Rheinische Friedrich-Wilhelms-Universität Bonn

Institut für Informatik III

Semantische dreidimensionale Karten für autonome mobile Roboter

Dissertation

zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Andreas Nüchter

aus Halle an der Saale

2. Mai 2006

Semantische dreidimensionale Karten für autonome mobile Roboter

Andreas Nüchter

2. Mai 2006

(AKA)

DISKI

Dissertationen zur Künstlichen Intelligenz

Mit Unterstützung des Fachbereichs 1 „Künstliche Intelligenz“ der Gesellschaft für Informatik e.V. herausgegeben von

DISKI 303

Andreas Nüchter
Kurt-Schumacher-Damm 38
D-49078 Osnabrück
Email: andreas@nuechti.de

Dissertation zur Erlangung des akademischen Grades doktor rerum naturalium (Dr. rer. nat.) vorgelegt an der Mathematisch-Naturwissenschaftliche Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Tag der Einreichung der Dissertation: 2. Mai 2006
Tag der mündlichen Prüfung: 27. September 2006

Erstgutachter: Prof. Dr. Joachim Hertzberg
Zweitgutachter: Prof. Dr. Armin B. Cremers

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn http://hss.ulb.uni-bonn.de/diss_online elektronisch publiziert

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

„infix“ ist ein Imprint der Akademischen Verlagsgesellschaft Aka GmbH

Reproduziert von einer Druckvorlage des Autors
Druck und Verarbeitung: Hundt Druck GmbH, Köln
Printed in Germany

ISSN 0941-5769
ISBN 3-89838-303-2

Zusammenfassung

Intelligentes autonomes Roboterhandeln in Alltagsumgebungen erfordert den Einsatz von 3D-Karten, in denen Objekte klassifiziert sind. 3D-Karten sind u.a. zur Steuerung notwendig, damit der Roboter komplexen Hindernissen ausweichen und sich mit 6 Freiheitsgraden (x-, y-, z-Position, Nick-, Gier-, und Rollwinkel) lokalisieren kann. Soll der Roboter mit seiner Umgebung interagieren, wird Interpretation unumgänglich. Über erkannte Objekte kann der Roboter Schlussfolgerungen ziehen, sein Wissen wird inspizier- und kommunizierbar. Aus diesen Gründen ist die automatische und schnelle semantische 3D-Modellierung der Umgebung eine wichtige Fragestellung in der Robotik. 3D-Laserscanner sind eine junge Technologie, die die Erfassung räumlicher Daten revolutioniert und Robotern das dreidimensionale Abtasten von Objekten möglich macht. Die vorliegende Arbeit untersucht und evaluiert mit Hilfe eines 3D-Laserscanners und des mobilen Roboters Kurt3D die zur automatischen semantischen 3D-Kartenerstellung notwendigen Algorithmen.

Der erste Teil der Arbeit beschäftigt sich mit der Aufgabe, 3D-Scans in einem globalen Koordinatensystem zu registrieren. Korrekte, global konsistente Modelle entstehen durch einen 6D-SLAM Algorithmus. Hierbei werden 6 Freiheitsgrade in der Roboterpose berücksichtigt, geschlossene Kreise erkannt und der globale Fehler minimiert. Die Basis des 6D-SLAM ist ein sehr schneller ICP-Algorithmus. Im zweiten Teil geht es darum, die Punktmodelle mit Semantik zu versehen. Dazu werden 3D-Flächen in einer digitalisierten 3D-Szene detektiert und interpretiert. Anschließend sucht ein effizienter Algorithmus nach Objekten und bestimmt deren Pose, ebenfalls mit 6 Freiheitsgraden. Schließlich wird der in den zahlreichen Experimenten verwendete, mobile Roboter Kurt3D vorgestellt.

Schlagwörter: Robotergestützte 3D-Kartenerstellung, 6D SLAM, semantische Kartierung, 3D-Szeneninterpretation, 3D-Objekterkennung und -lokalisierung, 3D-Laserscanner, Scanmatching, Kurt3D.

Abstract

Intelligent autonomous acting in unstructured environments requires 3D maps with labelled 3D objects. 3D maps are necessary to avoid collisions with complex obstacles and to self localize in six degrees of freedom (x-, y-, z-position, roll, yaw and pitch angle). Meaning becomes inevitable, if the robot has to interact with its environment. The robot is then able to reason about the objects; its knowledge becomes inspectable and communicable. These arguments lead to requiring automatic and fast semantic environment modelling in robotics. A revolutionary method for gaging environments are 3D scanners, which enable robots to scan objects in a non-contact way in three dimensions. The presented work examines and evaluates the algorithms needed for automatic semantic 3D map building using a 3D laser range finder and the mobile robot Kurt3D.

The first part deals with the task to register 3D scans in a common coordinate system. Correct, globally consistent models result from a 6D SLAM algorithm. Hereby 6 degrees of freedom of the robot pose are considered, closed-loops are detected and the global error is minimized. 6D SLAM is based on a very fast ICP algorithm. In the second part semantic descriptions are derived from the point model. For that purpose 3D planes are detected and interpreted in the digitalized 3D scene. After that an efficient algorithm detects objects and estimates their pose with 6 degrees of freedom, too. Finally, the mobile robot Kurt3D, that was used in numerous experiments is presented.

Keywords: robotic 3D mapping, 6D SLAM, semantic mapping, 3D scene interpretation, 3D object detection and localization, 3D laser range finder, scan matching, Kurt3D.

Geleitwort des Doktorvaters

Roboter sind dreidimensionale Wesen in einer dreidimensionalen Welt, und über diese Welt sollen sie zuweilen mit Menschen oder mit anderen Robotern kommunizieren. Müssen sie dann nicht ihre Umgebung in einer 3D-Karte abbilden, in der zudem individuelle Objekte oder Orte bezeichnet sind?

2D-Geometriekarten von Robotern erstellen lassen – das ist bereits gut verstanden. SLAM (*Simultaneous Localization and Mapping*) auf Basis von Odometrie und ebenen Laserscans ist in wenigen Jahren von einem anspruchsvollen Forschungsthema zu einem Standardverfahren auf mobilen Robotern geworden. Die resultierenden 2D-Karten repräsentieren aber ausschließlich die Umgebungsgeometrie in der Laser-Schnittebene, keine kompakten und segmentierten Objekte.

Andreas Nüchters Arbeit setzt an diesen beiden Problemen an, und wie nicht anders zu erwarten, hilft die Lösung des einen bei der Lösung des anderen. Der Haupt-Umgebungssensor ist ein 3D-Laserscanner, also eine direkte Erweiterung der 2D-Scanner, aus deren Daten üblicherweise 2D-Karten erstellt werden. Die Algorithmen sind aber nicht so direkt von 2D auf 3D zu erweitern. In einer dreidimensionalen Welt haben Körper, und also auch Roboter, im allgemeinen sechs Freiheitsgrade (drei Translationen, drei Rotationen). Der Übergang von 3D-Posen (zwei Translationen, eine Rotation) auf 6D-Posen bringt aber erheblichen Rechenaufwand mit sich – ohne weiteres wachsen zum Beispiel Wahrscheinlichkeitsverteilungen über den Poseraum exponenziell mit jeder neuen Dimension.

Der SLAM-Ansatz in dieser Arbeit, der 6D-Posen verkraftet, basiert auf dem bekannten ICP (*Iterative Closest/Corresponding Points*) Algorithmus. Andreas Nüchters Verdienst in diesem Teil der Arbeit besteht darin, ICP für die Registrierung von 3D-Punktwolken unter 6D-Posen in Echtzeit auf einem normalen Laptop lauffähig gemacht zu haben. Wer jemals versucht hat, ICP für dieses naheliegende Problem einzusetzen, der wird die sorgfältige Analyse des Problems und – vor allem! – die Lösung schätzen, die sich hier in Kapitel 3 finden. Das Verfahren funktioniert übrigens nicht nur im Labor: Es ist erprobt auf Datensätzen aus dem Freiland, aus einer Kohlemine, auf Daten von einem anderen 3D-Sensor als Laserscannern, und unter den Randbedingungen von RoboCup Rescue Wettbewerben – kurzum: Soweit Algorithmen und Software im Rahmen einer Dissertation „gehärtet“ werden können, ist das hier geschehen.

Der zweite große Teil der Arbeit handelt davon, die 3D-Karten zu *interpretieren* in dem Sinn, Objekte und Strukturen darin zu erkennen. Das ist nichts anderes als das „Szenenverstehen“, das die KI von Anbeginn angestrebt hat und zu dem gerade auch in Deutschland schon früh wichtige Beiträge auf Basis von Bilddaten geleistet wurden. Zwei Aspekte aus diesem Teil der Arbeit möchte ich herausstellen. Erstens die Idee, Objekte nicht in der 3D-Repräsentation zu erkennen, sondern in 2D-Projektionen der Szene, die klassische Algorithmen aus der Bildverarbeitung anzuwenden erlauben. Zweitens den nachweislichen „Mehrwert“, den die Interpretation von Szenenelementen für die Interpretation anderer Szenenelemente wie auch für die Korrektur von Messfehlern bringt: Ist eine Fläche zum Beispiel als Fußboden erkannt, fällt es leichter, Objekte zu segmentieren und damit zu interpretieren, die darauf stehen; ist eine Fläche als Wand erkannt, erlaubt das, nahe liegende Datenpunkte als Messfehler „glattzuspachteln“. In Integration der Dateninterpretation in die Robotersteuerung schließlich kann es helfen zu wissen, was in der Szene vorkommt, um zu entscheiden, wo in einem laufenden SLAM-Prozess die vorhandene Karte erweitert werden soll.

Andreas Nüchters Ergebnisse zur Interpretation von 3D-Karten sind von stärker experimentellem Charakter als seine „abgehangenen“ Ergebnisse zur 3D-Geometriekartenerstellung. Das erstaunt nicht, handelt es sich doch bei der Interpretation um eines der *big problems* der KI. Der erste Teil der Arbeit ist allen zum Lesen empfohlen, die eine effiziente Lösung des Problems suchen, 3D-Punktwolken unter sechs Freiheitsgraden zu registrieren. Der zweite Teil der Arbeit ist für die, die neue Ansätze und Ideen für das alte Problem des Szenenverstehens suchen. Die ganze Arbeit ist für die, die sich gerade aus einer Mischung von Lösungen und der daraus entwickelten neuen Gestalt eines alten Problems inspirieren lassen wollen.

Osnabrück, im Oktober 2006 Prof. Dr. Joachim Hertzberg

Danksagung

Herzlich danken möchte ich meinem Erstgutachter Herrn Prof. Dr. Joachim Hertzberg, der meine Forschungsarbeit in jeder Hinsicht unterstützt hat. Er gewährte mir wichtige Einblicke in die interessanten Forschungsgebiete „Künstliche Intelligenz“ und „Robotik“.

Mein Dank gilt ebenfalls Frau Dr. Simone Frintrop, Herrn Dipl.-Inform. Kai Lingemann, Herrn Dr.-Ing. Hartmut Surmann und Herrn Dipl.-Ing. (FH) M.Sc. Kai Pervölz, mit denen ich die letzten Jahre zusammen geforscht habe. Ihre Bereitschaft, meine Fragen zu beantworten und fachliche Diskussionen zu führen, hat diese Arbeit durch wertvolle Hinweise und Ideen vorangetrieben.

Dank sagen möchte ich schließlich der Kommission für die Übernahme der Betreuung dieser Arbeit: Prof. Dr. A. B. Cremers, Prof. Dr. Joachim Hertzberg, Prof. Dr. Rolf Klein und Prof. Dr. Thomas Christaller.

Ein großes Dankeschön auch an Jutta Busenbender und Wolfgang Nüchter, die den Text mit viel Geduld Korrektur gelesen haben.

Osnabrück, im Mai 2006 Andreas Nüchter

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

$\mathbb{R}$	Menge der reellen Zahlen
M, D, P	Mengen von Messdaten, Teilmenge von $\mathbb{R}^3$
N_m, N_d, n	Anzahl (der Messpunkte)
$\mathbf{R}, \mathbf{H}, \mathbf{S}, \mathbf{N}, \mathbf{P},$ $\mathbf{X}, \mathbf{Y}, \mathbf{U}, \mathbf{\Lambda}, \mathbf{V}$	Matrizen reeller Zahlen, auch Rotationsmatrizen
$\mathbf{Q}, \bar{\mathbf{Q}}$	Matrixdarstellung eines Quaternions
$\mathbf{m}, \mathbf{d}, \mathbf{n}, \mathbf{p}, \mathbf{t}, \mathbf{a}, \mathbf{u}$	Vektoren reeller Zahlen (Messpunkte, Translation)
$\mathbf{c}_m, \mathbf{c}_d$	Schwerpunktvektoren
$w_{i,j}$	Gewichte $w_{i,j} \in \mathbb{R}$
$\mathbf{e}$	Einheitsvektor, d.h. Vektor entlang einer Achse des Koordinatensystems der Länge 1.
Trace $(\mathbf{A})$	Spur einer Matrix, d.h. Trace $(\mathbf{A}) = a_{1,1} + \ldots + a_{n,n}$. Daher gilt: $\mathbf{a}^T\mathbf{R}\mathbf{b} =$ Trace $\left(\mathbf{R}^T ab^T\right)$.
$e(x, z, \theta)$	Fehlerfunktion zur Evaluation der Scanmatchinggenauigkeit
$S_m(.)$	Raumregion $\subset \mathbb{R}^3$
$v_m(.)$	Raumvolumen $\subset \mathbb{R}^3$
$u_m(.)$	Wahrscheinlichkeitsverteilung
$\mathbf{P}_{\text{robot}}$	Pose des Roboters mit 6 Freiheitsgraden (3 für die Translation, 3 für die Rotation)

$$\mathbf{P}_{\text{robot}} = \left(\begin{array}{c|c} \mathbf{R}_{\text{robot}} & \mathbf{0} \\ \hline \mathbf{t}_{\text{robot}} & 1 \end{array}\right)$$

$$= \left(\begin{array}{ccc|c} \cos\theta_y \cos\theta_z & \sin\theta_x \sin\theta_y \cos\theta_z + \cos\theta_x \sin\theta_z & -\cos\theta_x \sin\theta_y \cos\theta_z + \sin\theta_x sin\theta_z & \mathbf{0} \\ -\cos\theta_y \sin\theta_z & -\sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z & \cos\theta_x \sin\theta_y \sin\theta_z + \sin\theta_x \cos\theta_z & \\ \sin\theta_y & -\sin\theta_x \cos\theta_y & \cos\theta_x \cos\theta_y & \\ \hline x & y & z & 1 \end{array}\right)$$

mit der Rotation um die Koordinatensystemachsen $\theta_x, \theta_y, \theta_z$ und der Position des Roboters x, y, z.

$\mathcal{E}$	3D-Fläche
$X(\mathcal{E}, \varepsilon)$	Menge von 3D-Punkten, die durch die Ebene $\mathcal{E}$ und durch die ε-Umgebung erfasst sind
$\mathcal{P}$	Menge der 3D-Flächen
$\mathcal{R}$	Menge der Relationen für die Szeneninterpretation
$\mathcal{L}$	Menge der Bezeichnungen für die Szeneninterpretation
`P0, P1, ...`	Variablen für 3D-Flächen
p	3D-Fläche, gegeben entweder durch drei Punkte ($\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in \mathbb{R}^3$) oder in Hessescher Normalform ($\mathbf{p}_1$, $\mathbf{n} \in \mathbb{R}^3$ mit $\|\mathbf{n}\| = 1$)
$E(D, A)$	Fehlerfunktion für das kompetitive Lernen von Objekten
A	Menge von Referenzvektoren
$\mathbf{w}_{i_0}$	Referenzvektor ($\in A$)
μ, σ	Mittelwert und Standardabweichung
$\square$	Ende eines Beweises oder Beweis klar

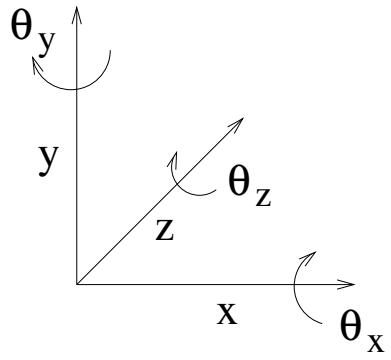


Abbildung 1: Definition des Koordinatensystems.

Kapitel 1

Einleitung

Von jeher träumten Menschen davon, künstliche Wesen zu erschaffen, die ihnen lästige oder gefährliche Arbeiten abnehmen, die sie unterhalten oder die ihrer Herrschaft unterliegen. Diese Faszination spiegelt sich in literarischen Werken wider, wo Golems, aus Lehm geschaffene künstliche Menschen, Homunkuli, auf chemischem Weg erzeugte Miniaturmenschen, oder Androiden als rein technisch konstruierte Roboter, immer wieder auftauchen [206].

Die Bezeichnung „Elektrogehirne“, die in den 50er Jahren aufkam, verweist auf den Glauben vieler Menschen, dass Computer fast schon wie Menschen denken könnten und es nicht mehr lange dauere, bis sie den Menschen in ihrer Denkleistung überflügelten. Zu dieser Zeit etablierten optimistische Computerexperten die „Künstliche Intelligenz“ als neue Forschungsrichtung. Sie sollte Computern beibringen, zu sprechen, gesprochene Sprache zu verstehen und sie in andere Sprachen zu übersetzen, wie ein Experte zu Problemen Ratschläge zu geben, Bilder und Handschriften zu erkennen, Roboter auf anderen Planeten zu steuern oder z. B. Schach zu spielen. Heute erscheint der damalige Optimismus als übertrieben, da die entwickelten Computerprogramme viele der angestrebten KI-Fähigkeiten noch nicht in überzeugender Weise erreichen [54].

