendigkeit, die Registrierung durchzuführen.

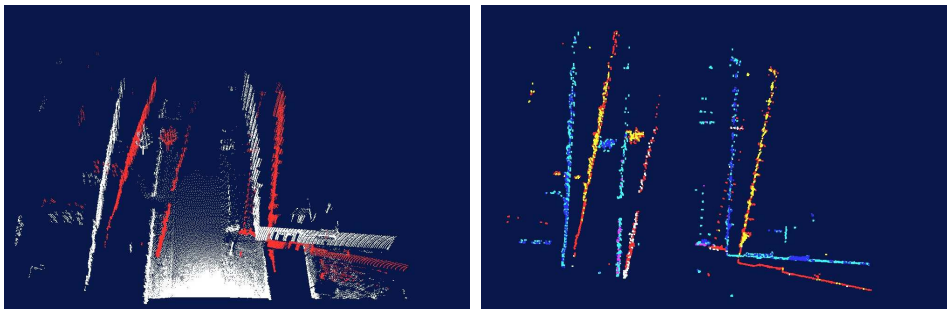

Abbildung 3.3: Registrierung nach 0 Iterationen

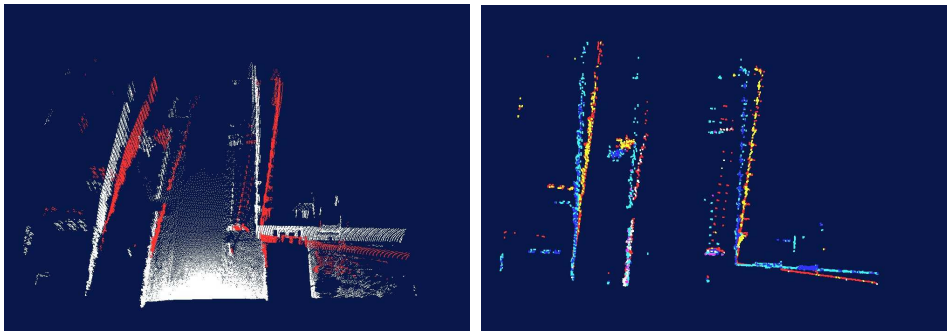

Abbildung 3.4: Registrierung nach 5 Iterationen

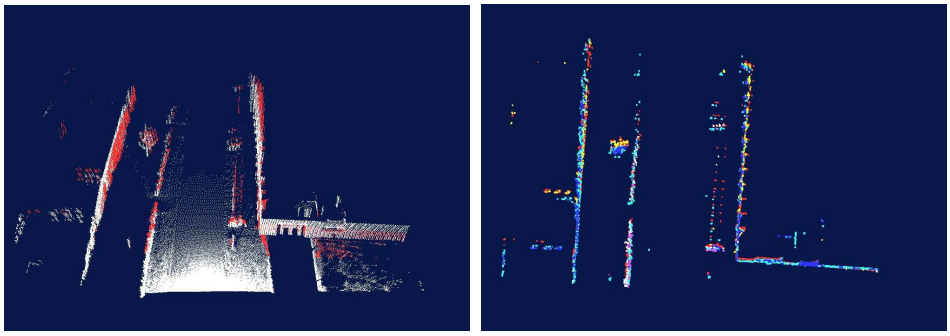

Abbildung 3.5: Registrierung nach 10 Iterationen

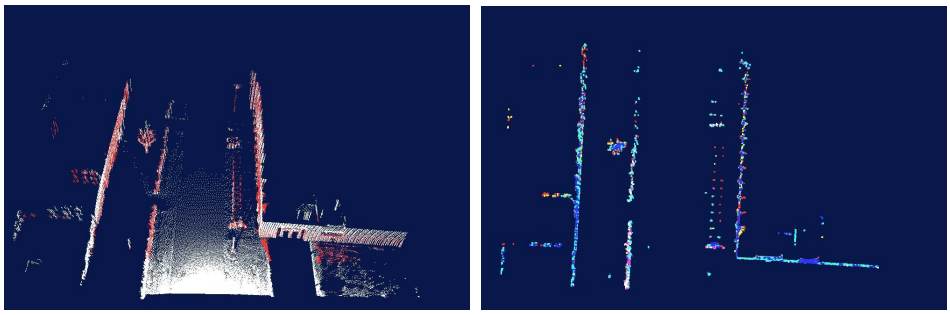

Abbildung 3.6: Registrierung nach 30 Iterationen

3.5 Navigation

3.5.1 Globle Registrierung / Lokalisierung

Die globale Registrierung wird zur Lokalisierung während der Navigation genutzt. Bei der Navigation sind die inkrementell registrierten Einzelscans in einer großen Extended Elevation Map integriert. Zusätzlich werden die Positionen und Ausrichtungen der einzelnen registrierten Maps in einer Datenbank gespeichert. Zur Lokalisierung ist es notwendig, den Roboter kurz anzuhalten und einen Lokalisierungsscan aufzunehmen. Ausgehend von der durch die Odometriesensoren und einer früheren Lokalisierung geschätzten Position sucht man in der Datenbank den Einzelscan, der mit dem Lokalisierungsscan die größte Überlappung aufzuweisen hat. In diesem Scan kann man dann den aktuellen Lokalisierungsscan registrieren.

In diesem Zusammenhang liegt die Frage nahe, warum man den Lokalisierungsscan nicht direkt in der kompletten Extended Elevation Map Registriert. Dafür gibt es verschieden Gründe:

- Speicheroptimierung
- Laufzeitoptimierung
- Optimierung der Lokalisierung

Speicher- und Laufzeitoptimierung

Zum sicheren Registrieren eines neuen Scans ist es wichtig, möglichst alle Scanpunkte in die Extended Elevation Map einzufügen. Geht man allerdings

von einer Umgebung von 100×100 Metern so kommen wie bereits erwähnt sehr schnell 10 bis 20 Millionen Scanpunkte zusammen. Diese große Anzahl an Punkten verursacht einen erheblichen Speicher- und Rechenaufwand. Hat man allerdings zum Ziel zuerst eine Strecke abzufahren, dabei die Umgebung abzuscannen, eine Karte zu bauen und schließlich darin Pfade zu planen und zu navigieren, ist es nicht notwendig so viele Punkte zu betrachten. Durch die Winkelauflösung des Laser-Range-Finders und der Messanordnung auf dem Roboter ist es nämlich nur im näheren Umfeld des Roboters möglich, traversierbares Gelände sicher auch als traversiebar zu erkennen. In größerer Entfernung reicht dazu die Datendichte einfach nichtmehr aus. Deshalb fügt man nur Scanpunkte in die Elevation Map ein, die in diesem Bereich liegen. Dieser Bereich liegt je nach Geschwindigkeit der Bewegung des Lasers bei der Datenakquisition zwischen 3 und 10 Metern.

Optimierung der Lokalisierung

Fügt man viele verschiedene Einzelscans in eine Elevation Map ein, summieren sich die Registrierungsfehler. Dieser Effekt wird durch die Diskretisierung der Scanpunkte in Gitterzellen noch verstärkt. Da der Registrierungsfehler bei einem einzelnen Scan sehr gering ist, ist es deshalb besser, zur Lokalisierung nur den Scan mit der größten Überlappung zu verwenden. Gibt es keinen Einzelscan mit ausreichender Überlappung, ist es immernoch möglich, den Registrierungsalgorithmus auf die Elevation Map des gesamten Gebietes anzuwenden.

3.5.2 Pfadplanung

Abbildung 3.7 zeigt eine generierte 2D Map, in der ein Pfad mittels A^*-Suche [18] berechnet wurde. Da die Pfadplanung in zwei Dimensionen mittels A^*-Suche zu den Standardalgorithmen der Robotik gehört und nicht den Kern dieser Arbeit darstellt, möchten wir hierrauf nicht weiter eingehen. Die rote Farbe in Abbildung 3.7 illustriert die Nichtraversierbarkeit dieser Zellen. Die grünen Zellen wurden wie in Kaptiel 2.5 beschrieben als traversierbar erkannt. Die roten Flecken in der Straße haben ihre Ursache im Grasbewuchs, der an dieser Stelle vorliegt. Das rechte Bild in Abbildung 3.7 zeigt die Kostenkarte, mit deren Hilfe dieser Pfad geplant wurde. Die Kosten definieren sich direkt aus der Traversierbarkeit der Zellen. Die hier abgebildete Karte wurde darüber hinaus durch fünffaches Falten geglättet. Als Heuristik für die A^*-Suche wird der euklidische Abstand zum Zielpunkt verwendet.