Schwierigkeiten bereitet den technischen Geräten unter anderem das Sehen. In definierten Umgebungen sind sie zwar dazu in der Lage, beispielsweise zu kontrollieren, ob Flaschen richtig gespült sind, ein Werkstück die richtige, vorgegebene Form und Lage hat oder Lötstellen zu überprüfen. Dies geschieht mit Hilfe winziger Kameras und der Fähigkeit blitzschnell Bilder zu vergleichen.

Aber die normale, natürliche Umwelt stellt andere und höhere Anforderungen an das Sehen von Robotern: Hier muss das Durcheinander von Linien, Schatten, Texturen und Lichteffekten interpretiert und ausgewertet werden. Soll der Roboter sich in seiner Umwelt zielgerichtet bewegen, kommt zu der Aufgabe, aktuelle Eindrücke zu interpretieren und Objekte zu erkennen, noch ein Planungsaspekt hinzu, der ein Modell der Umgebung benötigt. Ein solches Modell stellen zum Beispiel Umgebungskarten dar. Für natürliche Umgebungen ist es unumgänglich, dass diese Umgebungskarten dreidimensional sind und der mobile Roboter mit sechs Freiheitsgraden repräsentiert wird. Komplexere Robotersteuerungen benötigen also 3D-Umgebungskarten, in denen Objekte verzeichnet sind. Damit wird das Wissen des Rotors kommunizier- und inspizierbar.

In der vorliegenden Arbeit werden die algorithmischen und technischen Methoden zur Steuerung des mobilen Roboters Kurt3D vorgestellt (vgl. Abbildung 1.1). Der Roboter ist in der Lage, seine Umgebung dreidimensional zu kartieren, Objekte zu erkennen und diese in die Karte einzutragen.

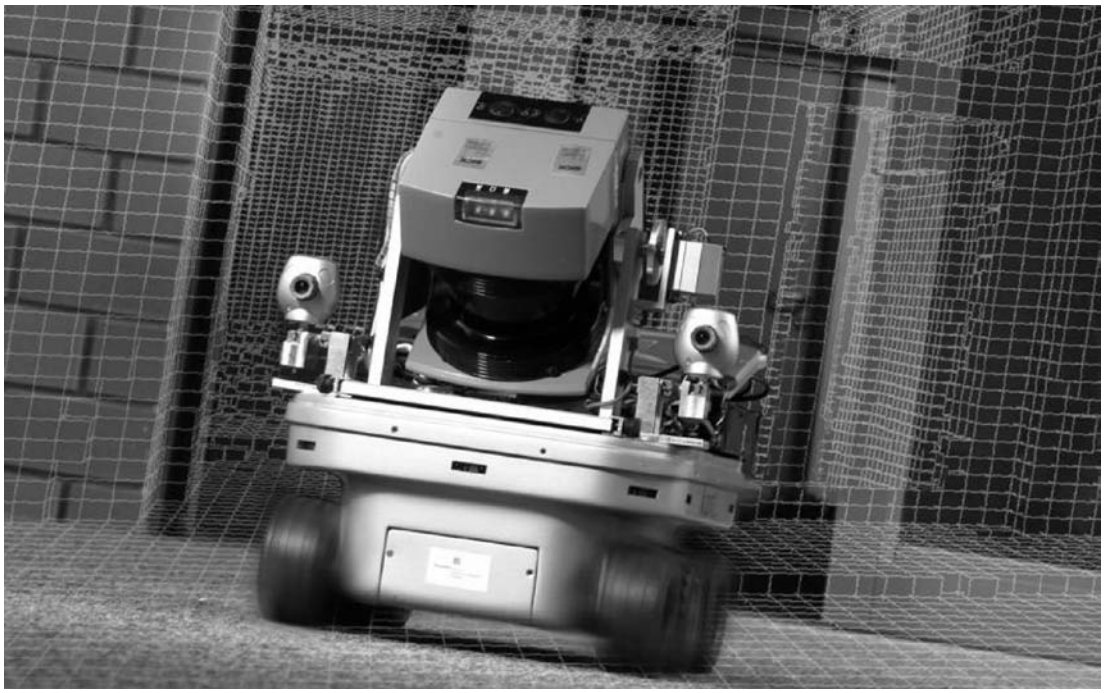


Abbildung 1.1: Der Roboter Kurt3D. Foto: Dieter Klein.

Dabei setzt Kurt3D hauptsächlich einen 3D-Laserscanner zur Wahrnehmung ein. Ein Laserscanner tastet die Umgebungsoberfläche aktiv durch das Aussenden von Lichtstrahlen ab.

Die durch den mobilen Roboter Kurt3D erstellten Karten heißen „semantische dreidimensionale Karten“. Sie sind wie folgt definiert.

Definition:

> Eine **semantische dreidimensionale Karte** für mobile Roboter ist eine metrische Karte, die neben den geometrischen Informationen zu den 3D-Messpunkten Zuordnungen von Punkten zu bekannten Strukturen oder Objektklassen der Szene enthält.

Die Einsatzmöglichkeiten eines Roboters, der seine Umgebung dreidimensional erfassen kann, sind sehr vielfältig. Zum Beispiel ist es gefährlich, Menschen in Katastrophengebiete zu schicken, um nach überlebenden zu suchen. Roboter sind kleiner, leichter und wendiger und werden in Zukunft in diesem Anwendungskontext eine große Rolle spielen. Diese und weitere Anwendungen werden zum Test der hier vorgestellten Verfahren benutzt.

1.1 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in 6 Kapitel:

Kapitel 1 ist diese Einleitung. Sie grenzt das Thema der Arbeit ab, stellt den Aufbau der Arbeit vor und legt den wissenschaftlichen Beitrag dar.

Kapitel 2 stellt Kartentypen für autonome mobile Roboter vor. Da Roboterkarten eng mit den zum Einsatz kommenden Sensorsystemen zusammenhängen, beschreibt dieses Kapitel kurz gängige Sensoren und legt dabei einen Schwerpunkt auf die Akquisition von Tiefendaten.

Kapitel 3 gibt einen der beiden Schwerpunkte dieser Arbeit wieder. Die vorgestellte Lösung des simultanen Lokalisations- und Kartierungsproblems in 6 Dimensionen (6D-SLAM) ermöglicht das Erstellen genauer dreidimensionaler Karten. Hierbei wird die Roboterpose mit 6 Freiheitsgraden berücksichtigt (x-, y- und z-Position sowie Roll-, Nick- und Gierwinkel). Zahlreiche Experimente demonstrieren die Leistungsfähigkeit des Verfahrens. Die präsentierten Anwendungen runden das Kapitel ab.

Kapitel 4 beschreibt die semantische Interpretation der in Kapitel 3 erstellten 3D-Karten. Diese Interpretation gliedert sich in zwei Teile. Zuerst werden die Grobstrukturen in der Karten analysiert und dabei Wände, Fußböden, Decken und Türen klassifiziert. Daran anschließend stellt der zweite Teil ein Verfahren vor, das in den aufgenommenen Daten nach Objekten, z.B. Stühle, Menschen oder weiteren Robotern, sucht und diese klassifiziert.

Kapitel 5 fasst die Ergebnisse der Arbeit zusammen. Zunächst wird der autonome mobile Roboter Kurt3D mit seiner Hard- und Software vorgestellt. Danach folgt die Präsentation des Verfahrens zur Aufnahme von semantischen 3D-Karten am Beispiel des AVZ-Gebäudes an der Universität Osnabrück. Die Anwendung des Roboters Kurt3D im RoboCup Rescue Wettbewerb in Lissabon 2004 komplettiert das Ergebniskapitel.

Kapitel 6 schließt die Arbeit ab, zeigt offene Probleme und gibt einen Ausblick auf künftige Arbeiten.

1.2 Wissenschaftlicher Beitrag

Mobile Roboter sind ein aktuelles Forschungsthema, da in der Automatisierungstechnik, die für den Einsatz stationärer Industrieroboter bekannt ist, nun auch eine Tendenz hin zum Einsatz mobiler Systeme besteht. Auch spielen mobile Roboter als Ausbildungsgerät für Informatik und Ingenieurwissenschaften eine große Rolle: Viele Universitäten besitzen Arbeitsgruppen mit Roboterbezug. Der wissenschaftliche Beitrag dieser Arbeit lässt sich an den zahlreichen Vorveröffentlichungen ablesen und thematisch wie folgt einordnen:

Sensorentwicklung. Die Arbeiten zur Entwicklung der in dieser Arbeit verwendeten Sensorik, d.h. des 3D-Laserscanners, wurden 2001 in [191,192] veröffentlicht. Die Idee, einen 2D-Laserscanner horizontal drehbar zu lagern, war zu diesem Zeitpunk neu.

6D SLAM. 6D SLAM wurde erstmalig in [153] und im Zusammenhang mit dem Roboter Kurt3D in [145,147,148,195] vorgestellt. Die Vorgängerarbeiten zum Registrieren mehrerer 3D-Scans – wobei der Roboter jedoch nur 3 Freiheitsgrade besaß (Ariadne Roboter mit Roboterpose als x- und z-Position und Gierwinkel) — ist in den Veröffentlichungen [144, 194] dokumentiert. Die Innovationen liegen hauptsächlich in einer sehr schnellen Implementierung der Registrierungsalgorithmen [145, 153], in einer Heuristik zur Bestimmung von Startposen [147] sowie der Systemintegration [128, 129, 131, 148, 154, 195].

Interpretation von 3D-Scans und Szenen. Die semantische Szeneninterpretation durch mobile Roboter mündete in den Papieren [150, 152]. Die Anwendung des Viola-Jones Klassifikators für die automatische Objekterkennung wurde zunächst 2004 in [149] vorgestellt und in [146] weiterentwickelt. Die Neuerungen bestanden in der Anwendung des Klassifikators auf Tiefenbilder, in der Kombination von Tiefen- und Reflektionsbild und in der genauen Modelleinpassung.

Anwendungen. Neben der direkten Anwendung der Software im RoboCup Rescue Wettbewerb [148,154] wurden Teile der Software für die 3D-Kartierung einer verlassenen Kohlemine [153] und zur Einpassung von Gesichtsprofilen [95] verwendet.

Aufmerksamkeit. Weiterhin wurden Teile der in dieser Arbeit entwickelten Verfahren für wissenschaftliche Untersuchungen im Kontext computergestützter Aufmerksamkeitsalgorithmen genutzt [85–87]. Hierbei ist der 3D-Laserscanner als bi-modale Sensorquelle interessant. Der hier vorgestellte Objekterkennung wurde ein von S. Frintrop entwickelter, neuartiger Aufmerksamkeitsfilter vorgeschaltet, um die Performanz zu erhöhen [84].

Kapitel 2

Umgebungskarten in der autonomen mobilen Robotik

Dieses Kapitel beschreibt die Sensorik zur Umgebungswahrnehmung von Robotern und legt dabei einen Schwerpunkt auf die räumliche Wahrnehmung. Karten für mobile Roboter sind eng mit der verwendeten Sensorik verknüpft. Die grundlegenden Kartentypen werden im zweiten Abschnitt vorgestellt.

2.1 Sensoren zur Umgebungswahrnehmung

2.1.1 Lasermesssysteme zur Akquisition von 3D-Daten

Ein Laserscanner bestimmt die Distanz zu einem Objekt in einer gegebenen Richtung. Er ist dadurch charakterisiert, dass die gemessene Entfernung direkt aus der Laufzeitverzögerung einer elektromagnetischen Welle, z.B. eines Laserstrahls, errechnet wird. Hierbei unterscheidet man zwei Systeme: Erstens gibt es gepulste Lasersysteme (PW), die einen Laserstrahl aussenden und die Zeit messen, bis reflektiertes Licht beim Empfänger eintrifft. Über die Lichtgeschwindigkeit c wird die Entfernung errechnet. Leutze-, Schmersal- und SICK-Scanner arbeiten nach diesem Messprinzip und werden erfolgreich in der Robotik eingesetzt [5–7]. Zweitens existieren Systeme, die mit kontinuierlichen Wellen (CW) messen. In diesem Fall wird die Phasendifferenz der ausgehenden und der reflektierten Wellen gemessen. Laserscanner, die nach diesem Prinzip arbeiten, sind in der Regel sehr präzise, aber auch teuer, so dass sie äußerst selten für Robotikanwendungen eingesetzt werden [2, 4, 9].

Neben den direkten Laserscannern gibt es außerdem Projektionsscanner. Sie projizieren ein Licht- oder Lasermuster auf die abzutastende Fläche und detektier sie mit einer Kamera. Über Triangulierung lässt sich die Tiefe des Bildpunktes bestimmen. Anwendung findet dieser Scanner u.a. bei der Digitalisierung von historischen Statuen [72, 125] oder dem Scannen von Menschen für die Modeindustrie [1].

(a) (b) (c) (d) (e) (f)

Abbildung 2.1: (a) SICK-Laserscanner [6], (b) Schmersal-Laserscanner [5], (c) CYRAX-Laserscanner [2], (d) Riegl-Laserscanner [4], (e) Zoller+Fröhlich-Laserscanner [9] und (f) Cyberware Projektionsscanner des Michelangelo-Projekts [125].

2.1.2 Weitere Sensoren zur Umgebungswahrnehmung

Neben Laserscannern kommen in der mobilen Robotik weitere bildgebende Sensoren zur Umgebungserfassung zum Einsatz. Zusätzlich zu analogen und digitalen Kameras, die hohe Bildraten erlauben, werden Entfernungskameras und thermographische Kameras eingesetzt [155]. Stereo-Sehen zur Gewinnung von Tiefendaten, sowie omnidirektionale Kameras sind auf mobilen Robotern weit verbreitet. Stereokameras haben den Nachteil, dass die erzeugten Tiefenkarten oftmals sehr ungenau und nicht dicht sind, da die Umgebung variantenreiche Texturen aufweisen muss. Technologien, die 3D-Informationen aus Umgebungsbildern mit Hilfe des optischen Flusses, von Konturen, der Texturinformation oder der Schattenwürfe errechnen, sind ebenfalls noch nicht ausgereift und finden kaum Anwendung.

Passive Sensoren registrieren ausschließlich Signale aus der Umgebung. Neben Kameras gehören Infrarotdetektoren und Bumper zu dieser Sensorkategorie. Häufig kommen jedoch aktive Sensoren zum Einsatz, zu denen auch die bereits beschriebenen Laserscanner gehören. Aktive Sensoren senden ein Signal in die Umgebung aus und registrieren das Veränderte.

Ultraschallsensoren sind in der mobilen Robotik weit verbreitet. Als Ring um den Roboter angeordnet dienen sie als Sicherheitssensoren zur Hinderniserkennung und zur Selbstlokalisation. Sie bestimmen die Distanz zu einem Objekt durch das Aussenden eines kurzen Ultraschallimpulses. Ein Empfänger detektiert die Reflektion von einem Objekt, das sich vor Sender befindet. Der Abstand kann aus der verstrichenen Zeit zwischen dem Aussenden und Empfangen des Signals errechnet werden. Falls das Signal jedoch in einem flachen Winkel auf das Objekt auftrifft, wird es nicht zum Roboter zurückreflektiert. Es entstehen Fehlmessungen.

Als Sicherheitssensoren verwenden Roboter oft auch Infrarotsensoren, die ebenfalls ringförmig am angebracht sind. Die Sensoren emmitieren einen modulierten infraroten Lichtimpuls. Gemessen wird entweder die Intensität der Reflektion an einem Objekt oder der Einfallswinkel des reflektierten Lichts (Triangulation).

Radarsensoren verwenden frequenzmodulierte elektromagnetische Wellen kleiner Wellenlänge ($f >$ 600 MHz). Ausgesandte und empfangene Wellen überlagern sich und die Differenz der Frequenzen wird bestimmt [53]. Dadurch kann der Abstand zu Objekten bestimmt werden. Radarsysteme funktionieren auch im Nebel oder im Regen [189].

Ein relativ neuer Trend in der Robotik ist der Einsatz von Entfernungsbildkameras (vgl. Abbildung

(a) (b) (c)

Abbildung 2.2: (a) CSEM Swiss Ranger [8], (b) PMDTechnologies PMDTech[Vision]19k [3], (c) Rettungsroboter des Centre for Autonomous Systems [155].

2.2). Für die Entfernungsmessung senden diese Kameras moduliertes Licht aus und messen für alle Richtungen gleichzeitig die Phasendifferenz. Die Kameras haben den gleichen Öffnungswinkel wie herkömmliche Kameras, benötigen relativ viel Energie und sind bis zu Abständen von 8 Metern einsetzbar.

Schließlich gibt es noch Sensoren bzw. Sensorsysteme zur Bestimmung der Roboterposition und -orientierung. Bei der Wahrnehmung der Roboterpose kommen im Wesentlichen zwei unterschiedliche Sensortypen zum Einsatz.

Sensoren zur lokalen Schätzung der Roboterpose. Dieser Sensortyp registriert die Poseänderung des Roboters bezüglich der Umgebung. Zu ihnen zählt Odometrie, die die Roboterpose aus den Radumdrehungen errechnet. Hierbei werden die Radumdrehungen oftmals mit analogen Widerständen oder als Encoder-Ticks mittels einer Lichtschranke gemessen. Zu diesem Sensortyp gehören außerdem Neigungssensoren, die die Orientierung des Roboters bezüglich des Gravitationsvektors, d.h. bezüglich des Vektors in Richtung des Erdmittelpunktes, messen. Gyroskope sind Intertialsensoren und schätzen ebenfalls die lokale Roboterorientierung, indem sie Beschleunigungen wahrnehmen. Über deren Integration lässt sich die Roboterpose bestimmen.

Sensoren zur globalen Lokalisierung. Der zweite Sensortyp erhebt Roboterposition in globalen Koordinaten. Zum Beispiel schätzt GPS (engl.: *global positioning system*) die Roboterposition durch eine Differenzmessung zwischen Signallaufzeiten, wobei die auf eine Trägerfrequenz aufmodulierten Codes ausgewertet werden. Drei bis zwölf geostationäre Satelliten, die NAVSTAR-Satelliten, erzeugen die Signale. DGPS (engl.: *differential global positioning system*) nutzt ein weiteres (Langwellen-)Funksignal, das von einer Stelle, deren Position man genau kennt, ausgesandt wird. Daher lässt sich der aktuelle GPS-Fehler erkennen und korrigieren. Ein weiteres Sensorsystem ist GALILEO, das europäische Äquivalent zu dem in den Vereinigten Staaten von Amerika entwickelten GPS-System. GALILEO befindet sich zurzeit im Aufbau [11, 12]. Ein weitere Sensor zur zur globalen Lokalisierung ist der Kompass, der die Orientierung zum Nordpol hin misst.

Für anwendungsspezifische Aufgaben kommen auf mobilen Robotern noch weitere Spezialsensoren zum Einsatz, wie beispielsweise Infrarotkameras, Gassensoren, Temperatursensoren oder Strahlungssensoren. Diese spielen jedoch für die metrische Umgebungskartierung keine Rolle.

2.2 Karten für mobile Roboter

Umgebungskartierung durch autonome mobile Roboter

Ein grundlegender Forschungsgegenstand im Kontext mobiler Roboter ist die Umgebungsrepräsentation. Sie bildet die Basis für Aktionen des Roboters. Ein mobiler Roboter ist in der Regel mit Sensoren zur Umgebungswahrnehmung ausgestattet. Die Eindrücke von der Umgebung, d.h. die Sensorwerte, bilden bereits eine lokale Karte. Das Kartierungsproblem besteht nun darin, aus mehreren lokalen Karten eine globale zu erzeugen (vgl. Abbildung 2.3). Diese Karte muss mit der wirklichen Umgebung übereinstimmen, also korrekt und konsistent sein. Ist die Position des Roboters genau bekannt, können die lokalen Karten auf der Grundlage dieser Position zusammengefügt werden. Leider ist die Selbstlokalisation eines Roboters stets fehlerbehaftet. Daher darf der Kartenbau nicht nur auf der Roboterposition basieren, sondern muss auch auf der Grundlage der Sensorwerte geschehen. In diesem Zusammenhang spricht man vom simultanen Lokalisations- und Kartierungsproblem (*SLAM*, engl.: *simultaneous localization and mapping problem*).

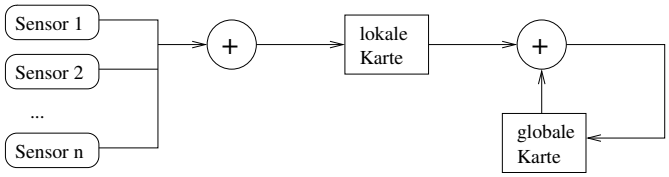


Abbildung 2.3: Lokale Karten sind eine Folge der Sensorenanordnung am Roboter. Die Integration verschiedener lokaler Karten ergibt die globale Karte.

Kartentypen. Karten für Roboter hängen eng mit den auf dem System eingesetzten Sensoren zusammen (vgl. Abbildung 2.3). Somit ergeben sich für verschiedene Sensoren unterschiedliche Klassen von Roboterkarten. Ebenso lassen sich Roboterkarten anhand ihrer Struktur klassifizieren.

Raster-Karten repräsentieren metrische Informationen über die Umgebung des Roboters in einem Belegtheitsgitter. Jede Zelle enthält die Wahrscheinlichkeit, mit der sie durch ein unbestimmtes Objekt belegt ist. Infolgedessen ist die Umgebung diskret abgebildet [198].

Terrain-Karten sind Höhenkarten. Mit jeder Zelle (x, y) ist ein Höhenwert z assoziiert. Des Weiteren sind die Standardabweichungen in z-Richtung abgelegt und verdeckte Zellen markiert. Kweon und Kanade demonstrieren im Zusammenhang mit dem Navlab Projekt Terrain-Karten zur Fahrtwegplanung. Dabei fusionieren sie Daten eines Abstandssensors mit Stereobildern [122].

Merkmal-Karten enthalten extrahierte Merkmale, beispielsweise Linien (Wände) und Ecken. Ihre Position wird zusammen mit der zugehörigen Unsicherheit abgespeichert. Durch Verfolgen der Merkmale kann die Position nachgehalten werden. So ist eine schnelle Lokalisierung möglich [124]. Merkmal-Karten können auch explizite 3D-Objekte beinhalten [65].

Hybride Raster- und Topologische Karten sind von mechanischen Federsystemen inspiriert und stellen einen topologischen Graphen dar. Einzelne Regionen bzw. Merkmale sind durch

Kanten verbunden. Diese Kanten kodieren Längen, Orientierungen und Zeiten [96]. Topologische Karten repräsentieren die Zusammenhänge zwischen Regionen.

Kognitive Karten wurden von Kuipers und Byun eingeführt [121]. Auf topologischer Ebene stellen sie die Konnektivität von Regionen dar, auf der geometrischen summieren sie anschließend metrische Informationen auf. Dadurch eliminieren sie metrische Fehler und erzielen bessere Kontroll- und Navigationsfähigkeiten des Roboters.

Graphen dienen ausschließlich der topologischen Umgebungsrepräsentation. Topologische Darstellungen sind sehr kompakt [196], ermöglichen das Schlussfolgern über Beziehungen von Orten, erlauben aber keine exakte metrische Lokalisierung mehr. Aus Raster-Karten können topologische gewonnen werden, indem man z.B. die expliziten Freiräume zusammenfasst.

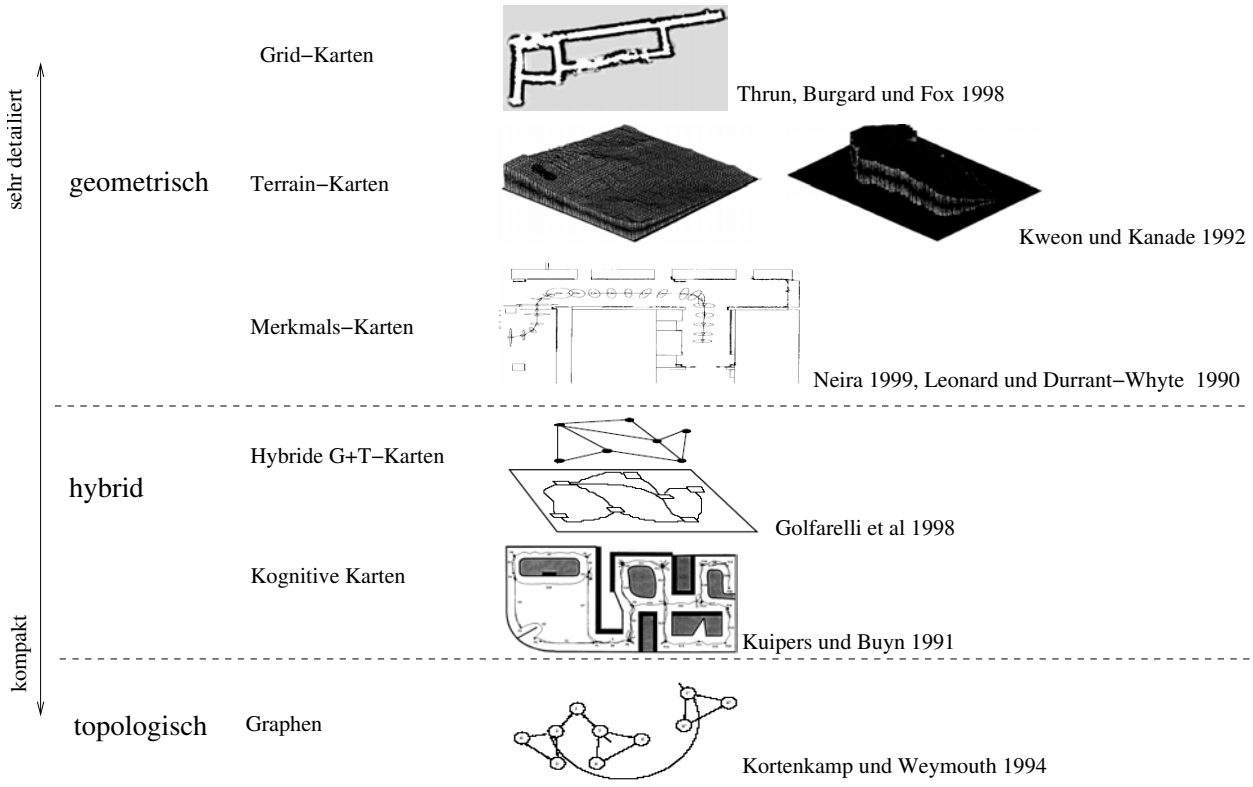


Abbildung 2.4: Kartentypen für mobile Roboter.

Robotergestützte Kartierung. Prinzipiell könnten dem Roboter Umgebungskarten für seine anwendungsspezifischen Aufgaben a priori mitgegeben werden. Leider ist das manuelle Erstellen solcher Karten sehr zeitaufwändig. Beispielsweise berichten Thrun et al., dass das Erstellen der Karte des Deutschen Museums in Bonn für den Roboter RHINO eine ganze Woche gedauert habe [196].

Die Kartierung von Umgebungen durch Roboter hat eine lange Tradition. Kartierungsalgorithmen können vergangene Informationen mit aktuellen und zukünftigen kombinieren [202]. Elfes und

Moravecs Kartierungsalgorithmus für Belegtheitsgitter ist ein frühes Beispiel für den Einzug probabilistischer Methoden in die Robotik [67, 68]. Erste Beispiele für topologische Kartierungen sind die Arbeiten von Kuipers et al. [121], Choset et al. [51] und anderen [120, 160, 216]. Seit den 90er Jahren dominieren die probabilistischen Kartierungsverfahren die Lösungsansätze für das simultane Lokalisierungs- und Kartierungsproblem. Erste Arbeiten dazu wurden von Smith et al. [182] geleistet. Seitdem laufen sie sowohl unter dem Namen SLAM als auch unter CML (engl. *concurrent mapping and localization*). Ein exzellenter Überblick findet sich in [197].

3D-Kartierung. Wenige Forschergruppen benutzen 3D-Laserscanner, um 3D-Karten aufzubauen [18, 93, 105, 179]. Hierzu gehören das RESOLV- und das AVENUE-Projekt. In beiden Projekten ist der Kartenaufbau ein Stop-and-Go-Prozess, da sich der Roboter während der Scanaufnahme nicht bewegt. Das RESOLV-Projekt zielte darauf ab, Innenräume für künstliche Realitäten und Tele-Präsenz-Anwendungen zu modellieren. Dabei kam ein RIEGL Laserscanner auf einem Roboter zum Einsatz [179]. Die Szenen wurden mit Hilfe des ICP-Scanmatching-Algorithmus [31] zusammengesetzt. Im AVENUE-Projekt wurde ein Roboter zur Modellierung urbaner Umgebungen entwickelt [18]. Das System basiert auf einem Pioneer-Roboter, einem CYRAX-Laserscanner und einem merkmalsbasierten Scanmatchingverfahren [184]. Dennoch haben Georgiev et al. in ihrer aktuellen Arbeit den 3D-Scanner nicht in die Roboterkontrollarchitektur eingebunden [93]. Die Forschergruppe um M. Hebert rekonstruiert Umgebungen mit dem Zoller+Fröhlich Scanner und versucht insbesondere, ohne initiale Poseschätzungen auszukommen [105]. Bei den genannten Ansätzen steht der Roboter während der Scanaufnahme. Die 3D-Daten im einzelnen Scan sind somit in sich konsistent.

Im Unterschied dazu nehmen Wulf et al. die 3D-Daten während der Fahrt auf [40, 211]. Dafür wenden sie einen 3D-Scanner auf der Basis eines sich kontinuierlich drehenden SICK Scanners an. Sie setzen weitere Inertialsensoren (Odometrie, Gyroskop) und einen SLAM-Algorithmus für planare Umgebungen ein, um Karten zu berechnen.

Thrun et al. [102, 199], Früh et al. [89], Zhao et al. [215] und Howard et al. [109] verwenden zwei 2D-Laserscanner, um 3D-Daten zu akquirieren. Ein Laserscanner ist horizontal am Roboter angebracht und lokalisiert ihn in der Ebene. Auf der Grundlage dieser Lokalisierung rechnet der Kartierungsalgorithmus die vom vertikal angebrachten Scanner aufgenommenen Daten in 3D-Daten um [199]. Da der vertikale Scanner nicht in der Lage ist, die Seiten von Objekten zu scannen, setzen Zhao et al. [215] zwei zusätzliche vertikale Scanner ein, die um 45° gedreht sind. Die Qualität der 3D-Karten hängt bei diesen Verfahren stark von der Selbstlokalisierung des Roboters ab.

Andere Ansätze zur 3D-Kartierung verwenden CCD-Kameras für die Umgebungswahrnehmung des Roboters [34, 62, 177]. Das Benutzen von Kameras in natürlichen Umgebungen ist wegen der sich ändernden Lichtverhältnisse jedoch schwierig. Dies hat zur Folge, dass kantenbasierte Repräsentationen gewählt werden, z.B. SIFT-Merkmale [134, 177]. Des Weiteren haben kamerabasierte Vision-Ansätze für Roboter, beispielsweise Stereo oder Structure-From-Motion, Schwierigkeiten, zuverlässig Navigations- und Kartierungsinformationen in Echtzeit zu liefern. Daher kombinieren z.B. Biber et al. SICK-Scanner-basierte 2D-Kartierungsverfahren mit einer omnidirektionalen Kamera, um 3D-Karten zu erstellen [34]. Diebel et al. nutzen zur Gewinnung von Tiefenbildern ein aktives Stereosystem, bestehend aus Kameras und einem Projektor [62].

Kapitel 3

Das Problem der gleichzeitigen Lokalisation und Kartierung in 6 Dimensionen

Dieses Kapitel beschreibt das Zusammenfügen mehrerer 3D-Scans zu einer konsistenten 3D-Karte. Die vorgestellten Algorithmen beachten alle 6 Freiheitsgrade und korrigieren die geschätzte Pose (Position und Orientierung) des Roboters.

3.1 Das Registrieren von 3D-Scans

Das vollständige Erfassen komplexer Objekte und Szenen erfordert das Scannen von mehreren Roboterposen aus. Nach dem Scanvorgang werden die aufgenommenen 3D-Scans so aneinander gefügt, dass sie die Objekte und die Szene richtig repräsentieren. Das Aneinanderfügen von Scans in einem gemeinsamen Koordinatensystem heißt Registrieren.

Die Bewegung eines Roboters auf natürlichen Untergründen vor allem außerhalb von Gebäuden muss neben der Position auch den Roll-, Gier-, und Nickwinkel berücksichtigen. Die Aufgabe der Registrierung besitzt daher 6 Freiheitsgrade. Die Pose $\mathbf{P}_{\text{Roboter}}$ enthält die 6 freien Parameter $(x, y, z, \theta_x, \theta_y, \theta_z)$ und wird durch eine Matrix angegeben (vgl. Seite xix). Ist die Position des Scanners und damit jene des Roboters genau bekannt, können die 3D-Scans auf der Grundlage dieser Position registriert werden. Leider ist die Selbstlokalisation des Roboters stets mit einem Fehler behaftet. Das Zusammenfügen der 3D-Scans darf deshalb nicht nur auf der Roboterposition basieren, sondern muss auch auf der Grundlage der 3D-Scans selbst geschehen. Letzterer Vorgang heißt Scanmatching. Für das Matching von 3D-Scans, die sich überlappen, wurden in den letzten Jahren verschiedene Verfahren entwickelt und in der Literatur vorgestellt [31, 174, 181]. Sie können folgendermaßen unterteilt werden [181]:

Matching als Optimierungsproblem. Das Registrieren als Optimierungsproblem bedeutet, eine Kostenfunktion für die Qualität eines Matchings einzuführen. Das Registrieren der 3D-Scans erfolgt über eine Suche nach derjenigen Transformation, die die Kostenfunktion minimiert. Unterschiedliche Kostenfunktionen und Transformationssuchstrategien wurden bereits erforscht [31, 133, 172, 181, 213].

Merkmalbasiertes Matching. Dieses Verfahren basiert auf der Suche nach verschiedenen Merkmalen in zwei zu registrierenden 3D-Scans. Aus der Korrespondenz zwischen gleichen Merk-

malen bestimmen die Algorithmen anschließend die Lage der Scans. Die Merkmalextraktion kann unter Umständen rechenaufwändig sein [181].