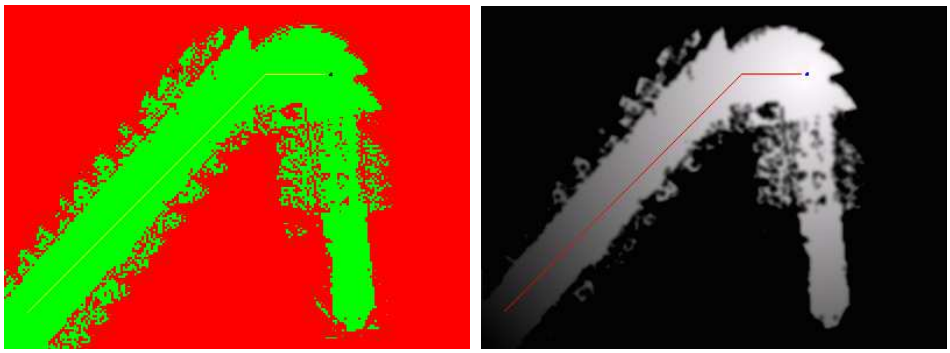

Abbildung 3.7: 2D-Karte und Kostenkarte

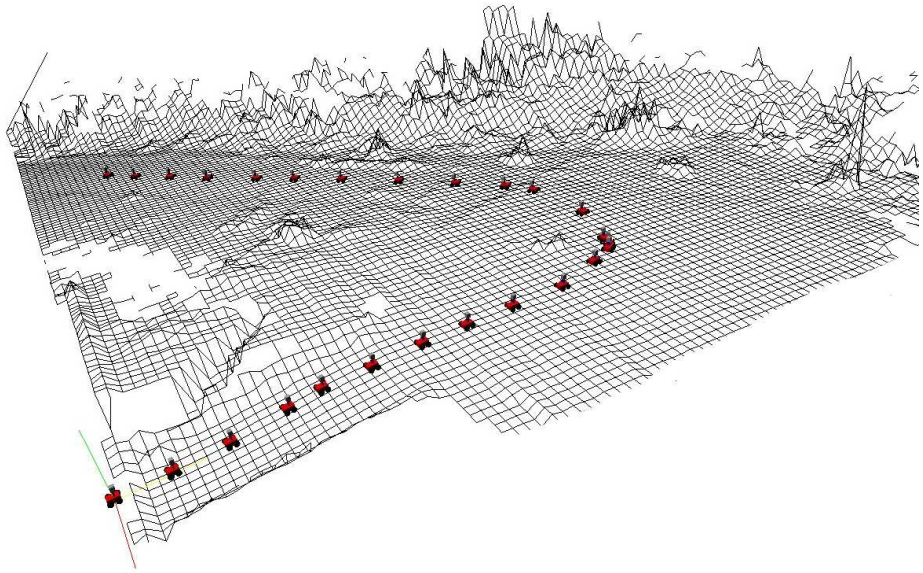

Abbildung 3.8: Elevation Map

Abbildung 3.8 zeigt die Elevation Map, aus der die 2D-Karte und die Kostenkarte aus Abbildung 3.7 berechnet wurden. Die Szene befindet sich auf dem Campus der Fakultät für angewandte Wissenschaften. Man sieht wie der Roboter Herbert die Laderampe des Kellergeschosses von Gebäude 079 nach oben auf die Straße fährt. Die eingezeichneten Roboter zeigen, an welchen Stellen Daten zum Erstellen der Kostenkarte aufgenommen wurden. Für die Traversierbarkeitsberechnung wurde eine Zellengröße von 0,1m $\times$ 0,1m verwendet. Zur Visualisierung in Abbildung 3.8 musste bedingt durch Aliasing eine Zellengröße von 0,4 $\times$ 0,4m gewählt werden.

3.5.3 Navigations-Algorithmus

Vorrausgesetzt es gibt eine ausreichende Menge von Scans, so dass es einen traversierbaren Pfad von A nach B gibt, geht man bei der Navigation durch eine zuvor gelernte Elevation Map M folgendermaßen vor. Man plant einen Pfad von A nach B und erzeugt sich gegebenfalls Zwischenzielpunkte. Jetzt kann man einen Zielpunkt nach dem anderen anfahren und dabei den foldenden Algorithmus 3.5.3 ausführen. Die Größe d_{driven} bezeichnet die gefahrene Strecke seit der letzten Lokalisierung. Die Position $p_t(o_t, l_t)$ des Roboters zum Zeitpunkt t wird während der Fahrt durch ständige Odometriemessungen o_t und die letzte Lokalisierung l_t geschätzt.

Input: Datenbank der Registrierten Scans, gelernte Elevation Map M, Positionsschätzung p_t zu jedem Zeitpunkt t, Abstand zum Ziel d_{goal}, nächster Zielpunkt g_i
while $d_{goal} \geq \epsilon$ **do**
  lokalisiere Roboter
  $p_t(o_t, l_t) := l_t$
  $d_{driven} := 0$
  while $d_{driven} \leq \delta \wedge d_{goal} \geq \epsilon$ **do**
    fahre in Richtung des nächsten Zielpunktes g_i
    $p_t(o_t, l_t) := p_t(o_t, l_t) + o_t$
    $d_{driven} := |l_t - p_t(o_t, l_t)|$
    $d_{goal} := |d_{driven} - p_t(o_t, l_t)|$
  end while
end while

In Worten bedeutet dies, dass der Roboter immer solange in Richtung des nächsten Zielpunktes navigiert, bis er entweder eine bestimmte Distanz zurückgelegt hat oder am Zielpunkt angekommen ist. Trifft einer der beiden Fälle zu, wird der Roboter angehalten, um sich zu lokalisieren.

KAPITEL 4

Experimente

4.1 Lernen von Elevation Maps

Wie wir in Kapitel 3 bereits gezeigt haben, werden Elevation Maps eines größeren Gebietes inkrementell aus den Elevation Maps der Einzelscans gelernt. Im folgenden Beispiel wird eine Karte eines Gebietes erstellt, das ca. 35m × 70m groß ist. Der maximale Höhenunterschied dieses Gebietes beträgt ungefähr 6m. Die Karte wurde aus 36 verschiedenen Elevation Maps zusammengesetzt und integrierte insgesamt 9,5 Millionen 3D-Scanpunkte. Abbildung 4.1 zeigt die gelernte Karte mit einer Zellengröße von 0,2m × 0,2m. Abbildung 4.2 zeigt noch einmal Detailansichten der selben Karte aus verschiedenen Perspektiven.

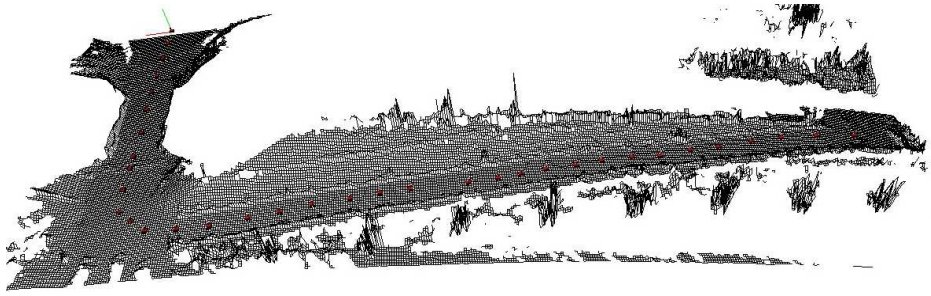

Abbildung 4.1: Map aus 36 Scans

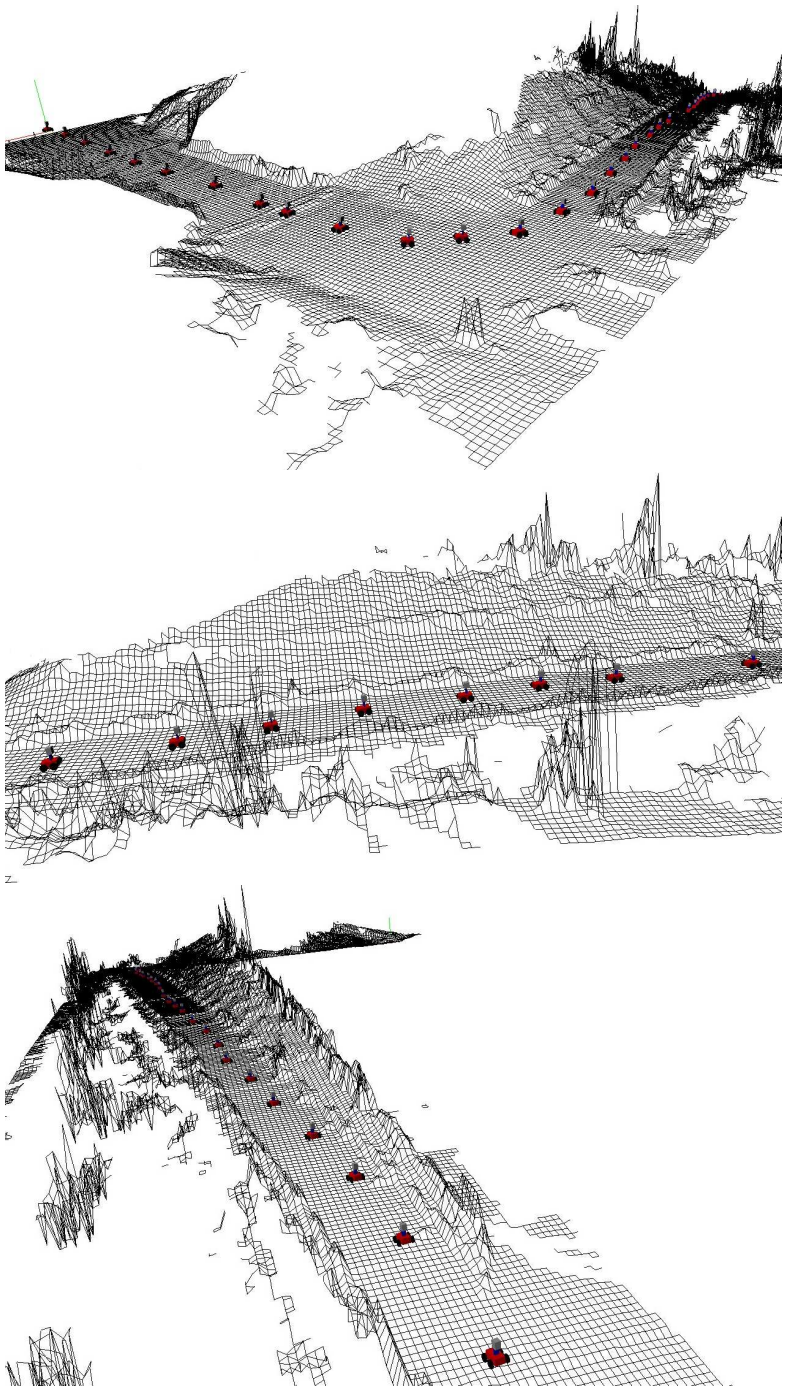

Abbildung 4.2: Detailansichten

Abbildung 4.3 verdeutlicht noch einmal einen Vorteil der erweiterten Elevation Map. In der dargestellten Szene passiert der Roboter einen Baum, der sich im vorderen Teil des Bildes befindet. Das obere Bild zeigt, wie diese Situation in einer nicht erweiterten Elevation Map dargestellt wird. Das untere Bild zeigt den Baum in der erweiterten Elevation Map. Er erscheint dort wesentlich dünner, da nur sein Stamm dargestellt wird. Es wäre anhand der Extended Elevation map also möglich sehr nahe am Stamm vorbei unter diesem Baum hindurchzufahren.