Die Kombination beider Kategorien führt darüber hinaus zu so genannten hybriden Verfahren [174]. Diesem Kapitel befasst sich mit der ersten Kategorie. Untersuchungen der zweiten für das Matching von 3D-Scans finden sich in [144, 194].

3.1.1 Matching als Optimierungsproblem

Die folgende Methode zum Registrieren von 3D-Punktmengen ist Bestandteil vieler Veröffentlichungen. Daher gibt dieser Abschnitt nur eine kurze Zusammenfassung. Der vollständige Algorithmus wurde 1991 erfunden und ist in [31, 50, 144, 213] als ICP-Algorithmus (Iterativer Algorithmus der nächsten Punkte, engl.: *Iterative Closest Points*) beschrieben.

Gegeben seien zwei unabhängig voneinander aufgenommene 3D-Punktmengen M (Modellmenge, $|M| = N_m$) und D (Datenmenge, $|D| = N_d$), die eine Oberfläche im 3D-Raum repräsentieren. Gesucht ist die Transformation, bestehend aus einer Rotation $\mathbf{R}$ und einer Translation $\mathbf{t}$, die folgende Fehlerfunktion minimiert:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t})\right|\right|^2 . \tag{3.1}$$

Den Gewichten $w_{i,j}$ wird dabei der Wert 1 zugewiesen, falls der i-te 3D-Punkt der Menge M den gleichen Punkt im Raum beschreibt wie der j-te 3D-Punkt der Menge D. Ist dies nicht der Fall, gilt $w_{i,j} = 0$. Für die Minimierung sind zwei Dinge zu berechnen: Erstens die korrespondierenden Punkte $w_{i,j}$ und zweitens die Transformation ($\mathbf{R}$, $\mathbf{t}$), die $E(\mathbf{R}, \mathbf{t})$ von den errechneten Punktkorrespondenzen ausgehend minimiert.

Der ICP-Algorithmus berechnet iterativ die Punkt-Korrespondenzen. In jedem Iterationsschritt wählt der Algorithmus für einen gegebenen 3D-Punkt der Menge D den nächsten 3D-Punkt in M als korrespondierenden Punkt. Beim Bilden dieser Korrespondenzen kommt in der implementierten Programmfassung ein Schwellwert $d_{\max}$ für den maximal zulässigen Abstand zum Einsatz. Nur wenn dieser nicht überschritten wird, bildet man das Punktpaar. Anschließend berechnet der ICP-Algorithmus die Transformation, die die Gleichung (3.1) minimiert. Die Annahme ist, dass die Punktkorrespondenzen in der letzten Iteration korrekt sind [31, 144, 194].

1. Für jeden Punkt $\mathbf{d}_i \in D$ berechne bzw. suche den am nächsten gelegenen Punkt in M. Es werden die für (3.1) benötigten $w_{i,j}$ bestimmt. Der maximal zulässige Abstand hierbei ist $d_{\max}$.
2. Berechne aus der in Schritt 1 bestimmten Korrespondenz die Transformation ($\mathbf{R}$ und $\mathbf{t}$), die die Fehlerfunktion $E(\mathbf{R}, \mathbf{t})$ (3.1) minimiert.
3. Wende die in Schritt 2 gefundene Transformation auf die Menge D an.
4. Berechne die Differenz des durchschnittlichen quadratischen Fehlers. Falls diese Differenz kleiner als ein Schwellwert ε ist, terminiere. Ansonsten gehe zu Schritt 1.

Besl und McKay präsentieren ein Konvergenz-Theorem und beweisen, dass das Verfahren der iterativen nächsten Punkte monoton einem lokalen minimalen Fehler E_{min} entgegenstrebt: Alle Schritte des ICP-Algorithmus reduzieren den nach unten beschränkten Fehler [31, 77, 144]. Offensichtlich gilt dieses Argument nicht, wenn ein maximal zulässiger Punktabstand d_{max} verwendet wird: Die Punktanzahl im Scanmatching verändert sich durch das Auswahlkriterium und somit kann der Wert der Fehlerfunktion $E(\mathbf{R}, \mathbf{t})$ (3.1) steigen.

In jeder Iteration muss im zweiten Schritt die Transformation, bestehend aus $\mathbf{R}$ und $\mathbf{t}$, die die Fehlerfunktion $E(\mathbf{R}, \mathbf{t})$ (3.1) minimiert, bestimmt werden. Es existieren verschiedene Strategien, um das Minimum zu finden, die sich in direkte und indirekte Verfahren einteilen lassen. Zu den indirekten Verfahren gehört die Simulation eines physikalischen Federsystems [66, 188]. Dabei werden die in der Funktion (3.1) aufsummierten Abstände als Federn betrachtet, die die zweite Punktmenge in die Position mit minimaler Energie bewegen. Ebenso zählen zu den indirekten Verfahren der Gradientenabstieg, die Gauss-Newton Methode und der Levenberg-Marquardt Algorithmus [31], die in den Abschnitten B.4.1 und B.4.2 des Anhangs kurz erläutert werden. Die indirekten Verfahren haben den Nachteil, dass sie für die Berechnung der Transformation Auswertungen der Funktion $E(\mathbf{R}, \mathbf{t})$ (3.1) an verschiedenen Stellen benötigen. Dadurch verbrauchen sie zwangsläufig mehr Rechenzeit als die direkten Verfahren, die nun näher ausgeführt werden:

Auflösung der Doppelsumme

Betrachtet man die Formel (3.1), stellt man fest, dass die Doppelsumme aufgelöst werden kann. Es ist:

$$\begin{aligned} E(\mathbf{R}, \mathbf{t}) &= \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t})\right|\right|^2 \\ &\propto \frac{1}{N} \sum_{i=1}^{N} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t})\right|\right|^2, \qquad \text{mit} \quad N = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \end{aligned} \tag{3.2}$$

Die Proportionalität gilt, da sich alle korrespondierenden Punkte als Tupel $(\mathbf{m}_i, \mathbf{d}_i)$ darstellen lassen.

Bemerkung: Für die Implementierung des ICP-Algorithmus wird in der Regel eine Rechnung analog (3.2) durchgeführt. Statt einer Matrix $\mathbf{W}$ für die Gewichte speichert man einen Vektor, der die Punktpaare enthält. Alle folgenden Rechenvorschriften behalten ihre Gültigkeit; der Speicheraufwand wird dabei von $\mathcal{O}(n^2)$ auf $\mathcal{O}(n)$ reduziert.

Die Minimierung der Fehlerfunktion des ICP-Algorithmus in geschlossener Form

Vier Algorithmen sind zurzeit bekannt, die die Fehlerfunktion des ICP-Algorithmus in geschlossener Form minimieren [133]. Die Schwierigkeit bei der Minimierung besteht darin, die Orthonormalitätsbedingung für die Rotationsmatrix $\mathbf{R}$ aufrecht zu erhalten. Im Folgenden werden diese vier Algorithmen kurz vorgestellt. Dabei trennen die ersten drei Verfahren die Berechnung der Rotation $\mathbf{R}$ von der Berechnung der Translation $\mathbf{t}$, um die Rotation separat zu bestimmen. Die Translation findet man anschließend ausgehend von der Rotation. Für diese Trennung werden aus den Matchingtupeln zwei Punktmengen M' und D' berechnet, indem von jedem Punkt der Schwerpunkt

der Punkte subtrahiert wird, die in das Matching eingegangenen sind. Es gilt

$$\mathbf{c}_m = \frac{1}{N}\sum_{i=1}^{N}\mathbf{m}_i, \qquad \mathbf{c}_d = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d}_i \tag{3.3}$$

und

$$M' = \{\mathbf{m}'_i = \mathbf{m}_i - \mathbf{c}_m\}_{1,\ldots,N}, \qquad D' = \{\mathbf{d}'_i = \mathbf{d}_i - \mathbf{c}_d\}_{1,\ldots,N}. \tag{3.4}$$

Nach dem Einsetzen der Gleichungen (3.3) und (3.4) in die Fehlerfunktion $E(\mathbf{R},\mathbf{t})$ (3.2) ergibt sich

$$E(\mathbf{R},\mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N}||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i - \underbrace{(\mathbf{t} - \mathbf{c}_m + \mathbf{R}\mathbf{c}_d)}_{=\tilde{\mathbf{t}}}||^2 \tag{3.5}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left|\left|\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i\right|\right|^2 - \frac{2}{N}\tilde{\mathbf{t}}\cdot\sum_{i=1}^{N}\left(\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i\right) + \frac{1}{N}\sum_{i=1}^{N}\left|\left|\tilde{\mathbf{t}}\right|\right|^2. \tag{3.6}$$

Um die obige Summe zu minimieren, müssen alle Terme minimiert werden. Der zweite Summand in (3.6) ist Null, da sich die Werte auf den Schwerpunkt beziehen. Der dritte Teil hat für $\tilde{\mathbf{t}} = \mathbf{0}$ bzw. $\mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d$ ein Minimum. Folglich bleibt nur der erste Summand in (3.6) übrig und die neue Fehlerfunktion lautet:

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{N}\left|\left|\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i\right|\right|^2. \tag{3.7}$$

Transformationsschätzung mittels der Singulärwertzerlegung einer Matrix. Dieses Verfahren wurde von Arun, Huang und Blostein [21] entwickelt. Die Rotation $\mathbf{R}$ wird als orthonormale 3×3 Matrix ausgedrückt.

Satz 1. Die optimale Rotation ergibt sich als $\mathbf{R} = \mathbf{V}\mathbf{U}^T$. Dabei stammen $\mathbf{V}$ und $\mathbf{U}$ aus der Singulärwertzerlegung $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$ einer Korrelationsmatrix $\mathbf{H}$. Die 3×3 Korrelationsmatrix $\mathbf{H}$ ist dabei gegeben durch

$$\mathbf{H} = \sum_{i=1}^{N}\mathbf{d}'_i\mathbf{m}'^T_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix}, \tag{3.8}$$

mit $S_{xx} = \sum_{i=1}^{N} m'_{ix}d'_{ix}$, $S_{xy} = \sum_{i=1}^{N} m'_{ix}d'_{iy}$, $\ldots$.

Beweis: Da eine Rotation längenerhaltend ist, gilt immer $||\mathbf{R}\mathbf{d}'_i||^2 = ||\mathbf{d}'_i||^2$. Mit Hilfe dieser Eigenschaft wird die Fehlerfunktion (3.7) erweitert zu

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{N}\left|\left|\mathbf{m}'_i\right|\right|^2 - 2\sum_{i=1}^{N}\mathbf{m}'_i\cdot\mathbf{R}\mathbf{d}'_i + \sum_{i=1}^{N}\left|\left|\mathbf{d}'_i\right|\right|^2.$$

Wie man leicht sieht, geht die Rotation nur in den mittleren Term ein. Um die obige Gleichung zu minimieren, genügt es also, den Ausdruck

$$\sum_{i=1}^{N}\mathbf{m}'_i\cdot\mathbf{R}\mathbf{d}'_i = \sum_{i=1}^{N}\mathbf{m}'^T_i\mathbf{R}\mathbf{d}'_i \tag{3.9}$$

zu maximieren. Unter Zuhilfenahme der Spur einer Matrix lässt sich der letzte Ausdruck zu

$$\text{Trace}\left(\sum_{i=1}^{N} \mathbf{R}\mathbf{d}_i'\mathbf{m}_\mathbf{i}'^T\right) = \text{Trace}\left(\mathbf{RH}\right) \tag{3.10}$$

umschreiben. Dazu muss die Matrix $\mathbf{H}$ wie in (3.8) definiert sein.

Lemma 1. Für jede positiv definite Matrix $\mathbf{A}\mathbf{A}^T$ und jede orthonormale Matrix $\mathbf{B}$ gilt:

$$\text{Trace}\left(\mathbf{A}\mathbf{A}^T\right) \geq \text{Trace}\left(\mathbf{B}\mathbf{A}\mathbf{A}^T\right).$$

Beweis: Sei $\mathbf{a}_i$ die i-te Spalte von $\mathbf{A}$. Damit lässt sich errechnen:

$$\begin{aligned}\text{Trace}\left(\mathbf{B}\mathbf{A}\mathbf{A}^T\right) &= \text{Trace}\left(\mathbf{A}^T\mathbf{B}\mathbf{A}\right)\\ &= \sum_{i=1}^{N} \mathbf{a}_i^T(\mathbf{B}\mathbf{a}_i^T).\end{aligned}$$

Durch die Ungleichung von Chauchy-Schwarz gilt:

$$\mathbf{a}_i^T(\mathbf{B}\mathbf{a}_i) \leq \sqrt{(\mathbf{a}_i^T\mathbf{a}_i)(\mathbf{a}_i^T\mathbf{B}^T\mathbf{B}\mathbf{a}_i)} = \mathbf{a}_i^T\mathbf{a}_i.$$

Damit ergibt sich $\text{Trace}\left(\mathbf{B}\mathbf{A}\mathbf{A}^T\right) \leq \sum_i^N \mathbf{a}_i^T\mathbf{a}_i = \text{Trace}\left(\mathbf{A}\mathbf{A}^T\right)$ und die Behauptung des Lemmas. □

Sei nun die Singulärwertzerlegung von

$$\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T,$$

wobei $\mathbf{U}$ und $\mathbf{V}$ orthonormale 3×3 Matrizen und $\mathbf{\Lambda}$ eine 3×3 Diagonalmatrix ohne negative Einträge ist. Sei jetzt

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T.$$

Offensichtlich ist $\mathbf{R}$ orthonormal und es ergibt sich

$$\begin{aligned}\mathbf{RH} &= \mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{\Lambda}\mathbf{V}^T\\ &= \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\end{aligned}$$

als eine symmetrische und positiv definite Matrix. Mit Lemma 1 gilt für jede 3×3 orthonormale Matrix $\mathbf{B}$:

$$\text{Trace}\left(\mathbf{RH}\right) \geq \text{Trace}\left(\mathbf{BRH}\right).$$

Daher ist $\mathbf{R}$ diejenige 3×3 orthonormale Matrix, die (3.10) maximiert und somit die gewünschte Rotation für (3.2) und (3.1). □

Nachdem eine Lösung für die Rotation $\mathbf{R}$ bestimmt ist, ergibt sich die Translation $\mathbf{t}$ nach Formel (3.5) als

$$\mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d. \tag{3.11}$$

Die Transformationsschätzung mit Hilfe von Orthonormal-Matrizen. Dieser Algorithmus ähnelt der ersten Methode, wurde aber unabhängig von Horn, Hilden und Negahdaripour entwickelt [108]. Wiederum wird die Korrelationsmatrix $\mathbf{H}$ nach der Gleichung (3.8) berechnet. Anschließend entwickelt man jedoch eine so genannte Polardekomposition, das heißt $\mathbf{H} = \mathbf{PS}$, wobei $\mathbf{S} = (\mathbf{H}^T\mathbf{H})^{1/2}$ ist.

Satz 2. Seien die Matrizen $\mathbf{H}$, $\mathbf{S}$ und $\mathbf{P}$ wie oben beschrieben definiert, dann ergibt sich die optimale Rotation als

$$\mathbf{R} = \mathbf{H}\left(\frac{1}{\sqrt{\lambda_1}}\mathbf{u}_1\mathbf{u}_1^T + \frac{1}{\sqrt{\lambda_2}}\mathbf{u}_2\mathbf{u}_2^T + \frac{1}{\sqrt{\lambda_3}}\mathbf{u}_3\mathbf{u}_3^T\right), \tag{3.12}$$

mit $\{\lambda_i\}$ als Eigenwerte und $\{u_i\}$ als zugehörige Eigenvektoren der Matrix $\mathbf{HH}^T$ [108].

Beweis: Die Rotation, die den Fehler in Gleichung (3.2) minimiert, entspricht derjenigen Orthonormalmatrix, die die Spur von $\mathbf{RH}$ (vgl. Formel (3.10)) maximiert. Der Algorithmus muss wiederum das Maximum finden und dabei die Orthonormalitätsbedingung aufrechterhalten. Die quadratische Matrix $\mathbf{H}$ lässt sich zerlegen in ein Produkt, bestehend aus einer Orthonormalmatrix $\mathbf{P}$ und einer positiv semidefiniten Matrix $\mathbf{S}$ [108]. Die Matrix $\mathbf{S}$ ist eindeutig bestimmt. Falls $\mathbf{H}$ nicht singulär ist, ist auch $\mathbf{P}$ eindeutig berechenbar. Somit lässt sich schreiben:

$$\mathbf{H} = \mathbf{PS}, \tag{3.13}$$

wobei

$$\mathbf{S} = (\mathbf{H}^T\mathbf{H})^{1/2}$$

die positive definite Wurzel der symmetrischen Matrix $\mathbf{H}^T\mathbf{H}$ und

$$\mathbf{P} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1/2}$$

eine Orthonormalmatrix ist. Es lässt sich überprüfen, dass $\mathbf{H} = \mathbf{PS}$, $\mathbf{S}^T = \mathbf{S}$ und $\mathbf{P}^T\mathbf{P} = \mathbb{1}$ gilt (vgl. Anhang A.1, Lemma 9).