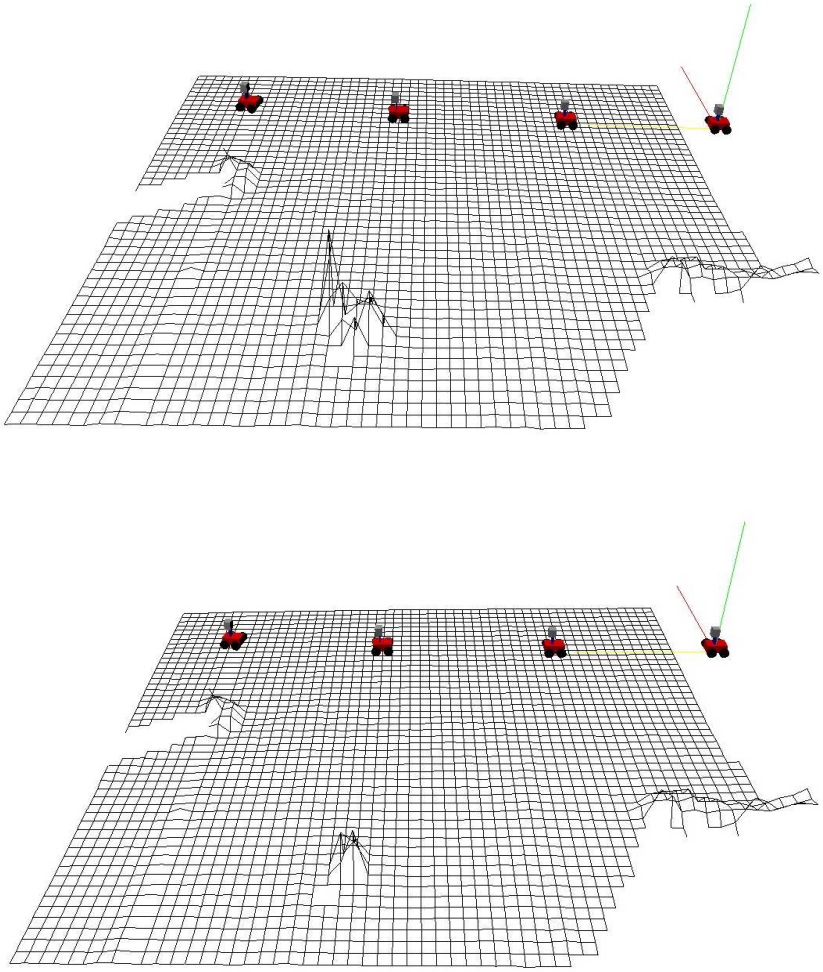

Abbildung 4.3: Roboter passiert einen Baum

4.2 Statistische Experimente

Hier wollen wir die Robustheit und Effizienz des vorgestellten Verfahrens empirisch nachweisen. Eine Möglichkeit dafür ist ein Vergleich mit bereits bestehenden Varianten.

In den folgenden Tests werden wir die Robustheit und Effizienz des Matchens durch den ICP-Algorithmus für Extended Elevation Maps nachweisen. Wir werden Zeigen, dass die Erweiterung der Elvation Map um die Klassifizierung der nichttraversierbaren Zellen erhebliche Verbesserungen mit sich bringt.

Die Test führen wir mit Hilfe zweier "perfekt" registrierter Scans durch. Abbildung 4.4 zeigt die Punktmodelle und Abbildung 4.5 die Elevationmap dieser beiden Scans. Da bei realen Daten ein absolut perfektes Matching nicht existiert, habe ich diesen Fall durch ein ICP-Scan-Matching mit einer extrem niedrigen Abbruchschranke für die durchschnittliche Fehleränderung erzeugt. Für diese Schranke wurde $\epsilon = 10^{-9}$ gewählt.

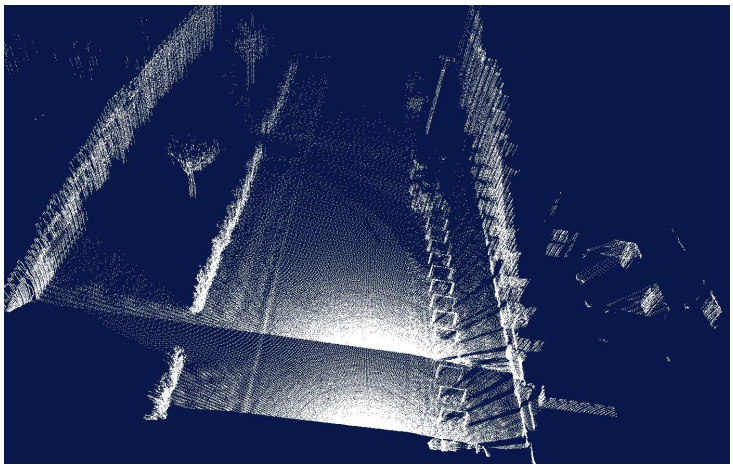

Abbildung 4.4: Perfektes Scan-Matching

Die perfekt registrierten Scans erlauben es ein Fehlermaß wie folgt zu definieren: um die Eigenschaften des Scan-Matching-Algorithmus zu überprüfen, werden zufällig Positionsabweichungen aus verschiedenen Klassen solcher Abweichungen gewählt. Diese Klassen geben den Bereich der maximalen Abweichung an. Einer der beiden Scans wird um diese Positionsabweichung euklidisch transformiert und danach wieder registriert. Durch die Distanz der neuen, aus der neuen Registrierung resultierenden Position, zur "perfekten" Registrierung lässt sich dann eine Aussage über die Qualität der verschiedenen Matching-Algorithmen treffen.

Die vier Matching-Algorithmen basieren alle auf dem ICP Algorithmus und unterscheiden sich nur in der Auswahl der Punkte, die zur Korrespondenzensuche herangezogen werden. Die vier Punktauswahlstrategien sind:

1. zufällige Auswahl von 3000 Punkten aus jeder Puntmenge,
2. zufällige Auswahl einer festen Zahl an Punkten aus allen nichttraversierbaren Zellen,
3. zufällige Auswahl einer festen Zahl an Punkten aus allen nichttraversierbaren Zellen, die als Zelle mit vertikalen Objekten, als Zellen, die von oben gesehen wurden, oder als Zelle mit einer vertikalen Lücke klassifiziert wurden und
4. zufällige Auswahl wie 3.), wobei bei der Korrespondenzensuche zusätzlich zwischen den Klassen separiert wird, wie in Kapitel 3.3 erklärt wird.

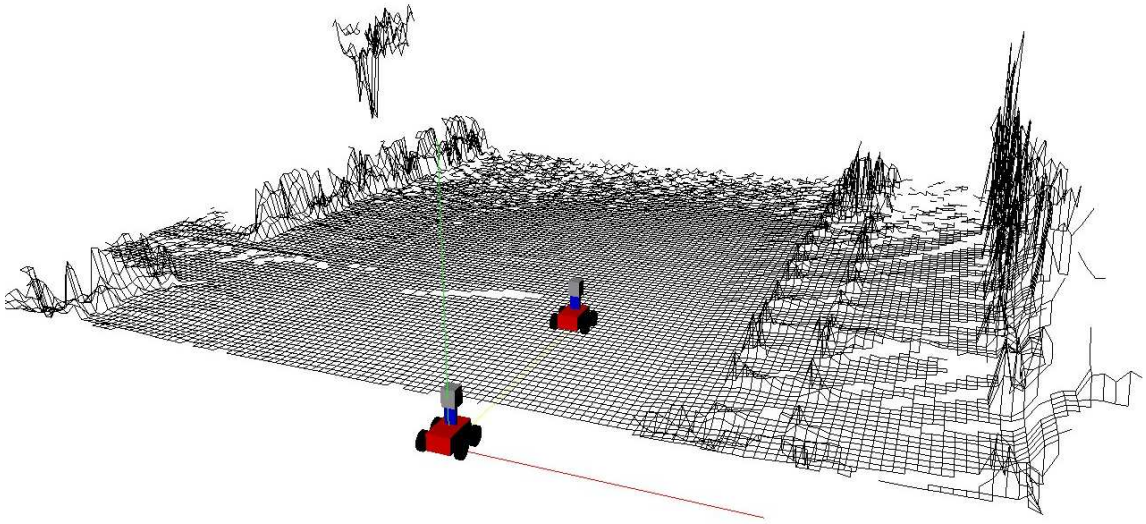

Abbildung 4.5: registrierte Test-Karten

Die Abbildungen 4.6, 4.7 4.8 und 4.9 zeigen exemplarisch die Unterschiede zwischen den vier Punktwahlstrategien anhand zweier Scans. Einer von beiden wurde durch eine zufällige Positionsabweichung verschoben.

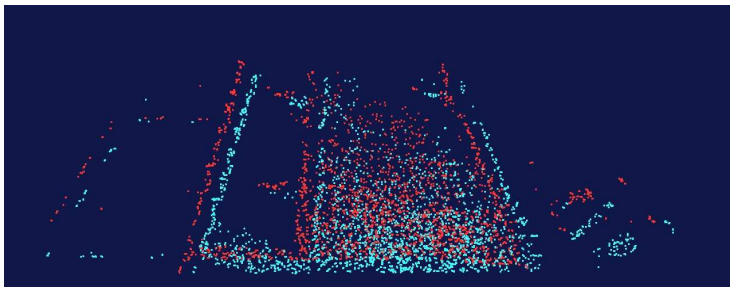

Abbildung 4.6: Punktwahlstrategie 1

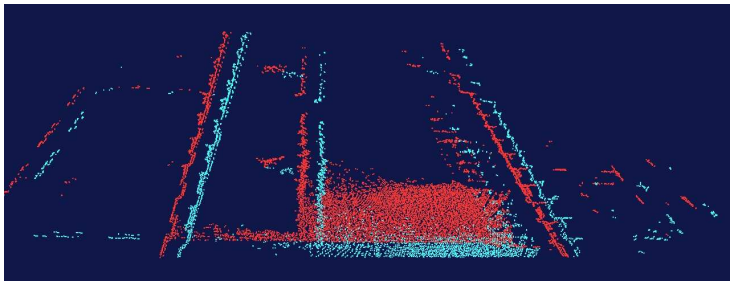

Abbildung 4.7: Punktwahlstrategie 2

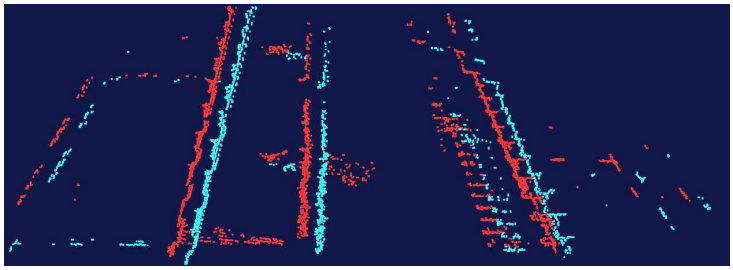

Abbildung 4.8: Punktwahlstrategie 3

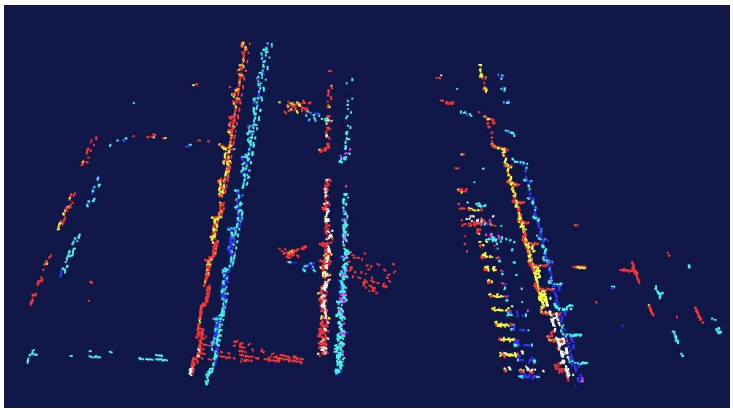

Abbildung 4.9: Punktwahlstrategie 4

Da es sehr schwierig ist für Translation und Rotation eine gemeinsame Fehlerfunktion zu modellieren, werden wir den Matching-Algorithmus für extendet Elevation Maps unabhängig voneinander für beide Transformationstypen betrachten.