Nun gilt es zu klären, wie sich die positiv definite Wurzel einer positiv definiten Matrix berechnet. Dazu wird die Matrix $\mathbf{H}^T\mathbf{H}$ durch die Menge ihrer Eigenwerte $\{\lambda_i\}$ und durch die korrespondierende Menge orthonormaler Eigenvektoren $\{\mathbf{u}_i\}$ dargestellt (siehe auch Anhang A.1, Lemma 5):

$$\mathbf{H}^T\mathbf{H} = \lambda_1\mathbf{u}_1\mathbf{u}_1^T + \lambda_2\mathbf{u}_2\mathbf{u}_2^T + \lambda_3\mathbf{u}_3\mathbf{u}_3^T.$$

Da $\mathbf{H}^T\mathbf{H}$ positiv definit ist, sind die Eigenwerte positiv und die Wurzel reell. Die symmetrische Matrix $\mathbf{S}$ konstruiert sich demnach wie folgt:

$$\mathbf{S} = \sqrt{\lambda_1}\mathbf{u}_1\mathbf{u}_1^T + \sqrt{\lambda_2}\mathbf{u}_2\mathbf{u}_2^T + \sqrt{\lambda_3}\mathbf{u}_3\mathbf{u}_3^T.$$

Der Beweis für den allgemeinen Fall ist in Anhang A.1, Lemma 7 gegeben. Da die Eigenvektoren orthonormal sind, gilt weiterhin

$$\mathbf{S}^2 = \lambda_1\mathbf{u}_1\mathbf{u}_1^T + \lambda_2\mathbf{u}_2\mathbf{u}_2^T + \lambda_3\mathbf{u}_3\mathbf{u}_3^T$$

und mit jedem von $\mathbf{0}$ verschiedenen Vektor $\mathbf{x}$

$$\mathbf{xSx} = \lambda_1(\mathbf{u}_1 \cdot \mathbf{x})^2 + \lambda_2(\mathbf{u}_2 \cdot \mathbf{x})^2 + \lambda_3(\mathbf{u}_3 \cdot \mathbf{x})^2 > 0.$$

Somit ist $\mathbf{S}$ positiv definit. Die Matrix $\mathbf{S}^{-1}$ lässt sich bestimmen, indem die Kehrwerte aus den Eigenwerten berechnet werden (vgl. Anhang A.1, Korollar 3):

$$\mathbf{S}^{-1} = (\mathbf{H}^T\mathbf{H})^{-1/2} = \frac{1}{\sqrt{\lambda_1}}\mathbf{u}_1\mathbf{u}_1^T + \frac{1}{\sqrt{\lambda_2}}\mathbf{u}_2\mathbf{u}_2^T + \frac{1}{\sqrt{\lambda_3}}\mathbf{u}_3\mathbf{u}_3^T.$$

Dieser Ausdruck wird benutzt, um die Orthonormalmatrix

$$\mathbf{P} = \mathbf{HS}^{-1} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1/2}$$

zu errechnen.

Um nun die Spur Trace $(\mathbf{RH})$ zu ermitteln, wendet man das oben beschriebene Verfahren an und erhält

$$\begin{aligned}\text{Trace}\,(\mathbf{RH}) =& \text{Trace}\,(\mathbf{RPS}) \\ =& \sqrt{\lambda_1}\,\text{Trace}\left(\mathbf{RPu}_1\mathbf{u}_1^T\right) + \sqrt{\lambda_2}\,\text{Trace}\left(\mathbf{RPu}_2\mathbf{u}_2^T\right) + \sqrt{\lambda_3}\,\text{Trace}\left(\mathbf{RPu}_3\mathbf{u}_3^T\right). \quad (3.14)\end{aligned}$$

Folgendes Lemma besitzt Gültigkeit für die Spur jeder Matrix:

Lemma 2. Für alle Matrizen $\mathbf{X}$ und $\mathbf{Y}$, für die die Produkte $\mathbf{XY}$ und $\mathbf{YX}$ eine quadratische Matrix ergeben, gilt $\text{Trace}\,(\mathbf{XY}) = \text{Trace}\,(\mathbf{YX})$. □

Für alle Summanden in (3.14) gilt demnach:

$$\text{Trace}\left(\mathbf{RPu}_i\mathbf{u}_i^T\right) = \text{Trace}\left(\mathbf{u}_i^T\mathbf{R}^T\mathbf{Pu}_i\right) = \text{Trace}\left(\mathbf{R}^T\mathbf{u}_i \cdot \mathbf{Pu}_i\right) = \mathbf{R}^T\mathbf{u}_i \cdot \mathbf{Pu}_i.$$

Da die Vektoren $\mathbf{u}_i$ der Konstruktion nach Einheitsvektoren und sowohl $\mathbf{P}$ als auch $\mathbf{R}$ orthonormale Matrizen sind, gilt $\mathbf{R}^T\mathbf{u}_i \cdot \mathbf{Pu}_i \leq 1$. Gleichheit ist genau dann gegeben, wenn $\mathbf{R}^T\mathbf{u}_i = \mathbf{Pu}_i$ (vgl. Anhang A.1, Lemma 4). Es folgt, dass

$$\text{Trace}\,(\mathbf{RPS}) \leq \sqrt{\lambda_1} + \sqrt{\lambda_2} + \sqrt{\lambda_3} = \text{Trace}\,(\mathbf{S})$$

gilt. Demnach wird der Maximalwert von Trace $(\mathbf{RPS})$ erreicht, wenn $\mathbf{R}^T\mathbf{P} = \mathbb{1}$ bzw. $\mathbf{R} = \mathbf{P}$ (siehe auch Anhang A.1, Lemma 6 und Korollar 1). Die Orthonormalmatrix, die in der Zerlegung (3.13) auftritt, ist somit die gesuchte Rotationmatrix und es gilt:

$$\mathbf{R} = \mathbf{P} = \mathbf{HS}^{-1} = \mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1/2}.$$

□

Die zugehörige Translation ergibt sich nun wiederum durch $\mathbf{t} = \mathbf{c}_m - \mathbf{Rc}_d$.

Die Transformationsschätzung unter Verwendung des Einheitsquaternions. Die gesuchte Transformation lässt sich auch durch ein Verfahren, das auf Einheitsquaternionen beruht,

berechnen. Diese Methode geht auf Horn zurück [107]. Das Einheitsquaternion ist eine Erweiterung der komplexen Zahlen auf 3 imaginäre Anteile [103, 104] und beschreibt eine Drehung um den Winkel θ um eine Rotationsachse $\mathbf{a} = (a_x, a_y, a_z)$. Es ist der 4-Vektor

$$\dot{\mathbf{q}} = \begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2)\, a_x \\ \sin(\theta/2)\, a_y \\ \sin(\theta/2)\, a_z \end{pmatrix}, \tag{3.15}$$

wobei $q_0 \geq 0$ und $q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1$ bzw. $\dot{\mathbf{q}} \cdot \dot{\mathbf{q}} = 1$ gilt. Des Weiteren besitzt jedes Quaternion $\dot{\mathbf{q}}$ die Matrizendarstellungen $\mathbf{Q}$ und $\bar{\mathbf{Q}}$, mit

$$\mathbf{Q} = \begin{pmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{pmatrix} \quad \text{und} \quad \bar{\mathbf{Q}} = \begin{pmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{pmatrix}. \tag{3.16}$$

Ein Vektor $\mathbf{v} = (v_x, v_y, v_z)^T \in \mathbb{R}^3$ wird durch ein Quaternion dargestellt, dessen erste Komponente 0 ist, also $\dot{\mathbf{v}} = (0, v_x, v_y, v_z)^T$. Der rotierte Vektor $\mathbf{v}_{\text{rot}}$ entsteht durch die Links-Multiplikation mit dem Quaternion $\dot{\mathbf{q}}$ und der Rechts-Multiplikation des konjugierten Quaternions $\dot{\mathbf{q}}^*$. Für Einheitsquaternionen entspricht das Konjugieren dem Invertieren [144]. Für $\mathbf{v}_{\text{rot}}$ gilt:

$$\dot{\mathbf{v}}_{\text{rot}} = \dot{\mathbf{q}}\dot{\mathbf{v}}\dot{\mathbf{q}}^* = (\mathbf{Q}\dot{\mathbf{v}})\dot{\mathbf{q}}^* = \mathbf{Q}^T(\mathbf{Q}\dot{\mathbf{v}}) = (\mathbf{Q}^T\mathbf{Q})\dot{\mathbf{v}}. \tag{3.17}$$

Eine 3×3 Rotationsmatrix $\mathbf{R}$ lässt sich aus einem Einheitsquaternion nach dem folgenden Schema berechnen [31, 144]:

$$\mathbf{R} = \begin{pmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y + q_z q_0) & 2(q_x q_z + q_y q_0) \\ 2(q_x q_y - q_z q_0) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_x q_0) \\ 2(q_x q_z + q_y q_0) & 2(q_y q_z + q_x q_0) & 1 - 2(q_x^2 + q_y^2) \end{pmatrix}. \tag{3.18}$$

Satz 3. Die Rotation, ausgedrückt als Einheitsquaternion $\dot{\mathbf{q}}$, die (3.1) minimiert, entspricht dem größten Eigenwert der Kreuz-Kovarianz-Matrix

$$\mathbf{N} = \begin{pmatrix} (S_{xx} + S_{yy} + S_{zz}) & (S_{yz} + S_{zy}) & (S_{zx} + S_{xz}) & (S_{xy} + S_{yx}) \\ (S_{yz} + S_{zy}) & (S_{xx} - S_{yy} - S_{zz}) & (S_{xy} + S_{yx}) & (S_{zx} + S_{xz}) \\ (S_{zx} + S_{xz}) & (S_{xy} + S_{yx}) & (-S_{xx} + S_{yy} - S_{zz}) & (S_{yz} + S_{zy}) \\ (S_{xy} + S_{yx}) & (S_{yz} + S_{zy}) & (S_{zx} + S_{xz}) & (-S_{xx} - S_{yy} + S_{zz}) \end{pmatrix}, \tag{3.19}$$

mit $S_{xx} = \sum_{i=1}^{N_m} m'_{ix} d'_{ix}$, $S_{xy} = \sum_{i=1}^{N} m'_{ix} d'_{iy}$, ...

Beweis: Analog zur Argumentation aus den beiden bereits aufgeführten Verfahren kann festgestellt werden, dass es genügt, die Summe (3.9) zu maximieren. An dieser Stelle der Herleitung kommen die Quaternionen zum Einsatz. Die Bestimmung der Rotationsmatrix $\mathbf{R}$,

die (3.9) maximiert, kann aufgefasst werden als das Finden des Einheitsquaternions $\dot{\mathbf{q}}$, das den Ausdruck

$$\sum_{i=1}^{N} \dot{\mathbf{m}}'_i \cdot (\dot{\mathbf{q}}\dot{\mathbf{d}}'_i\dot{\mathbf{q}}^*) = \sum_{i=1}^{N} (\dot{\mathbf{q}}\dot{\mathbf{m}}'_i) \cdot (\dot{\mathbf{d}}'_i\dot{\mathbf{q}})$$

maximiert (vgl. Gleichung (3.17)). Seien nun $\bar{\mathbf{M}}_i$ und $\mathbf{D}_i$ die Matrizen zu den Quaternionen $\dot{\mathbf{m}}'_i$ und $\dot{\mathbf{d}}'_i$ analog zu (3.16). Nach (3.17) wird die zu maximierende Summe

$$\sum_{i=1}^{N} (\bar{\mathbf{M}}_i\dot{\mathbf{q}}) \cdot (\mathbf{D}_i\dot{\mathbf{q}})$$

bzw.

$$\sum_{i=1}^{N} \dot{\mathbf{q}}^T\bar{\mathbf{M}}_i\mathbf{D}_i^T\dot{\mathbf{q}} \;=\; \dot{\mathbf{q}}^T \left(\sum_{i=1}^{N} \bar{\mathbf{M}}_i\mathbf{D}_i^T \right) \dot{\mathbf{q}}.$$

Schreibt man nun die Matrizen für $\bar{\mathbf{M}}_i$ und $\mathbf{D}_i$, ergibt eine Rechnung:

$$\begin{aligned}
& \dot{\mathbf{q}}^T \left(\sum_{i=1}^{N} \bar{\mathbf{M}}_i\mathbf{D}_i^T \right) \dot{\mathbf{q}} \\
= \; & \dot{\mathbf{q}}^T \left(\sum_{i=1}^{N} \begin{pmatrix} 0 & -m'_{ix} & -m'_{iy} & -m'_{iz} \\ m'_{ix} & 0 & m'_{iz} & -m'_{iy} \\ m'_{iy} & -m'_{iz} & 0 & m'_{ix} \\ m'_{iz} & m'_{iy} & -m'_{ix} & 0 \end{pmatrix} \begin{pmatrix} 0 & -d'_{ix} & -d'_{iy} & -d'_{iz} \\ d'_{ix} & 0 & -d'_{iz} & d'_{iy} \\ d'_{iy} & d'_{iz} & 0 & -d'_{ix} \\ d'_{iz} & -d'_{iy} & d'_{ix} & 0 \end{pmatrix}^T \right) \dot{\mathbf{q}} \\
= \; & \dot{\mathbf{q}}^T\mathbf{N}\dot{\mathbf{q}}. \qquad (3.20)
\end{aligned}$$

Die Matrix $\mathbf{N}$ entsteht bei der Multiplikation. Folgendes Lemma zeigt, wie das Einheitsquaternion, das (3.16) maximiert, bestimmt wird:

Lemma 3. Das Einheitsquaternion $\dot{\mathbf{q}}$ (es gilt $\dot{\mathbf{q}}\cdot\dot{\mathbf{q}} = 1$), das den Term $\dot{\mathbf{q}}^T\mathbf{N}\dot{\mathbf{q}}$ (3.20) maximiert, ist der Eigenvektor zu dem größten positiven Eigenwert der Matrix $\mathbf{N}$ [107].

Beweis: Die symmetrische Matrix $\mathbf{N}$ ist eine 4×4 Matrix. Das bedeutet, dass $\mathbf{N}$ vier Eigenwerte besitzt $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$. Zu diesen Eigenwerten können vier Eigenvektoren $(\dot{\mathbf{e}}_1, \dot{\mathbf{e}}_2, \dot{\mathbf{e}}_3, \dot{\mathbf{e}}_4)$ konstruiert werden, so dass

$$\mathbf{N}\dot{\mathbf{e}}_i = \lambda_i\dot{\mathbf{e}}_i \qquad \text{für} \quad i = 1, 2, 3, 4 \qquad \text{gültig ist.}$$

Die Eigenvektoren spannen einen vierdimensionalen Vektorraum auf (sie sind linear unabhängig); also kann ein beliebiges Quaternion $\dot{\mathbf{q}}$ als Linearkombination

$$\dot{\mathbf{q}} = \alpha_1\dot{\mathbf{e}}_1 + \alpha_2\dot{\mathbf{e}}_2 + \alpha_3\dot{\mathbf{e}}_3 + \alpha_4\dot{\mathbf{e}}_4$$

dargestellt werden. Da Eigenvektoren orthogonal sind, gilt:

$$\dot{\mathbf{q}} \cdot \dot{\mathbf{q}} = \alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2.$$

Dies muss gleich 1 sein, da nach einem Einheitsquaternion $\dot{\mathbf{q}}$ gesucht wird. Weiterhin gilt

$$\mathbf{N}\dot{\mathbf{q}} = \alpha_1\lambda_1\dot{\mathbf{e}}_1 + \alpha_2\lambda_2\dot{\mathbf{e}}_2 + \alpha_3\lambda_3\dot{\mathbf{e}}_3 + \alpha_4\lambda_4\dot{\mathbf{e}}_4,$$

da $\dot{\mathbf{e}}_1, \dot{\mathbf{e}}_2, \dot{\mathbf{e}}_3, \dot{\mathbf{e}}_4$ Eigenvektoren von $\mathbf{N}$ sind. Daraus lässt sich schließen, dass

$$\dot{\mathbf{q}}^T \mathbf{N} \dot{\mathbf{q}} = \dot{\mathbf{q}} \cdot (\mathbf{N}\dot{\mathbf{q}}) = \alpha_1^2 \lambda_1 + \alpha_2^2 \lambda_2 + \alpha_3^2 \lambda_3 + \alpha_4^2 \lambda_4$$

ist. Angenommen, die Eigenwerte seien der Größe nach sortiert, d.h. $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$, dann folgt daraus die Ungleichung

$$\dot{\mathbf{q}}^T \mathbf{N} \dot{\mathbf{q}} \leq \alpha_1^2 \lambda_1 + \alpha_2^2 \lambda_1 + \alpha_3^2 \lambda_1 + \alpha_4^2 \lambda_1 = \lambda_1.$$

Damit ist gezeigt, dass die quadratische Form niemals größer als der größte Eigenwert sein kann. Bei der Wahl von $\alpha_1 = 1$ und $\alpha_2 = \alpha_3 = \alpha_4 = 0$ wird das Maximum erreicht und das gesuchte Einheitsquaternion ist $\dot{\mathbf{q}} = \dot{\mathbf{e}}_1$ [107]. □

□

Nach der Berechnung der Rotationsmatrix $\mathbf{R}$ ergibt sich die Translation als $\mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d$.

Die Transformationsschätzung mit Dualquaternion. Die gesuchte Transformation lässt sich durch einen Algorithmus, der auf Dualzahlquaternionen beruht, berechnen. Die Wissenschaftler Walker, Shao und Volz haben diese Methode 1991 entwickelt [207]. Im Gegensatz zu den drei bereits vorgestellten Verfahren bestimmen sie die gesuchte Transformation in einem Schritt und benötigen nicht die Schwerpunkte, um die Berechnung der Rotation von jener der Translation zu entkoppeln.