4.2.1 Translationsfehler

Zur Bewertung des Scanmatchings verwenden wir bei dieser Versuchsreihe die euklidische Distanz $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Der Vektor $P_1(x_1, y_1)$ beschreibt hierbei die Roboterposition für den registrierten Scan, die sich aus dem "perfekten" Matching ergeben hat. Der Vektor $P_2(x_2, y_2)$ beschreibt die Roboterposition, welche die Registrierung nach der Addition einer Positionsabweichung ergeben hat. In dieser Versuchsreihe wird bei jedem Versuch die Rotationsabweichung aus einem festen Bereich zufällig gewählt. Der Bereich für die Translationsabweichung wird variiert. Man wählt also zufällig 30 Positionsabweichungen aus jeder Abweichungsklasse und registriert danach die daraus erhaltenen transformierten Scans. Dies führt man für alle vier Versionen des ICP-Algorithmus durch. Man verwendet selbstverständlich für jede Form des Algorithmus exakt die selben Abweichung.

Versuch 1

Tabelle 4.1 zeigt die verschiedenen Abweichungs-Klassen, die für Versuch 1 verwendet werden:

	max. Rot.-Abweichung	max. Abweichung in x und y
Abweichungsklasse 1	± 0 Grad	$\pm 0.5m$
Abweichungsklasse 2	± 0 Grad	$\pm 1.0m$
Abweichungsklasse 3	± 0 Grad	$\pm 1.5m$
Abweichungsklasse 4	± 0 Grad	$\pm 2.0m$
Abweichungsklasse 5	± 0 Grad	$\pm 2.5m$

Tabelle 4.1: Abweichungs-Klassen Versuch 1

Abbildung 4.10 zeigt die Divergenzwahrscheinlichkeit des ICP Algorithmus für die 4 unterschiedlichen Punktauswahlstrategien. Für Strategie 3 und 4 ergeben sich in diesem Fall die selben Wahrscheinlichkeiten. Diese Graphik zeigt die Wahrscheinlichkeit an, mit welcher der ICP-Algorithmus divergiert. Als Divergenz bezeichnet man den Fall, dass nach n Iterationsschritten für den Translationsfehler $d \geq \delta$ gilt. Für den Fall, dass der Algorithmus früher abbricht, gilt er als konvergent. Dies geschieht, wenn die Änderung des im ICP zu minimierenden Fehlers unter eine ϵ-Schranke absinkt. In den Versuchen wurde $\epsilon = 10^{-5}$, $\delta = 0.25m$ und $n = 30$ gesetzt.

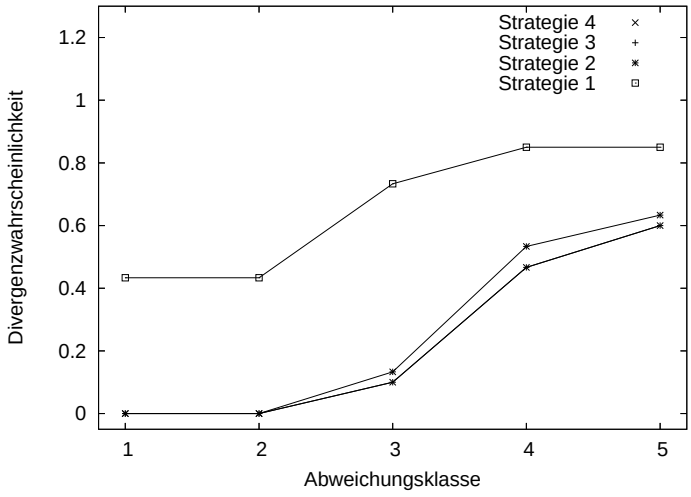

Abbildung 4.10: Divergenzwahrscheinlichkeit Versuch 1

Abbildung 4.11 zeigt die durch den t-Test berechneten Konfidenzintervalle. Diese Konfidenzintervalle sagen folgendes aus: ist der Algorithmus konvergent, dann liegen 95% der berechneten Fehler in dem Bereich, der durch die Konfidenzintervalle begrenzt wird. Das Symbol in der Mitte dieser Konfidenzintervalle bezeichnet den Mittelwert des Fehlers.

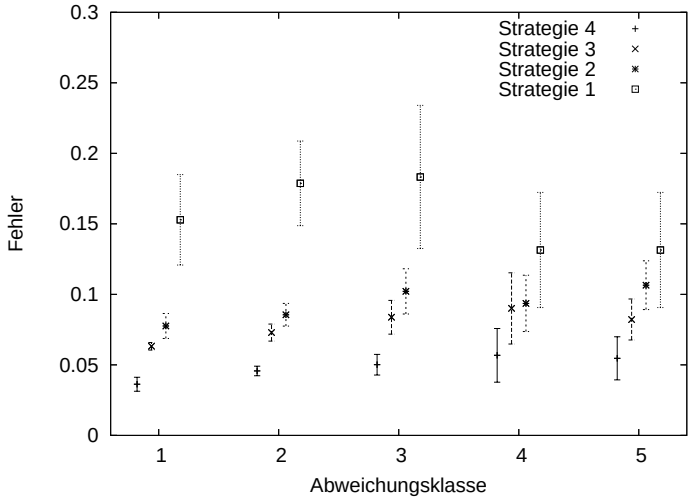

Abbildung 4.11: Konfidenzintervalle Versuch 1

Auswertung Versuch 1

Abbildung 4.10 zeigt, dass die Klassifizierung der Zellen die Konvergenzwahrscheinlichkeiten stark erhöhen. Auf die Wahrscheinlichkeiten hat die Separation zwischen den Klassen keine Auswirkungen. Die Konfidenzintervalle in Abbildung 4.11 lassen jedoch eine klare Tendenz erkennen: die Separation der Zellen verbessert die Registrierung. Das lässt sich dadurch erklären, dass die Mittelwerte des Translationsfehlers, der durch den ICP mit Zellseparation (Strategie 4) generiert wird, immer unterhalb von allen anderen Mittelwerten liegen. Vergleicht man die Konfidenzintervalle, die aus dem Anwenden der Punktwahlstrategie 4 entstehen, mit denen, die beim Anwenden der Standardimplementierung des ICP (Strategie 1) auftreten, so kann man sogar eine signifikante Verbesserung des ICP-Algorithmus durch die Zellklassifikation feststellen.

Versuch 2

Im zweiten Versuch haben wir den Einfluss einer größeren Rotationsabweichung untersucht. Tabelle 4.2 zeigt die verschiedenen Abweichungs-Klassen, die für Versuch 2 verwendet werden:

	max. Rot.-Abweichung	max. Abweichung in x und y
Abweichungsklasse 1	± 5 Grad	$\pm 0.5m$
Abweichungsklasse 2	± 5 Grad	$\pm 1.0m$
Abweichungsklasse 3	± 5 Grad	$\pm 1.5m$
Abweichungsklasse 4	± 5 Grad	$\pm 2.0m$
Abweichungsklasse 5	± 5 Grad	$\pm 2.5m$

Tabelle 4.2: Abweichungs-Klassen Versuch 2

Wir haben also die maximale Abweichung in x- und y-Richtung schrittweise um 0,5m vergrößert, um den Einfluss auf die Registrierungsgenauigkeit zu untersuchen. Abbildung 4.12 zeigt die Divergenzwahrscheinlichkeit des ICP Algorithmus für die 4 unterschiedlichen Punktauswahlstrategien.

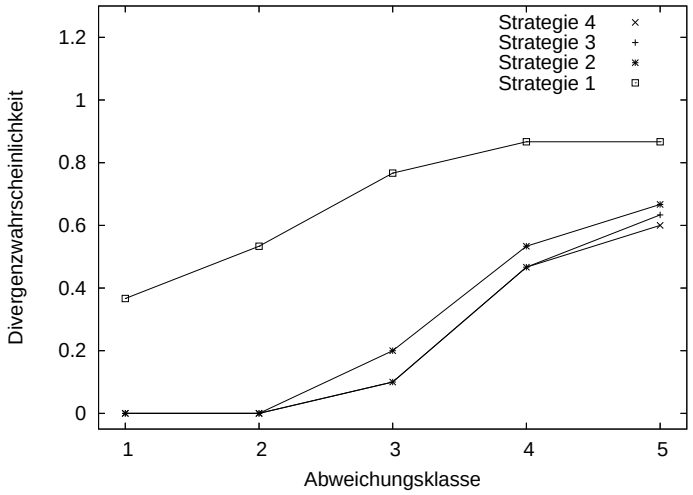

Abbildung 4.12: Divergenzwahrscheinlichkeit Versuch 2

Abbildung 4.13 zeigt die durch den t-Test berechneten Konfidenzintervalle.

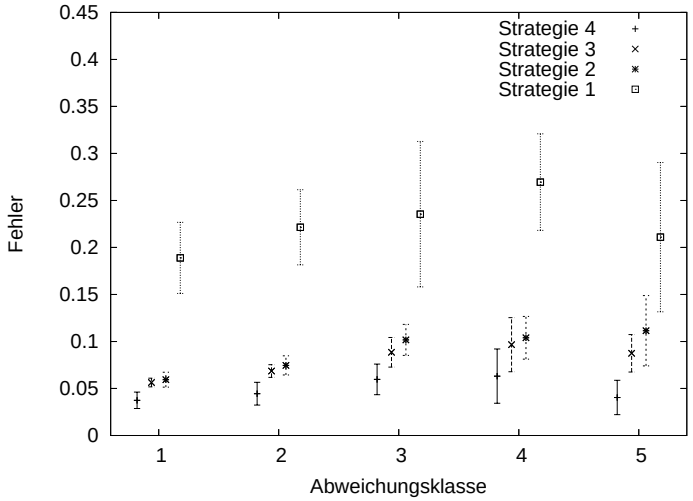

Abbildung 4.13: Konfidenzintervalle Versuch 2

Auswertung Versuch 2

Die Zusätzliche Rotation in Versuch 2 wirkt sich im Vergleich zu Versuch 1 in erster Linie negativ auf die Divergenzwahrscheinlichkeiten des ICP mit zufälliger Auswahl aus den kompletten Punktmengen (Strategie 1) aus. Die Divergenzwahrscheinlichkeit von Strategie 1 steigt etwas schneller an, als bei Versuch 1. Die Wahrscheinlichkeiten bezüglich der anderen Strategien bleiben nahezu unverändert. Für die Fehleranalyse aus Abbildung 4.13 gilt das selbe wie für die Wahrscheinlichkeiten. Die Mittelwerte der Konfidenzintervalle für Strategie 2, 3 und 4 bleiben im Vergleich zu Versuch 1 stabil, während die Mittelwerte bei Strategie 1 sichtbar ansteigen. Überdies zeigt sich der ICP mit Strategie 4 über alle Abweichungsklassen hinweg signifikant besser als die ICP-Version mit uniformem Sampling und für die erste Abweichungsklasse sogar signifikant besser als alle anderen Strategien.

Versuch 3

Um die Ergebnisse des zweiten Versuchs zu validieren, haben wir in einem dritten Versuch die Rotation nochmals um 5 Grad vergrößert. Tabelle 4.3 zeigt die verschiedenen Abweichungs-Klassen, die für Versuch 3 verwendet werden:

	max. Rot.-Abweichung	max. Abweichung in x und y
Abweichungsklasse 1	± 10 Grad	$\pm 0.5m$
Abweichungsklasse 2	± 10 Grad	$\pm 1.0m$
Abweichungsklasse 3	± 10 Grad	$\pm 1.5m$
Abweichungsklasse 4	± 10 Grad	$\pm 2.0m$
Abweichungsklasse 5	± 10 Grad	$\pm 2.5m$

Tabelle 4.3: Abweichungs-Klassen Versuch 3

Abbildung 4.14 zeigt die Divergenzwahrscheinlichkeit des ICP Algorithmus für die 4 unterschiedlichen Punktauswahlstrategien.

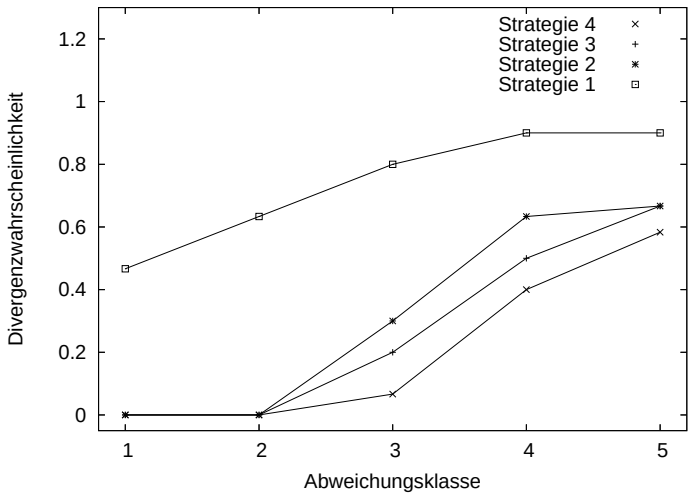

Abbildung 4.14: Divergenzwahrscheinlichkeit Versuch 3

Abbildung 4.15 zeigt die durch den t-Test berechneten Konfidenzintervalle.

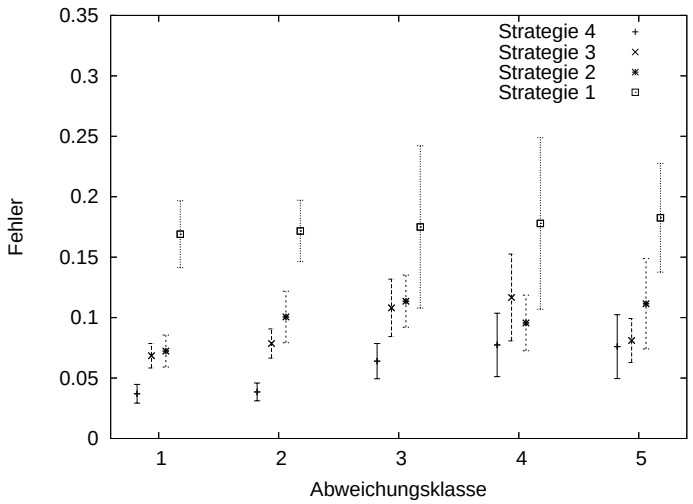

Abbildung 4.15: Konfidenzintervalle Versuch 3

Auswertung Versuch 3

Durch das weitere Addieren einer Rotation im Vergleich zu Versuch 2 erhöhen sich die Divergenswahrscheinlichkeiten der ersten drei Strategien. Es bleiben lediglich Divergenzwahrscheinlichkeiten des ICP auf den separierten Klassen im Vergleich zu Versuch 1 und Versuch 2 stabil. Bei der Fehleranalyse verhält sich nurnoch Strategie 4, mit kleinen Abweichungen in Abweichungsklasse 4 und 5 stabil. Überdies zeigt sich der ICP mit Strategie 4 über alle Abweichungsklassen hinweg signifikant besser als die ICP-Version mit zufälliger Punktauswahl und für die ersten drei Abweichungsklassen sogar signifikant besser als alle anderen Strategien. Wir konnten also die Ergebnisse der vorherigen Experimente bestätigen: Durch das Anwenden der Fehlerfunktion (3.15) ergibt sich in allen Versuchen eine signifikante Verbesserung im Vergleich zur Standardfehlerfunktion des ICP-Algorithmus.

4.2.2 Rotationsfehler

Bisher haben wir eine konstante Rotationsabweichung und eine variierende Tranlationsabweichung angenommen. Hier wollen wir die Translationsabweichung konstant halten und den Einfluss der Rotation untersuchen. Zur Bewertung des Scan-Matchings verwenden wir bei dieser Versuchsreihe den Rotationsfehler $d_{rot} = \theta_1 - \theta_2$. Die Größe θ_1 beschreibt die Orientierung des registrierten Scans, die sich aus der "perfekten" Registrierung ergeben hat. Die Größe θ_2 beschreibt die Orientierung, welche die Registrierung nach der Addition einer Positionsabweichung ergeben hat. In dieser Versuchsreihe wird bei jedem Versuch die Translations-Abweichung aus einem festen Bereich zufällig gewählt. Der Bereich für die Rotationsabweichung wird variiert.

Versuch 4

Tabelle 4.4 zeigt die verschiedenen Abweichungs-Klassen, die für Versuch 4 verwendet werden:

	max. Rot.-Abweichung	max. Abweichung in x und y
Abweichungsklasse 1	± 0 Grad	$\pm 0.0m$
Abweichungsklasse 2	± 5 Grad	$\pm 0.0m$
Abweichungsklasse 3	± 10 Grad	$\pm 0.0m$
Abweichungsklasse 4	± 20 Grad	$\pm 0.0m$

Tabelle 4.4: Abweichungs-Klassen Versuch 4

Abbildung 4.16 zeigt die Divergenzwahrscheinlichkeit des ICP Algorithmus für die vier unterschiedlichen Punktauswahlstrategien. Diese Kurven zeigen wieder die Wahrscheinlichkeit an, mit welcher der ICP-Algorithmus divergiert. Mit Divergenz bezeichnet man analog zur Translationsabweichung den Fall, dass nach n Iterationsschritten für den Rotationsfehler $d \geq \delta$ gilt. Für den Fall, dass der Algorithmus früher abbricht, gilt er als konvergent. Dies geschieht, wenn die Änderung des im ICP zu minimierenden Fehlers unter eine ϵ-Schranke absinkt. In den Versuchen wird $\epsilon = 10^{-5}$, $\delta = 1.5$ Grad und $n = 30$ gesetzt.

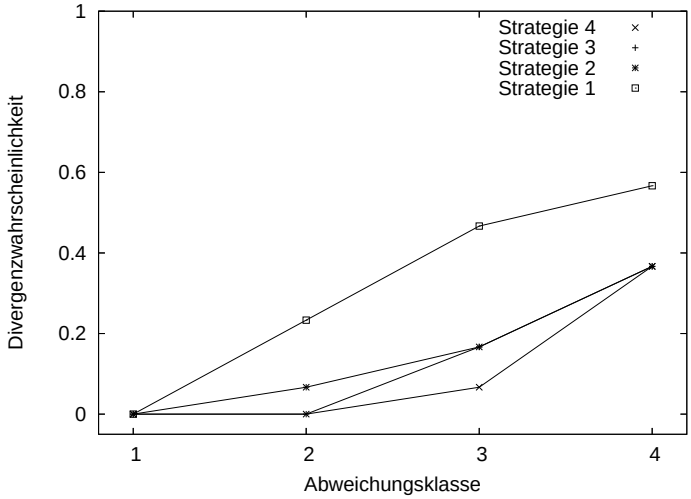

Abbildung 4.16: Divergenzwahrscheinlichkeit Versuch 4

Abbildung 4.17 zeigt die durch den t-Test berechneten Konfidenzintervalle.

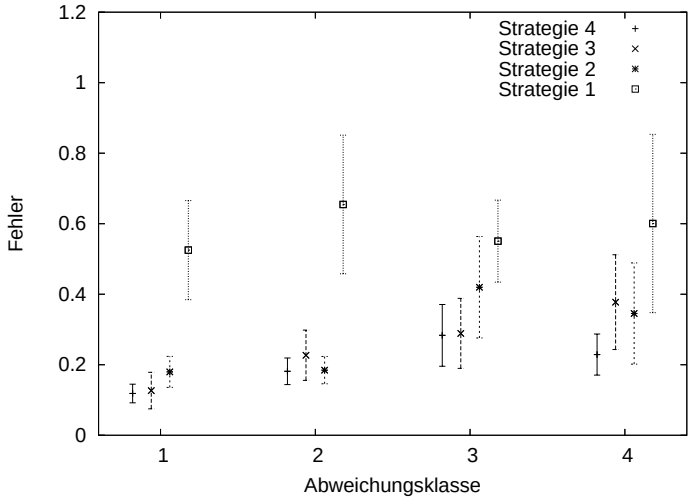

Abbildung 4.17: Konfidenzintervalle Versuch 4

Auswertung Versuch 4

Bei der Analyse der Divergenzwahrscheinlichkeiten zeigt sich, dass die Strategien 2, 3 und 4 für alle Abweichungs-Klassen eine niedrigere Divergenzwahrscheinlichkeit aufweisen als Strategie 1. Für die Abweichungsklasse 1 sind alle Divergenzwahrscheinlichkeiten Null, da keine Positionsabweichung vorliegt.