Das Quaternion $\dot{\mathbf{q}}$ (vgl. Formel (3.15)) ist der 4-Vektor $(q_0, q_x, q_y, q_z)^T$ bzw. das Paar $(q_0, \mathbf{q})^T$. Ein Dualquaternion $\hat{\dot{\mathbf{q}}}$ besteht aus zwei Quaternionen $\dot{\mathbf{q}}$ und $\dot{\mathbf{s}}$, das heißt

$$\hat{\dot{\mathbf{q}}} = \dot{\mathbf{q}} + \varepsilon \dot{\mathbf{s}},$$

wobei eine spezielle Multiplikationsregel für ε definiert ist. Es gilt: $\varepsilon^2 = 0$. Darüber hinaus sind die Matrixschreibweisen der Quaternionen wichtig (vgl. (3.16)):

$$\mathbf{Q} = \begin{pmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & -q_z & q_y \\ q_y & q_z & q_0 & -q_x \\ q_z & -q_y & q_x & q_0 \end{pmatrix} = \begin{pmatrix} q_0 & -\mathbf{q}^T \\ \mathbf{q} & q_0 \mathbb{1}_{3\times 3} + \mathbf{C}_\mathbf{q} \end{pmatrix}, \tag{3.21}$$

$$\bar{\mathbf{Q}} = \begin{pmatrix} q_0 & -q_x & -q_y & -q_z \\ q_x & q_0 & q_z & -q_y \\ q_y & -q_z & q_0 & q_x \\ q_z & q_y & -q_x & q_0 \end{pmatrix} = \begin{pmatrix} q_0 & -\mathbf{q}^T \\ \mathbf{q} & q_0 \mathbb{1}_{3\times 3} - \mathbf{C}_\mathbf{q} \end{pmatrix}. \tag{3.22}$$

Die Matrix $\mathbf{C}_\mathbf{q}$ ist demnach wie folgt definiert:

$$\mathbf{C}_\mathbf{q} = \begin{pmatrix} 0 & -q_z & q_y \\ q_z & 0 & -q_x \\ -q_y & q_x & 0 \end{pmatrix}. \tag{3.23}$$

Eine starre 3D-Transformation $(\mathbf{R}, \mathbf{t})$ erfüllt die Bedingungen

$$\dot{\mathbf{q}} \cdot \dot{\mathbf{q}} = 1 \qquad \text{und} \qquad \dot{\mathbf{q}} \cdot \dot{\mathbf{s}} = 0. \tag{3.24}$$

Die Rotationsmatrix $\mathbf{R}$ kann als

$$\mathbf{R} = (q_0^2 - \mathbf{q}^T\mathbf{q})\mathbb{1} + 2\mathbf{q}\mathbf{q}^T + 2q_0\mathbf{C}_\mathbf{q} \tag{3.25}$$

geschrieben werden, was äquivalent zu der Darstellung (3.18) ist. Der Translationsvektor ist $\mathbf{t} = \mathbf{p}$, wobei $\mathbf{p}$ dem Vektoranteil des Quaternions $\dot{\mathbf{p}}$ entspricht, das sich durch

$$\dot{\mathbf{p}} = \bar{\mathbf{Q}}^T\dot{\mathbf{s}} \tag{3.26}$$

bestimmt. Der erste Eintrag p_0 von $\dot{\mathbf{p}}$ ist dabei immer Null. Ein 3D-Vektor $\mathbf{v} = (v_x, v_y, v_z)^T \in \mathbb{R}^3$ wird wiederum durch ein Quaternion $\dot{\mathbf{v}} = (0, v_x, v_y, v_z)^T$ dargestellt, dessen erste Komponente 0 ist. Mit diesen Definitionen ist es einfach zu zeigen, dass folgende Gleichheit gilt:

$$\mathbf{R}\mathbf{x} + \mathbf{t} = \bar{\mathbf{Q}}^T\mathbf{Q}\dot{\mathbf{x}} + \bar{\mathbf{Q}}^T\dot{\mathbf{s}}. \tag{3.27}$$

Satz 4. Die Transformation, bestehend aus Rotation und Translation, die die Fehlerfunktion (3.1) minimiert, ist ein Eigenwertproblem einer 4×4 Matrixfunktion, die aus den Punktpaaren gebildet wird.

Beweis: Mit obigen Definitionen des Dualquaternions lässt sich die Fehlerfunktion (3.2) umschreiben:

$$E(\mathbf{R}, \mathbf{t}) = E(\dot{\mathbf{q}}, \dot{\mathbf{s}}) = \frac{1}{N}[\dot{\mathbf{q}}^T\mathbf{C}_1\dot{\mathbf{q}} + N\dot{\mathbf{s}}^T\dot{\mathbf{s}} + \dot{\mathbf{s}}^T\mathbf{C}_2\dot{\mathbf{q}} + \text{const.}] \ ,$$

wobei die Terme $\mathbf{C}_1, \mathbf{C}_2$, und const. unter Verwendung der Quaternionendarstellung $\mathbf{M}_i$ und $\mathbf{D}_i$, bzw. $\mathbf{C}_{\mathbf{m}_i}$ und $\mathbf{C}_{\mathbf{d}_i}$ für die 3D-Datenpunkte $\mathbf{m}_i$ und $\mathbf{d}_i$ (vgl. (3.21), (3.22) und (3.23)) wie folgt definiert sind:

$$\begin{aligned}
\mathbf{C}_1 &= -2\sum_{i=1}^{N}\mathbf{M}_i^T\bar{\mathbf{D}}_i = -2\sum_{i=1}^{N}\begin{pmatrix} \mathbf{C}_{\mathbf{m}_i}\mathbf{C}_{\mathbf{d}_i} + \mathbf{m}_i\mathbf{d}_i^T & -\mathbf{C}_{\mathbf{m}_i}\mathbf{d}_i \\ -\mathbf{m}_i^T\mathbf{C}_{\mathbf{d}_i} & \mathbf{m}_i^T\mathbf{d}_i \end{pmatrix}, \\
\mathbf{C}_2 &= -2\sum_{i=1}^{N}\bar{\mathbf{D}}_i - \mathbf{M}_i = -2\sum_{i=1}^{N}\begin{pmatrix} -\mathbf{C}_{\mathbf{d}_i} - \mathbf{C}_{\mathbf{m}_i} & \mathbf{d}_i - \mathbf{m}_i \\ -(\mathbf{d}_i - \mathbf{m}_i)^T & 0 \end{pmatrix}, \\
\text{const.} &= \sum_{i=1}^{N}(\mathbf{d}_i^T\mathbf{d}_i + \mathbf{m}_i^T\mathbf{m}_i).
\end{aligned}$$

Fügt man nun die Bedingungen für die Transformation (Gleichungen (3.24)) hinzu, wird die Fehlerfunktion zu

$$E(\dot{\mathbf{q}}, \dot{\mathbf{s}}) = \frac{1}{N}[\dot{\mathbf{q}}^T\mathbf{C}_1\dot{\mathbf{q}} + N\dot{\mathbf{s}}^T\dot{\mathbf{s}} + \dot{\mathbf{s}}^T\mathbf{C}_2\dot{\mathbf{q}} + \text{const.} + \lambda_1(\dot{\mathbf{q}}^T\dot{\mathbf{q}} - 1) + \lambda_2(\dot{\mathbf{s}}^T\dot{\mathbf{q}}) \ ; \tag{3.28}$$

hierbei sind λ_1 und λ_2 Lagrangesche Multiplikatoren (siehe Anhang B.5). Um das Minimum der Funktion zu bestimmen, werden die partiellen Ableitungen berechnet und diese Null gesetzt:

$$\frac{\partial E(\dot{\mathbf{q}}, \dot{\mathbf{s}})}{\partial \dot{\mathbf{q}}} = \frac{1}{N}\left[(\mathbf{C}_1 + \mathbf{C}_1^T)\dot{\mathbf{q}} + \mathbf{C}_2^T\dot{\mathbf{s}} + 2\lambda_1\dot{\mathbf{q}} + \lambda_2\dot{\mathbf{s}}\right] = 0 \tag{3.29}$$

$$\frac{\partial E(\dot{\mathbf{q}}, \dot{\mathbf{s}})}{\partial \dot{\mathbf{q}}} = \frac{1}{N}\left[2N\dot{\mathbf{s}} + \mathbf{C}_2\dot{\mathbf{q}} + \lambda_2\dot{\mathbf{q}}\right] = 0. \tag{3.30}$$

Multipliziert man Gleichung (3.30) mit $\dot{\mathbf{q}}$, ergibt sich $\lambda_2 = -\dot{\mathbf{q}}^T \mathbf{C}_2 \dot{\mathbf{q}} = 0$, weil die Matrix $\mathbf{C}_2$ schiefsymmetrisch ist. Daher ist $\dot{\mathbf{s}}$ gegeben durch:

$$\dot{\mathbf{s}} = -\frac{1}{2N}\mathbf{C}_2\dot{\mathbf{q}}. \tag{3.31}$$

Mit (3.31) substituiert in Gleichung (3.29), folgt

$$\mathbf{A}\dot{\mathbf{q}} = \lambda_1\dot{\mathbf{q}}, \qquad \text{mit} \qquad \mathbf{A} = \frac{1}{2}\left(\frac{1}{2N}\mathbf{C}_2^T\mathbf{C}_2 - \mathbf{C}_1 - \mathbf{C}_1^T\right). \tag{3.32}$$

Daher ist das Quaternion $\dot{\mathbf{q}}$ ein Eigenvektor der Matrix $\mathbf{A}$ und λ_1 ist der korrespondierende Eigenwert. Eine Substitution von (3.32) zurück in Gleichung (3.28) ergibt

$$E(\dot{\mathbf{q}}, \dot{\mathbf{s}}) = \frac{1}{N}(\text{const.} - \lambda_1).$$

Somit ist der Fehler minimal, wenn der Eigenvektor mit dem größten Eigenwert ausgewählt wird.

Nachdem das Quaternion $\dot{\mathbf{q}}$ berechnet ist, lässt sich das Quaternion $\dot{\mathbf{s}}$ nach Gleichung (3.31) bestimmen. Die gesuchte Rotation und Translation ergibt sich mit den Formeln (3.25) und (3.26). □

Lorusso et al. haben die vier Verfahren untersucht und miteinander verglichen [133]. Alle sind in etwa gleich schnell ($\mathcal{O}$(Anzahl der Punktpaare), mit ähnlichen Konstanten, optimal), besitzen ähnliche Genauigkeiten (Rechnergenauigkeit) und Stabilitäten gegenüber verrauschten Daten.

Die minimale Anzahl korrespondierender Punkte. Die Verfahren für die direkte Transformationsschätzung berechnen mit wenigen Operationen aus den korrespondierenden Punkten die Rotation $\mathbf{R}$ und Translation $\mathbf{t}$, die die Formel (3.1) minimieren. In den 9 Einträgen der Rotationsmatrix $\mathbf{R}$ stecken wegen des Orthonormalitätszwangs nur drei unabhängige Parameter: Die Rotation um die x-, y- und z-Achse. Weitere 3 Parameter müssen für die Translation $\mathbf{t}$ errechnet werden. Punktpaare in $\mathbb{R}^3$ induzieren 3 Gleichungen, weshalb theoretisch zwei Punktpaare notwendig sind, um eine optimale Transformation $(\mathbf{R}, \mathbf{t})$ zu errechnen. Da aber zwei Punkte co-linear sind, ist die Rotation nicht eindeutig bestimmt und man benötigt mindestens ein weiteres nicht co-lineares Punktepaar [107].

Korrektur der Roboterpose. Abbildung 3.1 zeigt drei Schritte des ICP-Algorithmus. Die berechneten Transformationen werden jeweils auf den zweiten, dunkleren Scan angewendet. Sind die Rotation $\mathbf{R}$ und die Translation $\mathbf{t}$, die zwei aufeinander folgende 3D-Scans ineinander überführen, bekannt, können diese auch dazu verwendet werden, die Pose des Roboters zu aktualisieren. Um effizient rechnen zu können und den so genannten Gimbal-Lock [71] zu vermeiden, wird die Roboterpose als 4×4 Matrix dargestellt, wobei die linke obere 3×3 Matrix die aktuelle Rotation darstellt. Die Translation ist durch die ersten 3 Einträge der vierten Zeile gegeben. Somit lässt sich die Aktualisierung der Roboterpose schreiben als:

$$\mathbf{P}_{\text{robot}} := \left(\begin{array}{c|c} \mathbf{R} & \mathbf{0} \\ \hline \mathbf{t} & 1 \end{array}\right) \cdot \mathbf{P}_{\text{robot}}.$$

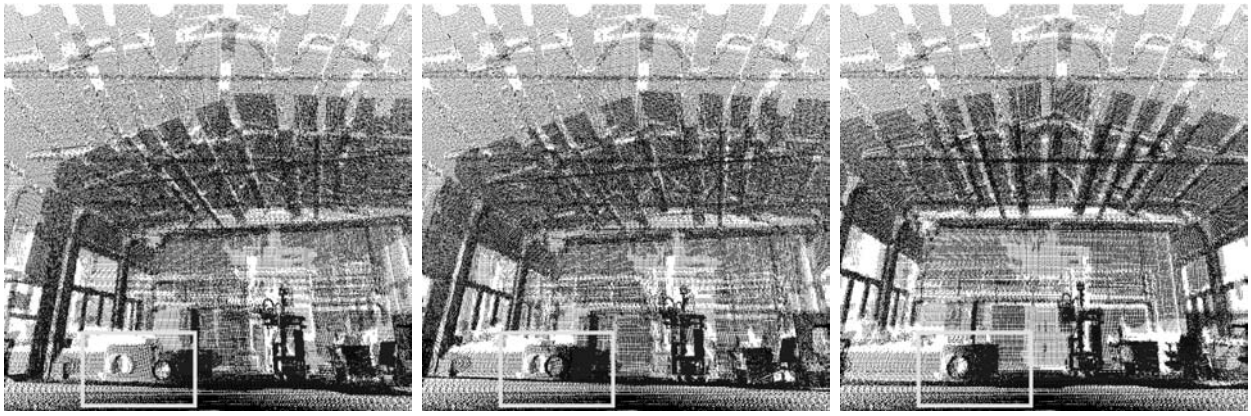


Abbildung 3.1: Visualisierung der Iterationsschritte des ICP-Algorithmus. Die anfängliche Lage zweier 3D-Scans im AiS Robotik-Pavillon sowie die Lage der 3D-Scans nach der ersten, dritten und 15. Iteration sind dargestellt. In jedem Iterationsschritt wird eine Rotation $\mathbf{R}$ und eine Translation $\mathbf{t}$ errechnet und auf den zweiten 3D-Scan angewendet.

3.1.2 Experimentelle Untersuchung des Matchings zweier 3D-Scans

Punkt-Korrespondenzen. Der erste Schritt des ICP-Algorithmus bestimmt die Punkt-Korrespondenzen $w_{i,j}$. Ein Schwellwert $d_{\max}$ für den maximal zulässigen Punktabstand kommt dabei zum Einsatz, d.h. es werden nur Punktpaare gebildet, wenn der Abstand kleiner als $d_{\max}$ ist. Setzt man $d_{\max} = \infty$, wird zu jedem Punkt in D der nächste in der Menge M bestimmt und für die Transformationsberechnung genutzt. In diesem Fall kann man nachweisen, dass die Fehlerfunktion (3.1) monoton abnimmt [31,144]. Jeweils alle Punktpaare in das Matching einzubeziehen, hat nachteilige Auswirkungen auf die Qualität des Scanmatchings: Da sich die 3D-Scans nur teilweise überlappen, werden immer auch Messpunkte einander zugeordnet, die nicht den gleichen Punkt im Raum beschreiben. Dadurch kommt es beim Minimieren von (3.1) zu einem fehlerhaften Scanmatching.

Setzt man nun einen geeignet gewählten Schwellwert $d_{\max}$ ein, so lässt sich erreichen, dass im letzten Iterationsschritt des ICP-Algorithmus nur Punkte einander zugeordnet werden, die tatsächlich den gleichen Raumpunkt beschreiben. Abbildung 3.2 zeigt die Qualität des Scanmatchings in Abhängigkeit des Schwellwertes $d_{\max}$. Für die Bestimmung der Qualität des Scanmatchings wurden zwei 3D-Scans im AiS Robotikpavillon (vgl. Abbildung 3.1) mit bekannter Pose aufgenommen. Diese Referenzpose wurde mit Hilfe eines weiteren, vom 3D-Laserscanner unabhängigen Messsystems bestimmt. Für das Experiment diente dazu ein Bandmaß. Somit ist der Referenzvergleich auf nur 3 freie Parameter (x, z, θ) beschränkt. Dennoch lässt sich die Referenzpose mit dem Ergebnis des Scanmatchings vergleichen. Folgende Funktion evaluiert die Qualität des Scanmatchings:

$$e(x, z, \theta) = ||(x, z) - (x_{\text{ref}}, z_{\text{ref}})|| + \gamma \, ||\theta - \theta_{\text{ref}}|| \,, \tag{3.33}$$

mit $x_{\text{ref}} = 120$ cm, $z_{\text{ref}} = 301$ cm und $\theta_{\text{ref}} = 45°$ der Poseänderung relativ zum ersten 3D-Scan.