Bei der Fehleranalyse ist auffällig, dass der Standard-ICP(Strategie 1) im Fall der Konvergenz für alle Abweichungs-Klassen ungefähr den selben Rotationsfehler liefert. Dieser Fehler schwankt um ca. 0,6 Grad. Bei den anderen Strategien ist es im Gegensatz dazu allerdings möglich, bei geringerem Anfangsfehler ein besseres Ergebnis zu erreichen. Strategie 4 ist auch in diesem Versuch bezüglich des Rotationsfehlers über alle Abweichungs-Klassen hinweg signifikant besser als Strategie 1 und tendenziell besser als die anderen drei Strategien. Außerdem legen die Ergebnisse den Schluss nahe, dass der Minimalfehler nach dem Registrieren mit Strategie 2, 3 und 4 immer geringer ist, als für Strategie 1. Dies folgt aus den Konfidenzintervallen für die Abweichungsklasse 1, bei der Rotation und Translation gleich null sind.

Versuch 5

Tabelle 4.5 zeigt die verschiedenen Abweichungs-Klassen, die für Versuch 5 verwendet werden:

	max. Rot.-Abweichung	max. Abweichung in x und y
Abweichungsklasse 1	± 0 Grad	$\pm 0.5m$
Abweichungsklasse 2	± 5 Grad	$\pm 0.5m$
Abweichungsklasse 3	± 10 Grad	$\pm 0.5m$
Abweichungsklasse 4	± 20 Grad	$\pm 0.5m$

Tabelle 4.5: Abweichungs-Klassen Versuch 5

Abbildung 4.18 zeigt die Divergenzwahrscheinlichkeit des ICP Algorithmus für die 4 unterschiedlichen Punktauswahlstrategien.

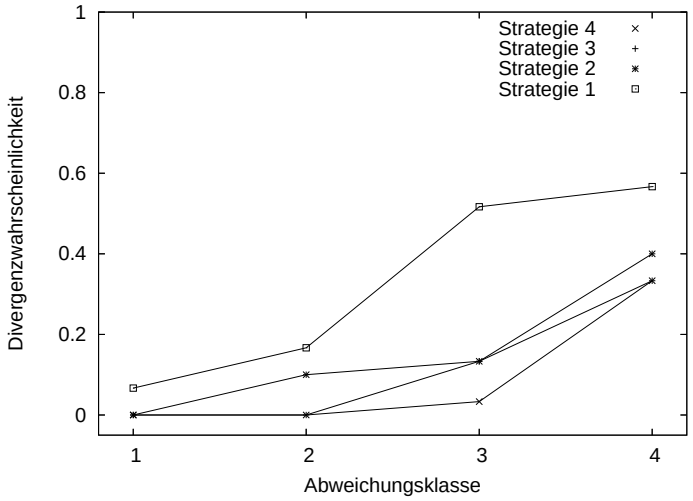

Abbildung 4.18: Divergenzwahrscheinlichkeit Versuch 5

Abbildung 4.19 zeigt die durch den t-Test berechneten Konfidenzintervalle.

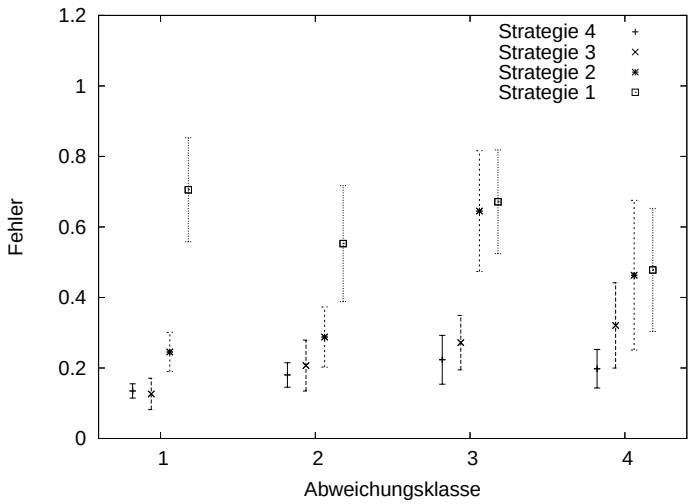

Abbildung 4.19: Konfidenzintervalle Versuch 5

Auswertung Versuch 5

Bei der Analyse der Divergenzwahrscheinlichkeiten zeigt sich, dass die Strategien 2, 3, 4 für alle Abweichungs-Klassen eine niedrigere Divergenzwahrscheinlichkeit aufweisen als Strategie 1. Für die Abweichungsklasse 1 haben alle bis auf den Algorithmus, der Strategie 1 verwendet, die Divergenzwahrscheinlichkeit Null. Die Fehleranalyse ähnelt stark Versuch 4. So zeigt sich wiederum, dass der Standard-ICP(Strategie 1) im Fall der Konvergenz für alle Abweichungs-Klassen ungefähr den selben Rotationsfehler liefert. Dieser Fehler schwankt auch hier um ca. 0,6 Grad. Diese Eigenschaft weist auch die ICP-Version auf, welche die Punktwahlstratgie 2 verfolgt. Bei den anderen beiden Strategien, die nur die klassifizierten Zellen verwenden, ist es im Gegensatz dazu auch in diesem Versuch der Fall, dass aus einem geringerem Anfangsfehler ein besseres Ergebnis resultiert. Strategie 4 ist auch in diesem Versuch bezüglich des Rotationsfehlers über alle Abweichungsklassen hinweg signifikant besser als Strategie 1 und tendenziell besser als die anderen drei Strategien.

Versuch 6

Tabelle 4.6 zeigt die verschiedenen Abweichungs-Klassen, die für Versuch 6 verwendet werden:

	max. Rot.-Abweichung	max. Abweichung in x und y
Abweichungsklasse 1	± 0 Grad	$\pm 1.0m$
Abweichungsklasse 2	± 5 Grad	$\pm 1.0m$
Abweichungsklasse 3	± 10 Grad	$\pm 1.0m$
Abweichungsklasse 4	± 20 Grad	$\pm 1.0m$

Tabelle 4.6: Abweichungs-Klassen Versuch 6

Abbildung 4.20 zeigt die Divergenzwahrscheinlichkeit des ICP Algorithmus für die 4 unterschiedlichen Punktauswahlstrategien. Die Divergenzwahrscheinlichkeiten von Strategie 3 und 4 entsprechen in diesem Versuch einander.

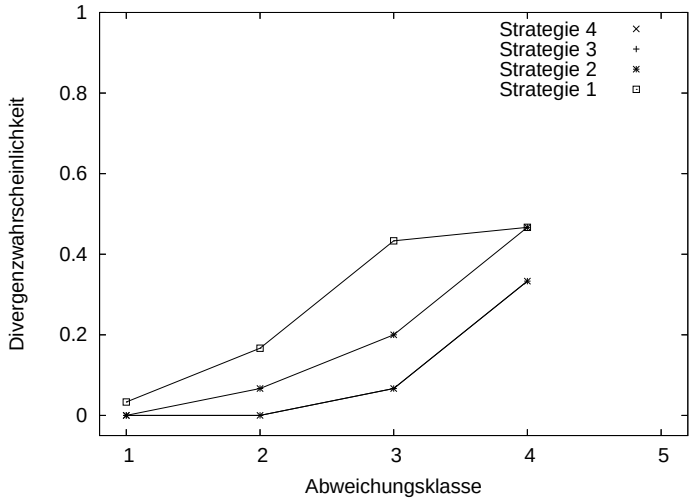

Abbildung 4.20: Divergenzwahrscheinlichkeit Versuch 6

Abbildung 4.21 zeigt die durch den t-Test berechneten Konfidenzintervalle.

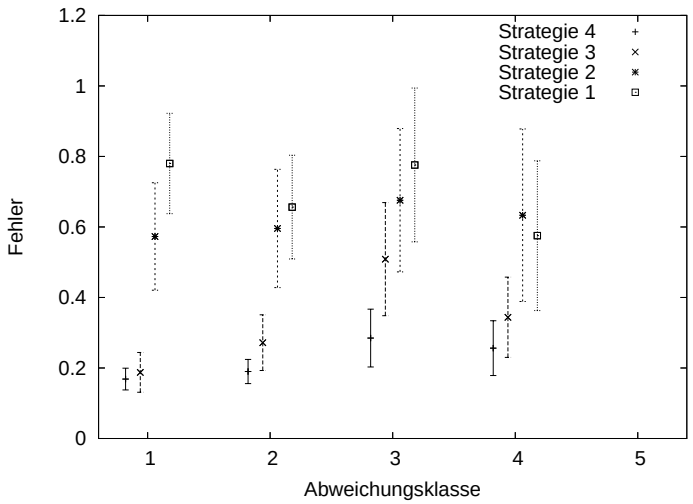

Abbildung 4.21: Konfidenzintervalle Versuch 6

Auswertung Versuch 6

Bei der Analyse der Divergenzwahrscheinlichkeiten zeigt sich auch hier einmal mehr, dass die Strategien 2, 3, 4 für alle Abweichungs-Klassen eine niedrigere Divergenzwahrscheinlichkeit aufweisen als Strategie 1. Strategie 3 und 4 liefern hier eine identische Divergenzwahrscheinlichkeit. Für die Abweichungsklasse 1 haben alle bis auf Strategie 1, die Divergenzwahrscheinlichkeit null. Die Fehleranalyse ähnelt wiederum sehr stark Versuch 4. Der Standard-ICP (Strategie 1) liefert im Fall der Konvergenz für alle Abweichungs-Klassen ungefähr den selben Rotationsfehler. Dieser Fehler schwankt auch hier um ca. 0,6 Grad. Bei den anderen Strategien ist es im Gegensatz dazu auch in diesem Versuch der Fall, dass aus einem geringerem Anfangsfehler ein besseres Ergebnis resultiert. Strategie 4 ist auch in diesem Versuch bezüglich des Rotationsfehlers über alle Abweichungs-Klassen hinweg signifikant besser als Strategie 1 und tendenziell besser als die anderen.