Folgende Kriterien spielen bei der Wahl von $d_{\max}$ eine Rolle:

1. Die Auflösung des 3D-Scans: Der Abstand der Punkte innerhalb eines 3D-Scans hat Einfluss

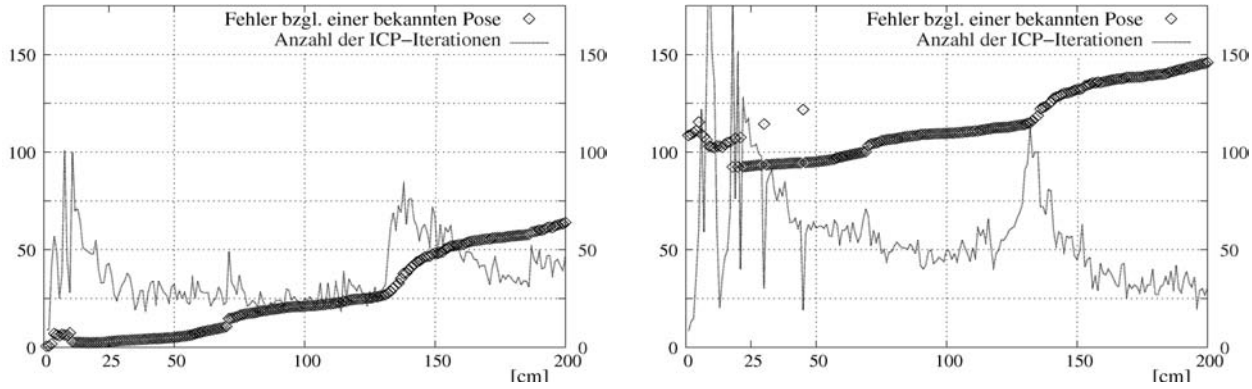


Abbildung 3.2: Qualität des Scanmatchings in Abhängigkeit des Schwellwertes $d_{\max}$. Links: Die Fehlerfunktion (3.33) und die Rechenzeit (Anzahl der ICP-Iterationen) für für die Startpose $(x, y, \theta) = (x_{\text{ref}}, y_{\text{ref}}, \theta_{\text{ref}})$. Rechts: Werte für die Startpose $(x, y, \theta) = (100\text{ cm}, 320\text{ cm}, 35°)$.

auf die Wahl von $d_{\max}$. Je gröber die Auflösung ist, desto größer sollte auch $d_{\max}$ gewählt werden. Der Schwellwert ist also auch hardwareabhängig.

2. Die anfängliche Lage der 3D-Scans im Raum spielt bei der Wahl von $d_{\max}$ eine Rolle. Ist die Startposition für das Scanmatching schlecht, kann ein großer Wert $d_{\max}$ dennoch zu einem korrekten Ergebnis führen, da somit viele Punktpaare in die Transformationsbestimmung eingehen.

3. Um ein möglichst genaues Matching zu erhalten, sollte $d_{\max}$ klein gewählt werden.

Aus diesen Punkten ergibt sich folgende Strategie, die beim Matchen von zwei 3D-Scans angewendet wird. Das Scanmatching startet mit einem großen maximalen Punktabstand, hier $d_{\max} = 25$ cm. Nach einer festen Anzahl von Iterationen (hier: 10) reduziert sich dieser Abstand auf 15 cm. Dadurch sollen auch 3D-Scans mit großen Ungenauigkeiten in der Startpose möglichst präzise registriert werden.

Ein weiterer Einfluss auf die Punktkorrespondenzen ergibt sich durch die Möglichkeit, die Korrespondenzen mittels $w_{i,j}$ zu gewichten. Folgende Gewichtungen wurden in [144] untersucht:

1. Konstante Gewichte:

$$w_{i,j} = \begin{cases} 1 & \text{, wenn } \mathbf{d}_j \text{ der nächste Punkt zu } \mathbf{m}_i \text{ ist und der Abstand dieser Punkte} \\ & \text{weniger als } d_{\max} \text{ beträgt,} \\ 0 & \text{, sonst.} \end{cases}$$

2. Gewichte in Abhängigkeit des Abstands:

$$w_{i,j} = \begin{cases} 1 - \frac{||\mathbf{m}_i - \mathbf{d}_j||}{d_{\max}} & \text{, falls } \mathbf{d}_j \text{ der nächste Punkt zu } \mathbf{m}_i \text{ ist und} \\ & ||\mathbf{m}_i - \mathbf{d}_j|| < d_{\max}, \\ 0 & \text{, sonst.} \end{cases}$$

3. Gewichte in Abhängigkeit des Reflektionswerts:

$$w_{i,j} = \begin{cases} 1 - |I(\mathbf{m}_i) - I(\mathbf{d}_j)| & \text{, falls } \mathbf{d}_j \text{ der nächste Punkt zu } \mathbf{m}_i \text{ ist und} \\ & ||\mathbf{m}_i - \mathbf{d}_j|| < d_{\max}, \\ 0 & \text{, sonst.} \end{cases}$$

Die Möglichkeit, die Punktpaare zu gewichten, hat jedoch keinen wesentlichen Einfluss auf das Scanmatching [144].

Robustheit / Einfluss der Startpose auf das Scanmatching. Um das Scanmatching von 3D-Scans erfolgreich auf Robotern einsetzen zu können, muss seine Robustheit untersucht werden, d.h. es folgt eine Untersuchung, bis zu welchem Grad zwei 3D-Scans zusammengefügt werden können. Dazu wurden zwei 3D-Scans im AiS Robotikpavillon von bekannten, unterschiedlichen Posen aus aufgenommen (vgl. Abbildung 3.1). Der zweite 3D-Scan wurde sukzessive verschoben und das Scanmatching ausgeführt. Auf diese Weise lassen sich die Abweichungen in den Posen bestimmen. Anschließend erlaubt ein Vergleich der Pose aus dem Scanmatching mit der bekannten die Evaluation des Matchings.

Die Abbildungen 3.3 und 3.4 visualisieren beispielhaft die Ergebnisse. Die Posen aller Scans jeder Grafik in Abbildung 3.3 sind um eine konstante Rotation gedreht; eingetragen sind die Fehler in der berechneten Transformation bei ablesbarer Translation in x- und z-Richtung. Die hierbei verwendete Fehlerfunktion ist wiederum (3.33).

Obwohl solche Referenzvergleiche nur 3 der 6 freien Parameter berücksichtigen, lassen sich wichtige Ergebnisse aus diesen Experimenten ableiten:

1. In der Regel ist ein Rotationsfehler schwieriger zu korrigieren als ein Fehler in der Translation. Deshalb müssen die Sensorik oder die Verfahren, die die Inititalposition für das Scanmatching liefern, möglichst genau die Rotation des 3D-Scans bestimmen. Dies gilt für alle drei Eulerwinkel gleichermaßen, da der Scanmatchingalgorithmus diese prinzipiell nicht unterscheidet.

2. Die gescannte Raumgeometrie hat entscheidenden Einfluss darauf, ob das Scanmatching korrekt ist. Führt man ein ähnliches Referenzexperiment mit 3D-Scans aus, die in engen, geraden Bürofluren aufgenommen wurden, ergeben sich die in Abbildung 3.5 aufgeführten Startposen als diejenigen, von denen aus ein korrektes Scanmatching durchgeführt werden kann. In diesem Fall ist also die Rotation nicht ganz so kritisch.

Die notwendige Rechenzeit für den ICP-Algorithmus hängt maßgeblich von der Bestimmung der nächsten Punkte ab (naive Suche $\mathcal{O}(n^2)$). Das Registrieren nimmt selbst bei kleinen Punktmengen etliche Stunden in Anspruch. Abhilfe garantieren die im Folgenden vorgestellten Algorithmen.

3.1.3 Datenreduktion für einen schnellen ICP-Algorithmus

Der erste Schritt des ICP-Algorithmus (vgl. Seite 12), die Bestimmung der Punktpaare, benötigt die meiste Rechenzeit. Bei einer naiven Suche beträgt die Komplexität $\mathcal{O}(n^2)$. Im Vergleich dazu liegt der Berechnungsaufwand des Schrittes 2 in $\mathcal{O}(n)$, da sich die Fehlerfunktion in geschlossener Form lösen lässt. Auch Schritt 3 besitzt nur eine lineare Komplexität. Die Gesamtlaufzeit wächst also analog der Suchdauer zur Bestimmung der Punktpaare.

Datenreduktion verringert den Zeitaufwand für das Matching. Verschiedene Verfahren, beispielsweise zufällige oder uniforme Punktauswahl, bieten sich für die Reduktion an. Ausgefeilte, bessere

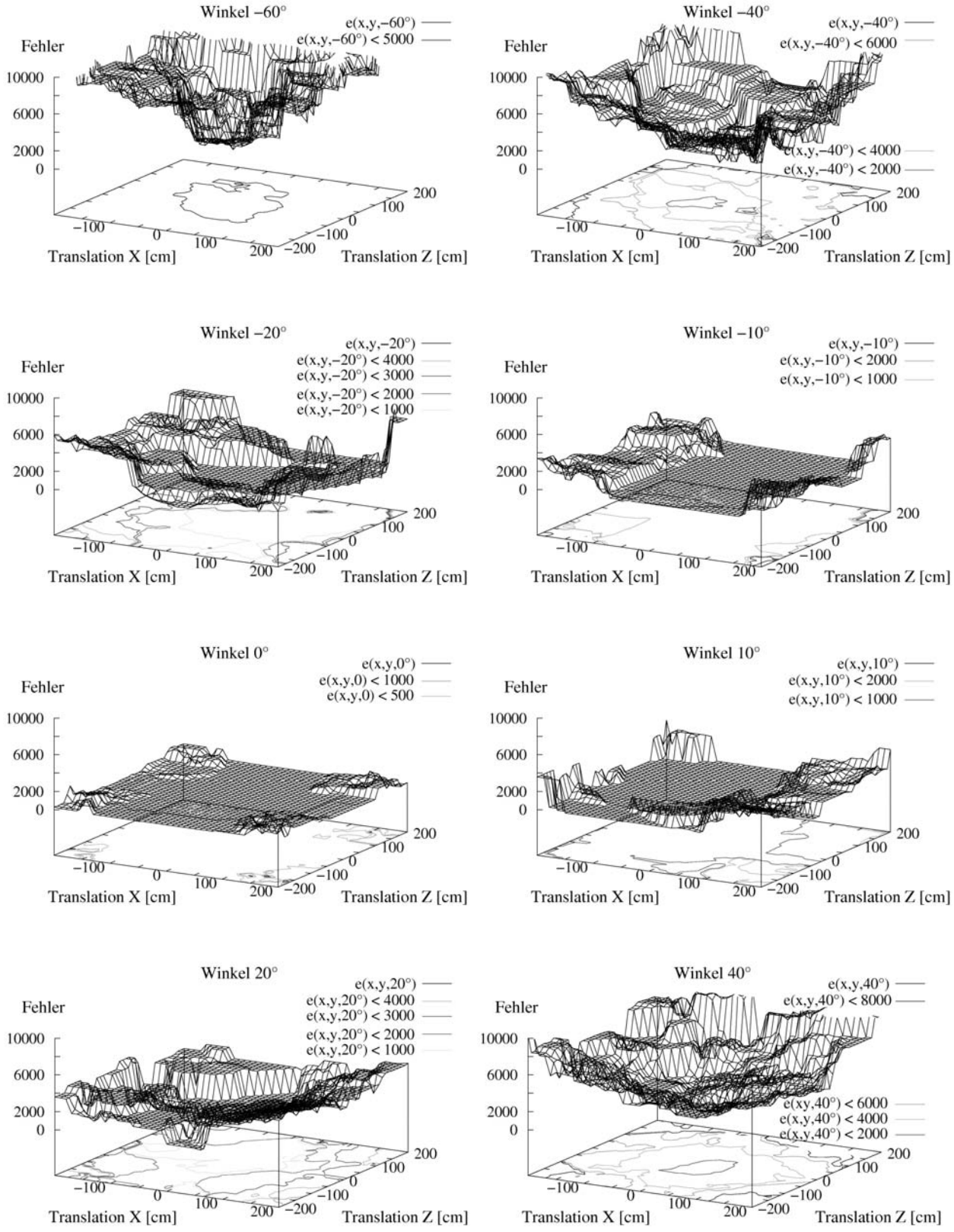


Abbildung 3.3: Der Fehler $e(x, z, \theta) = ||(x, z) - (120, 301)|| + \gamma \left|\left|\theta - \frac{\pi}{4}\right|\right|$ des Scanmatchings in Abhängigkeit der Startpose des zu matchenden 3D-Scans. Die benutzte Referenzpose $(x, y, \theta) = (120, 301, \frac{\pi}{4})$ wurde mit einem Bandmaß gemessen.

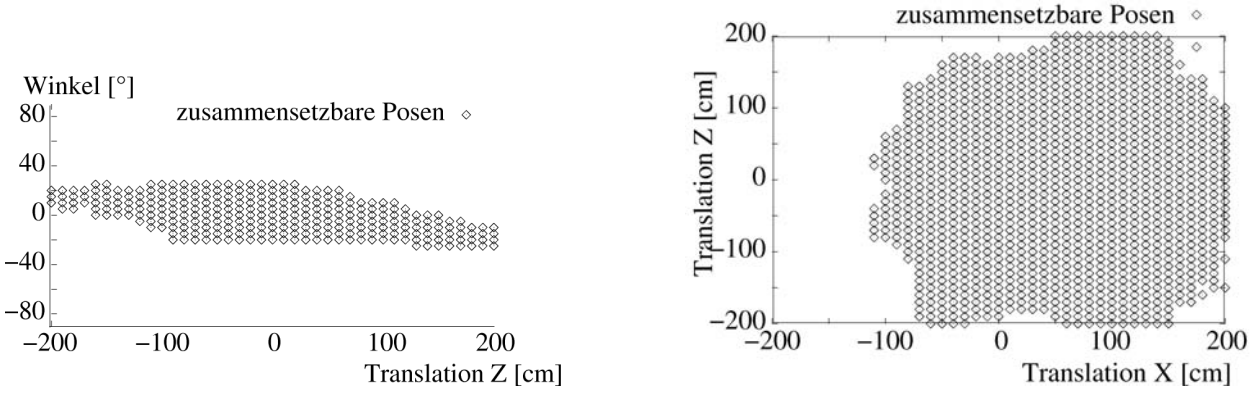


Abbildung 3.4: Startposen, die zu einem korrekten Scanmatching führen, in Zweitafelprojektion. Posen, die einen Fehler $e(x, z, \theta) < 100$ aufweisen (vgl. Formel (3.33)), sind eingezeichnet. Die Rotation hat einen wesentlichen Einfluss darauf, ob zwei 3D-Scans zusammengefügt werden können.

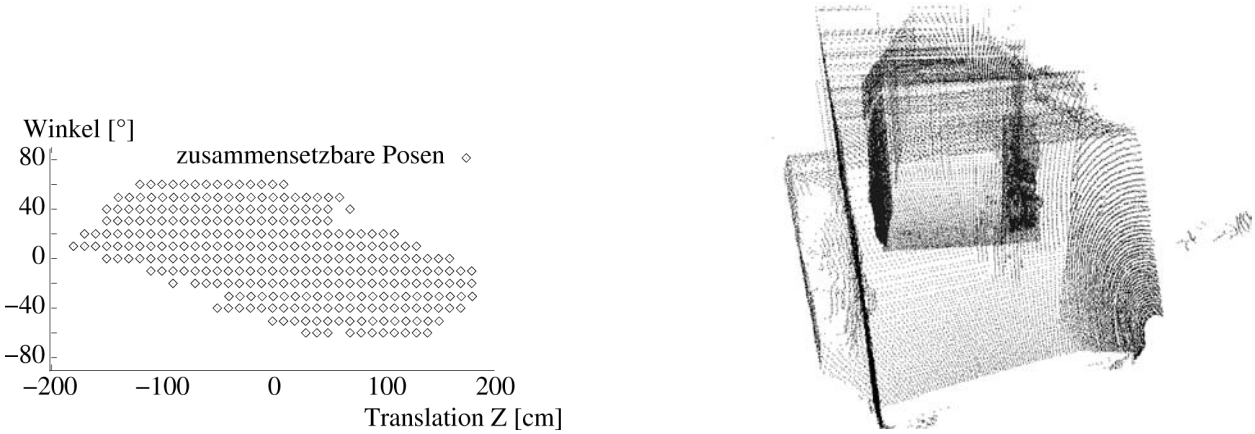


Abbildung 3.5: Links: Startposen, die zu einem korrekten Scanmatching führen, in Zweitafelprojektion. Posen, die einen Fehler $e(x, z, \theta) < 100$ aufweisen, sind eingezeichnet. Rechts: Die dazugehörigen 3D-Scans in Bürofluren.

Verfahren sind die Punktwahl unter Berücksichtigung der Normalen sowie das so genannte Kovarianzsampling [92, 172]. Das erste Auswahlverfahren selektiert Punkte so, dass die Oberflächennormalen möglichst in alle Richtungen zeigen. Beim Kovarianzsampling werden die Punktpaare zunächst uniform ausgewählt. Anschließend identifiziert der Samplingalgorithmus, der auf Levoy et al. zurückgeht, die nicht-stabilen Punktpaare, indem er eine Kovarianzmatrix abschätzt. Dadurch lässt sich verhindern, dass zwei merkmalsarme 3D-Scans aneinander vorbeigeschoben werden (sliding) [92].