4.2.3 Zusammenfassung der Ergebnisse

Durch die hier beschriebenen Tests gelangt man zu folgender Haupterkenntnis: die Erweiterung der Elevation Maps um die Klassifizierung der nichttraversierbaren Zellen, bringt einen signifikanten Vorteil beim Scan-Matching mit dem ICP-Algorithmus im Vergleich zur Standardversion, die aus den zu registrierenden Punktmengen für die Bestimmung der Korrespondenzen einen festen Prozentsatz der Scanpunkte zufällig auswählt. Die Separation zwischen den Klassen, also Punktwahlstrategie 4 bringt bezüglich des Translationsfehlers einen größeren Vorteil als bezüglich des Rotationsfehlers. Warum dies so ist wird klar, wenn man sich das Beispiel einer unendlich langen Wand mit einem Fenster vor Augen führt. Bei einem Matching von zwei Aufnahmen dieser Wand wird der Algorithmus ohne Separation (Strategie 3) der Zellen lediglich die beiden Aufnahmen bezüglich der Ausrichtung der Wand zur Deckung bringen. Es wird also nur der Rotationsfehler ausgeglichen. Erkennt der Matchingalgorithmus jedoch das Fenster als Zelle mit einer Lücke und separiert nach den Klassen (Strategie 4) bei der Wahl der Korrespondenzen, so werden die beiden Aufnahmen auch noch zusätzlich ihres Translationsfehlers korrigiert, da nicht nur die Wände sondern auch die Fenster in den Wänden zur Deckung gebracht werden. Dies erklärt, warum der Algorithmus, der die Separation benutzt, bezüglich des Rotationsfehlers nur tendenziell besser ist als die Version ohne Separation. Betrachtet man die Versuche 1-3 bei denen der Translationsfehler bewertet wird, so erkennt man, dass die Separation eine starke Verbesserung mit sich bringt und sich deutlich von den anderen drei Versionen des ICP-Algorithmus abhebt. Ferner fällt auf, dass der Standard-ICP-Algorithmus, der sich nur auf das Samplen aus Punktmengen stützt, einen nach unten beschränkten Rotationsfehler hat, der relativ unabhängig von der Positionsabweichung zu sein scheint. Es kann also vorkommen, dass die relative Ausrichtung zweier Scans, die durch eine sehr genaue Odometrie oder durch eine frühere Registrierung sehr gut vorbestimmt ist, nicht mehr weiter verbessert werden kann. In den Abweichungsklassen, die diesen Anfangsfehlerbereich von maximal $\pm$1m in jede Richtung abdecken, ist der Algorithmus durch die Erweiterung der Elevation Maps bezüglich des Translationsfehlers in allen Versuchen signifikant besser als die Versionen, welche die Erweiterung garnicht oder nur zum Teil nutzen. Für geringe Rotationsfehler ($\pm$10 Grad) bewirken alle drei Modifikationen des ICP eine signifikante Verbesserung gegenüber der Standardversion.

KAPITEL 5

Zusammenfassung und Ausblick

In dieser Arbeit haben wir ein System vorgestellt, mit dem es möglich ist, aus einzelnen 3D-Laserdaten, die mit einem mobilen Roboter aufgenommen wurden, Elevation Maps zu lernen und diese für die Navigation zu verwenden. Weil die Daten mit einem Roboter vom Boden aus akquiriert werden, kann es durch die unvollständige Weltsicht häufig zu fehlerhaften Annahmen über die Umwelt kommen. Deshalb erweitern wir diesen Ansatz um eine Klassifikation der Gitterzellen. Durch diese Klassifikation ist es möglich, Fehler in den Elevation Maps zu beseitigen, die Aufgrund der Art der Datenakquisition und der Annahme, dass es für jede Zelle genau eine Durchschnittshöhe geben muss, zustande kommen. Diese Erweiterung ermöglicht es dem Roboter bespielsweise unter Bäumen und Brücken durchzufahren. Zusätzlich haben wir in dieser Arbeit gezeigt, wie diese Klassifizierung die Datenassoziation verbessert, was zur Folge hat, dass der ICP-Algorithmus besser konvergergiert und genauere Elevation Maps liefert. Ein weiterer Vorteil beim Matching der erweiterten Elevation Maps im Gegensatz zu anderen ICP-Algorithmen ist, das die meisten der effizienten ICP-Versionen Zusatzinformationen, wie z.B. eine Schätzung der Überlappung (Overlap) benötigen. Diese Information ergibt sich bei uns direkt aus der Gitterstruktur derElevation Map. Die von uns verwendete und iplementierte ICP-Version für die Elvation Map erfordert als einzige Vorberechnung die Klassifizierung der Zellen. Alle in den Tests verwendeten Daten wurden mit einem realen Roboter in einer real existierenden Outdoor-Umgebung aufgenommen.

5.1 Ausblick

Diese Arbeit zeigt eine gute Möglichkeit, im Außenbereich mit einem autonomen Roboter Umgebungsmodelle auf Basis von Elevation Maps zu lernen und sie zur Navigation zu nutzen. Dieser Ansatz lässt viele interessante Möglichkeiten offen, das beschriebene System weiterzuentwickeln. Einerseits wäre es denkbar das Thema im Bereich der Navigation weiter zu vertiefen. Dabei wäre die Integration von GPS-Daten und die Nutzung von Neigungs- und Beschleunigungssensoren sicher angebracht.

Eine andere Möglichkeit an diesem Thema weiterzuarbeiten, ist der Bereich der Datenakquisition und dem Lernen großer Umgebungsmodelle. Dazu würde es sich anbieten, das Matching der Elevation Maps um Optimierungsalgorithmen zu erweitern, die ein 3D-loop-closing ermöglichen. Ein denkbares Ziel wäre es mit einem autonomen, explorierenden Roboter ein Modell vom gesamten Universitätsgelände aufzunehmen. Hierbei würde es sich sicher als hilfreich erweisen, Bilddaten einer Kamera zu integrieren, um das Modell realer darstellen zu können.

Darüber hinaus wollen wir untersuchen, in wie weit sich die hier beschriebenen Techniken im Kontext des Robocup Rescue einsetzen lassen.

KAPITEL 6

Anhang

6.1 Der mobile Outdoor-Roboter Herbert

6.1.1 Hardware

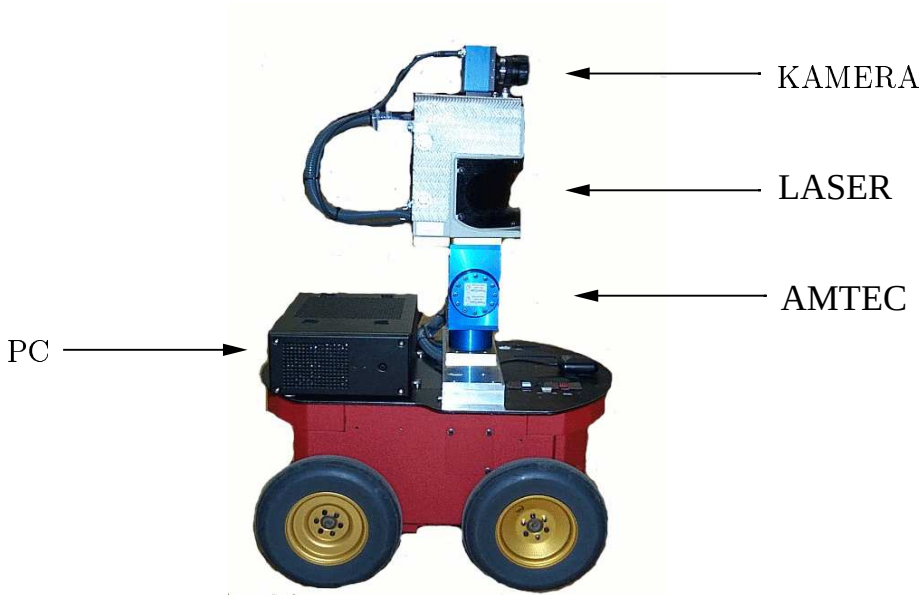

Abbildung 6.1: Roboter Herbert

Bei dem verwendeten Roboter mit dem Namen "Herbert" handelt es sich um einen Pioneer 2 All Terrain Roboter der Firma Active Media. Auf dieser Basis wurde eine AMTEC Pan & Tilt Einheit aufgesetzt, mit der sich ein SICK Laser-Range-Finder drehen und schwenken lässt. Zur Steuerung des Roboters wurde ein EPIA Mini-Computer verwendet, der auf dem Gehäuse des Pioneers montiert ist. Auf dem Laser ist zusätzlich noch eine Kamera montiert, die in dieser Arbeit allerdings nicht zum Einsatz kommt, weshalb sie an dieser Stelle lediglich erwähnt wird.

Die Roboter-Basis

Translations Geschwindigkeit	700 mm/Sek.
Steigung	max. 40%.
Batterien	3×12 VDC
Größe (L×B×H)	50 cm×49 cm×26 cm
Gewicht	12 kg
Nutzlast	30 kg

Tabelle 6.1: Technische Daten des Roboters Pioneer 2AT

Abbildung 6.2: Pioneer 2 AT

Der Laserscanner

Reichweite	max. 80 m
Scanwinkel	max. 180°
Messauflösung	10 mm
Winkelauflösung	0, 25°/ 0, 5°/ 1° (einstellbar)
Versorgungspannung	24 VDC ± 15%
Leistungsaufname	ca. 20 W
Größe (B×H×T)	155 mm×185 mm×156 mm
Gewicht	4,5 kg

Tabelle 6.2: Technische Daten des Laserscanners SICK LMS 291

Abbildung 6.3: SICK LMS 291

Der Pan-Tilt-Manipulator

	Pan	Tilt
Max. Winkelgeschwindigkeit [°/sec]	149	248
Auflösung [Winkelsec/Inc]	4	5
Lageerfassung [Inc/°]	894	672
Genauigkeit [°]	±0.02	±0.02
Spannung [V]	24± 1	
Max. Strom [A]	15	
Gewicht [kg]	3,4	

Tabelle 6.3: Technische Daten AMTEC PowerCube Wrist 90

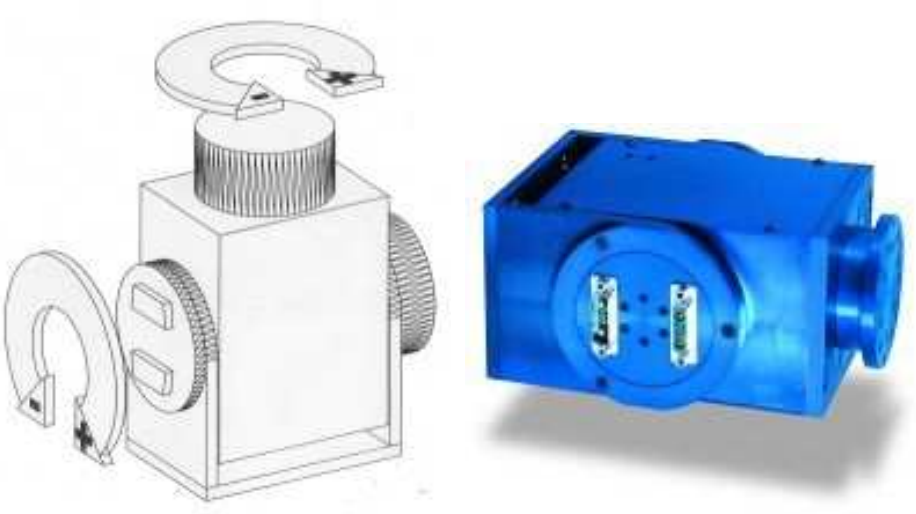

Abbildung 6.4: AMTEC PowerCube Wrist 90

Der Rechner

Mainboard	VIA Epia MII
Prozessor	VIA C3 1000MHz
Hauptspeicher	512MB DDR266 DIMM
Abmessungen	170mm×170mm
Versorgungspannung	12 VDC
Leistungsaufname	ca. 60 W

Tabelle 6.4: Technische Daten Epia Mini ITX

Abbildung 6.5: Epia Mini ITX Mainboard

6.1.2 Datenakquisition

Abbildung 6.6 zeigt den geometrischen Aufbau des verwendeten Pioneer 2 Roboters. Dieser Roboter ist mit einem SICK Laser-Range-Finder und einem Amtec PAN-TILT-Modul ausgestattet. Die Aufnahme einer 3D-Punktmenge erfolgt durch einen Schwenk des Lasers von einem negativen Tilt-Winkel zu einem positven. Beispielsweise von -15 Grad nach +40 Grad. Die 3D Punkte berechnen sich aus den gemessenen Distanzen d und den Winkeln azimut und tilt wie in Gleichung 6.1 beschrieben wird.

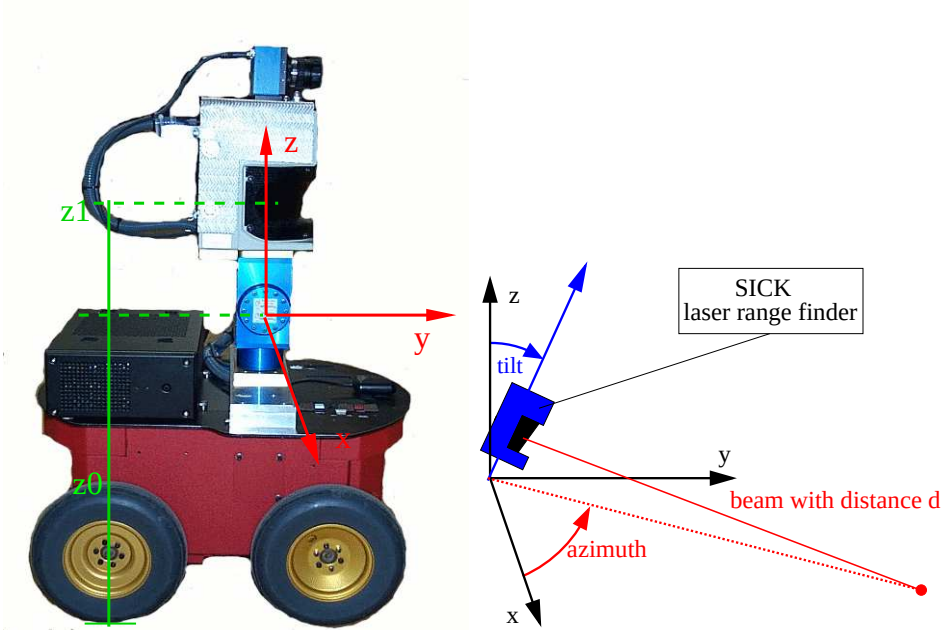

Abbildung 6.6: Roboter-Geometrie

$\alpha = azimut$
$\gamma = tilt$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \left(\frac{\pi}{2} - \gamma\right) z_1 \\ \cos\left(\frac{\pi}{2} - \gamma\right) z_1 \\ \sin\left(\frac{\pi}{2} - \gamma\right) z_1 + z_0 \end{pmatrix} + d \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) + \sin\left(\frac{\pi}{2} - \gamma\right) \\ -\sin(\alpha)\cos\left(\frac{\pi}{2} - \gamma\right) \end{pmatrix} \quad (6.1)$$

LITERATURVERZEICHNIS

[1] Allen, P., I. Stamos, A. Gueorguiev, E. Gold und P. Blaer: *AVENUE: Automated Site Modeling in Urban Environments*. In: *Proc. of the 3rd Conference on Digital Imaging and Modeling*, S. 357–364, 2001.

[2] Bares, J., M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons und W. R. L. Whittaker: *Ambler: An Autonomous Rover for Planetary Exploration*. IEEE Computer Society Press, 22(6):18–22, 1989.

[3] Besl, P. und N. McKay: *A method for registration of 3D shapes*. Transactions on Pattern Analysis and Machine Intelligence, 14(2):239–256, 1992.

[4] Chen, Y. und G. Medioni: *Object Modeling by Registration of Multiple Range Images*. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1991.

[5] D.Chetverikov, D.Svirko, D.Stepanov und P.Krsek: *The Trimmed Iterative Closest Point Algorithm*. In: *Proc. Texture 2002, the 2nd International workshop on texture analysis and synthesis, Copenhagen*, 2002.

[6] Dorai, C., J. Weng und A. K. Jain: *Optimal Registration of Object Views Using Range Data*. IEEE Trans. Pattern Anal. Mach. Intell., 19(10):1131–1138, 1997.

[7] G. Blais, M. D. L.: *Registering Multiview Range Data to Create 3D Computer Objects*. IEEE Trans. Pattern Anal. Mach. Intell., 17(8):820–824, 1995.

[8] Godin, G., M. Rioux und R. S. Baribeau: *3-D Registration Using Range Intensity Information*. In: *Proc. Videometrics III, International Symposium on Photonic Sensors Controls for Commercial Applications. Boston, Massachusetts, USA*, 1994.

[9] Greenspan, M. A. und Godin: *A Nearest Neighbor Method for Efficient ICP*. In: *Proc. of the 3rd International Conference on 3-D Digital Imaging and Modeling (3DIM01)*, 2001.

[10] Hebert, M., C. Caillas, E. Krotkov, I. Kweon und T. Kanade: *Terrain Mapping for a Roving Planetary Explorer*. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, S. 997–1002, 1989.

[11] Hygounenc, E., I.-K. Jung, P. Souères und S. Lacroix: *The Autonomous Blimp Project of LAAS-CNRS: Achievements in Flight Control and Terrain Mapping*. International Journal of Robotics Research, 23(4-5):473–511, 2004.

[12] Lacroix, S., A. Mallet, D. Bonnafous, G. Bauzil, S. F. and; M. Herrb und R. Chatila: *Autonomous Rover Navigation on Unknown Terrains: Functions and Integration*. International Journal of Robotics Research, 21(10-11):917–942, 2002.

[13] Levoy, M., K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade und D. Fulk: *The Digital Michelangelo Project: 3D Scanning of Large Statues*. In: *Proc. SIGGRAPH*, S. 131–144, 2000.

[14] Masuda, T., K. Sakaue und N. Yokoya: *Registration and Integration of Multiple Range Images for 3-D Model Construction*. In: *Proc. 13th International Conference on Pattern Recognition, Vol. 1, pp. 879-883, Vienna, Austria*, 1996.

[15] Maybeck, P.: *The Kalman Filter: An Introduction to Concepts*. In: *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[16] Moravec, H.: *Robot Spatial Perception by Stereoscopic Vision and 3D Evidence Grids*. Techn. Ber. CMU-RI-TR-96-34, Carnegie Mellon University, Robotics Institute, 1996.

[17] NEUGEBAUER, P. J., S. GROSSKOPF und H. SCHUMANN: *Geometrical Cloning of 3D Objects via Simultaneous Registration of Multiple Range Images*. In: *Proc. International Conference on Shape Modeling and Applications, Aizu-Wakamatsu, Japan*, 1997.

[18] NILSSON, N. J.: *A mobile automation: an application of artificial intelligence techniques*. In: *Proc. of the International Conference on Artificial Intelligence (IJCAI)*, 1969.

[19] OLSON, C.: *Probabilistic Self-Localization for Mobile Robots*. IEEE Transactions on Robotics and Automation, 16(1):55–66, 2000.

[20] PARRA, C., R. MURRIETA, M. DEVY und M. BRIOT: *3-D modelling and robot localization from visual and range data in natural scenes*. In: *1st International Conference on Computer Vision Systems (ICVS)*, Nr. 1542 in *LNCS*, S. 450–468, 1999.

[21] PERVÖLZ, K., A. NÜCHTER, H. SURMANN und J. HERTZBERG: *Automatic Reconstruction of Colored 3D Models*. In: *Proc. Robotik 2004*, 2004.

[22] PULLI, K.: *Multiview Registration for Large Data Sets.*. In: *3DIM*, S. 160–168, 1999.

[23] RUSINKIEWICZ, S. und M.LEVOY: *Efficient Variants of the ICP Algorithm*. In: *Proc. of the 3rd International Conference on 3-D, Digital Imaging and Modeling (3DIM01)*, 2001.

[24] SAMET, H.: *Applications of Spatial Data Structures*. Addison-Wesley Publishing Inc., 1989.

[25] SIMON, D.: *Fast and Accurate Shape-Based Registration*. Doktorarbeit, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1996.

[26] SINGH, S. und A. KELLY: *Robot Planning in the Space of Feasible Actions: Two Examples*. In: *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1996.

[27] THRUN, S., C. MARTIN, Y. LIU, D. HÄHNEL, R. EMERY MONTEMERLO, C. DEEPAYAN und W. BURGARD: *A Real-Time Expectation Maximization Algorithm for Acquiring Multi-Planar Maps of Indoor Environments with Mobile Robots*. IEEE Transactions on Robotics and Automation, 20(3):433–442, 2003.

[28] TURK, G. und M. LEVOY: *Zippered Polygon Meshes from Range Images*. In: *Proc. SIGGRAPH '94 (Orlando, Florida, July 24-29, 1994). In Computer Graphics Proceedings, Annual Conference Series, 1994, ACM SIGGRAPH, pp. 311-318*, 1994.

[29] WEIK, S.: *Registration of 3-D Partial Surface Models using Luminance and Depth Information*. In: *Proc. International Conference on Recent Advances in 3-D Digital Imaging and Modeling (3DIM '97)*, 1997.

[30] YE, C. und J. BORENSTEIN: *A New Terrain Mapping Method for Mobile Robot Obstacle Negotiation*. In: *Proc. of the UGV Technology Conference at the 2002 SPIE AeroSense Symposium*, 1994.

ABBILDUNGSVERZEICHNIS