Die hier vorgestellte Datenreduktion und Filterung berücksichtigen das Verfahren der Scanaufnahme: Das radiale und kontinuierliche Messen durch den Laser. Jeder Messprozess, auch das Messen mit dem 3D-Laserscanner, produziert verrauschte Daten. Außerdem können immer kleine Fehler auftreten. Bei dem 3D-Laserscanner sind jene Fehler vergleichbar mit denen, die in der Bildverarbeitung vorkommen [79]: Gauss-Rauschen und die so genannten Salz-und-Pfeffer-Fehler. Letztere

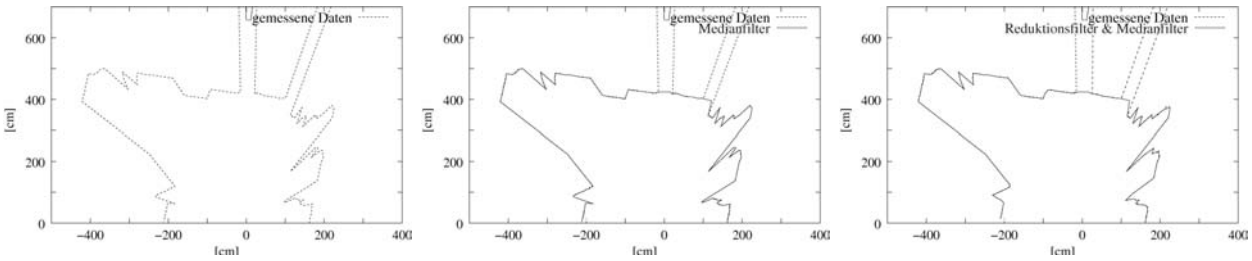


Abbildung 3.6: Anwendung des Reduktions- und Medianfilters. Der Glättungsfilter beseitigt Kantensprünge und fehlerhafte Daten. Der Reduktionsfilter reduziert die Datenmenge auf typischerweise 10%. Links: Orginaldatenpunkte. Mitte: Daten nach Anwendung des Medianfilters. Rechts: Median und Reduktionsfilterergebnis. Bemerkung: Die Punkte wurden mit Linien verbunden, um die Filtereffekte besser zu demonstrieren. Die reduzierten und gefilterten Daten beschreiben die gescannte Oberfläche nahezu perfekt.

treten zum Beispiel an Kanten auf. Da der Laserstrahl einen relativ großen Durchmesser besitzt (in 5 m zirka 10 cm), kommt es vor, dass er auf zwei Oberflächen trifft und den mittleren Entfernungswert als falschen Abstandswert ausgibt. Des Weiteren resultieren Reflexionen in falschen Entfernungswerten. Filtert man die Daten nicht, können wenige fehlerhafte Daten zu vielen falschen Punktpaaren und diese wiederum zu einem ungenauen 3D-Scanmatching führen.

Ein schneller Filter reduziert und glättet die Daten für den ICP-Algorithmus. Er besteht aus Median- und Reduktionsfilter und wird auf jede 2D-Scanebene angewandt, die 181, 361 oder 721 Messpunkte enthält [127]. Der Medianfilter entfernt die Ausreißer, die durch Salz-und-Pfeffer-Fehler auftreten. Dabei ersetzt er Datenpunkte mit dem Median von n umgebenen Punkten. Diese lassen sich effizient anhand ihrer Anordnung im 2D-Scan bestimmen, weil der Scanner die Messdaten geordnet liefert (entgegen dem Uhrzeigersinn). Der Median wird mittels euklidischer Distanz zum Scannerursprung bestimmt. Da der Filter nur Ausreißer entfernen und die anderen Messpunkte unverändert lassen soll, wird ein Messpunkt genau dann ersetzt, wenn die Differenz zwischen Median und Messpunkt größer als ein Schwellwert d_{med} ist. Der Medianfilter benutzt die Werte $n = 7$ und $d_{\text{med}} = 200$ cm.

Die Datenreduktion geschieht mit folgendem einfachen Verfahren: Der 3D-Scanner sendet Laserstrahlen radial aus, so dass eine Oberfläche, die sich in der Nähe des Scanners befindet, durch sehr viele Messpunkte abgetastet wird. Auf diese Weise lassen sich Messpunkte, die sich nahe am Scanner befinden, in einem Datenpunkt zusammenfassen. Die Anzahl der so entstandenen *reduzierten Punkte* ist in der Regel um eine Größenordnung kleiner als die Originaldatenzahl.

Im Ergebnis enthalten die 2D-Scanschnitte Punkte mit einem Mindestabstand von beispielsweise 10 cm. Es ist fast unmöglich, Unterschiede zwischen den mediangefilterten sowie reduzierten Daten und den Orginaldaten zu erkennen (vgl. Abbildung 3.6). Somit ist wie von Boulanger et al. [38] gefordert, ein wichtiges Sampling-Kriterium erfüllt, d.h. die Tiefenbilder sind so reduziert, dass die Oberflächenstruktur (Mannigfaltigkeit) erhalten bleibt. Zusätzlich zu dem beschriebenen Reduktionsfilter wird nur jeder m-te 2D-Scanschnitt für das Matching mit ICP-Algorithmus benutzt. m hängt dabei von der Scanauflösung ab. Für den 3D-Scanner typische Werte sind $m = 3, 5, 6$.

Die Algorithmen für das Glätten und Reduzieren der Daten sind als Echtzeit-Algorithmen implementiert und laufen parallel zu der Datenaufnahme ab. Die Rechenzeit der Methoden liegt unter der Zeit, in der eine Scanebene abgetastet wird (13 ms). Abbildung 3.7 zeigt einen 3D-Scan und die zugehörigen gefilterten und reduzierten Daten.

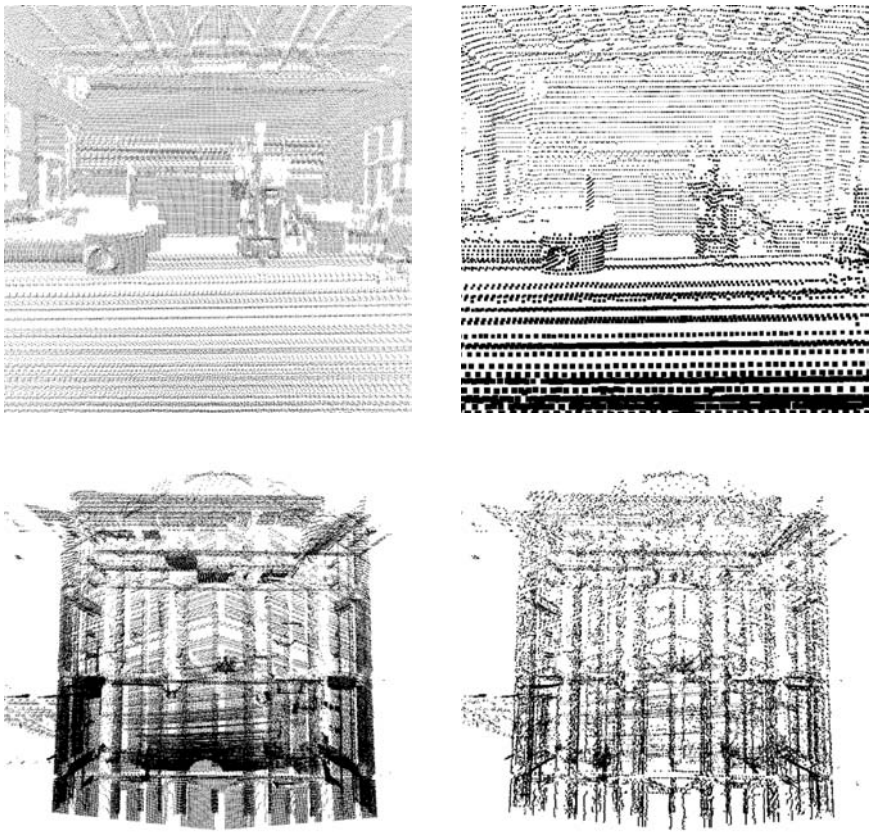


Abbildung 3.7: 3D-Scan mit reduzierten, gefilterten Daten (vgl. Abbildung 3.1). Oben: Ansicht von vorne. Unten: Ansicht von oben. Die Anzahl der Punkte wurde von 123101 (links) auf 10535 (rechts) reduziert und die Punkte sind vergrößert dargestellt.

3.1.4 Bestimmung der nächsten Punkte

Um die Rechenzeit für das Scanmatching zu reduzieren, muss die Suche nach dem nächsten Punkt in der Menge M zu einem gegebenen Punkt $\mathbf{p}_q \in D$ beschleunigt werden. Eine naive Implementierung betrachtet nacheinander alle Punkte in M. Daraus resultiert eine gesamte Rechenzeit von $\mathcal{O}(|N_D|\,|N_M|)$, also $\mathcal{O}(n^2)$. Die Datenreduktion aus dem letzten Abschnitt ändert prinzipiell nichts an diesem Laufzeitverhalten. Verschiedene Verfahren versprechen Abhilfe und lösen das so genannte Nächste-Punkt-Problem schneller. Formal gesehen besteht der Suchraum hierbei aus N Einträgen, die durch k reell-wertige Schlüssel beschrieben werden, und einem Abstandsmaß, der Euklidischen Distanz.

Binäre Suche. Im Fall $k = 1$ lässt sich die binäre Suche verwenden. Ein sortiertes Feld der Eingabewerte wird aufgestellt, in dem in $\mathcal{O}(\log n)$ Zeit im schlechtesten Fall der nächste Punkt gefunden werden kann.

Verallgemeinerte Voronoi Diagramme. Ist die Dimension $k = 2$, kann ein Voronoi Diagramm für die gegebene Punktmenge aufgestellt werden. Ein Voronoi Diagramm unterteilt die Ebene in Zellen, so dass diese Zellen den nächsten Punkt bestimmen. Ein Punkt-Zelle-Lokalisierungs-

algorithmus lässt sich benutzen, um diejenige Zelle zu finden, die den Anfragepunkt $\mathbf{p}_q$ enthält. Damit ist der nächste Punkt bestimmt.

Falls die Dimension größer als 2 ist, dann steigt die Komplexität des Voronoi-Diagramms auf $O(n^{\lceil k/2 \rceil})$.

TINN. Greenspan et al. geben eine Methode zur Bestimmung der nächsten Punkte an, die auf der Dreicksungleichung (engl.: *Triangle Inequality Nearest Neighbour*) basiert [99]. Sie benutzen einen Referenzpunkt $\mathbf{r}$. In einem Vorverarbeitungsschritt bestimmt ihr Algorithmus die Distanzen aller Punkte $\mathbf{p} \in M$ zu $\mathbf{r}$ und speichert diese zusammen mit $\mathbf{p}$ in einer sortierten Liste.

Zur Ausführungszeit wird die Distanz vom Referenzpunkt $\mathbf{r}$ zum Anfragepunkt $\mathbf{p}_q$ bestimmt. Mit Hilfe dieser Distanz wendet man eine binäre Suche in der Liste an. Startend von dem so gefundenen Kandidaten, wird die Liste in beide Richtungen durchlaufen und die Punkte $\mathbf{p}_i$ getestet, bis die Dreiecksungleichung

$$|R_q - R_i| \leq D_{qi} \leq R_q + R_i$$

für D_{qi} ein Minimum aufweist. Dabei ist R_q die euklidische Distanz des Anfragepunkts $\mathbf{p}_q$ zum Referenzpunkt $\mathbf{r}$, d.h. $R_q = ||\mathbf{p}_q - \mathbf{r}||$ [99]. Obwohl binäre Suche verwendet wird, ist die Laufzeit des Verfahrens schlechter als die Suche in mehrdimensionalen Bäumen.

Die Zellenmethode. Im Unterschied zu den in den letzten beiden Unterabschnitten erläuterten Methoden, die versuchen, die Suche für den schlechtesten Fall zu optimieren, betrachten die nun folgenden Verfahren Erwartungswerte. Sie machen Annahmen über die Wahrscheinlichkeitsverteilung der Eingabepunktmenge. Eine einfache Technik zur Lösung des Nächste-Punkt-Problems ist hier die Zellenmethode. Der k-dimensionale Raum der Schlüssel wird in kleine, gleich große, an den Koordinatenachsen ausgerichtete Zellen eingeteilt. In diesen Zellen werden die Daten in Listen gespeichert. Eine Spiralsuche in den Zellen, die von der Anfragezelle ausgeht, findet immer den nächsten Punkt [83, 99, 168].

Der Erwartungswert für die Performanz dieser Methode hängt von der angenommenen Punktverteilung ab. Ist diese uniform, lässt sich beweisen, dass die Zellenmethode optimal ist [168]. Im Wesentlichen hängt die Suchzeit von der Anzahl der Punkte in den Zellen ab. Die spiralförmige Suche um den Anfragepunkt herum kann nur beendet werden, wenn alle Zellen, die näher oder in gleicher Entfernung zum Anfragepunkt liegen, durchsucht wurden. Die Zellenmethode benötigt Speicherplatz in der Größenordnung $\mathcal{O}(d^k)$ mit d proportional zur räumlichen Ausdehnung des Problems und ist daher selten für reale Probleme geeignet.

Die folgenden Abschnitte stellen Verfahren zur Bestimmung von nächsten Punkten vor, die auf mehrdimensionalen binären Bäumen basieren. Hierbei spielt auch Approximation eine Rolle: Der nächste Punkt wird nicht exakt bestimmt, sondern näherungsweise erfasst.

kD-Bäume zur Lösung des Nächste-Punkt-Problems

kD-Bäume sind eine Generalisierung eines einfachen binären Suchbaums. Jeder Knoten eines kD-Baums repräsentiert eine Aufteilung einer Punktmenge auf zwei Nachfolgerknoten. Demnach gibt die Wurzel die gesamte Punktmenge an. Die Blätter repräsentieren eine Partition der Punktmenge in kleine, disjunkte Punktmengen. Die Mengen an den Blättern werden Buckets genannt. Des Weiteren enthält jeder Knoten des kD-Baums die Grenzen der gespeicherten Punktmenge. Eine effiziente Implementierung ist in [144] angegeben.

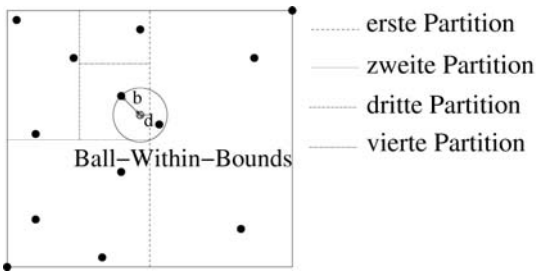


Abbildung 3.8: Aufbau eines kD-Baums, hier $k = 2$. Nachdem der Quader gefunden wurde, der die Punktmenge umschließt, wird die Menge partitioniert. Rekursiv werden weitere Trennlinien eingefügt. Ist bei einer Suche im kD-Baum der Abstand zum nächsten Punkt im Blatt größer als der zur Bucket-Grenze (Ball-Within-Bounds-Test), muss Backtracking durchgeführt werden.

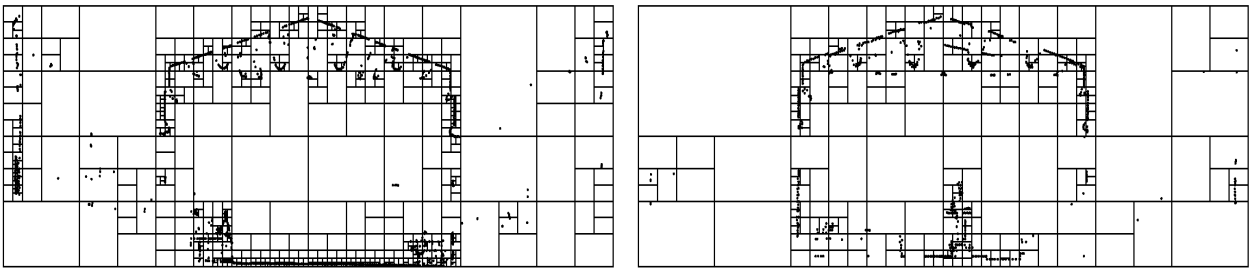


Abbildung 3.9: Visualisierung eines kD-Baums, $k = 3$. Die Abbildung korrespondiert mit dem 3D-Scan aus Abbildung 3.7. Aufgetragen ist die (x, y)-Projektion zweier Schnitte. Links: $z = 100$ cm. Rechts: $z = 550$ cm.

Aufbau eines kD-Baums. In k Dimensionen enthalten die Einträge k Schlüssel. Jeder dieser Schlüssel kann in einem inneren Baumknoten dazu dienen, die Datenmenge zu teilen. Diese so genannten Teiler werden im Original-kD-Baum, wie von Bentley 1975 vorgeschlagen [28], anhand der Tiefe der inneren Knoten gewählt: Die Teilerdimension D für jede Tiefe erhält man durch zyklische Auswahl aus den Dimensionen: $D = L \mod k + 1$, mit der aktuellen Tiefe des Baums L. Für die Wurzel gilt: $L = 0$. Anschließend bestimmt Bentley den Teiler zufällig aus der Punktmenge. Die Teilerdimension und der Teiler definieren eine Hyperebene im k-dimensionalen Raum. Die Daten werden nun gemäß ihrer Lage zu dieser Ebene auf die beiden Söhne des Knotens aufgeteilt und es entsteht die Partitionierung. Der kD-Baum wird solange aufgebaut, bis die Punktmenge in einem Knoten ein bestimmtes Limit b unterschreitet. Die Blätter des Baums enthalten die Punkte. Abbildung 3.8 veranschaulicht den Aufbau eines kD Baums.

Für 3D-Datenwerte ist $k = 3$. Der 3D-Baum enthält Separierungsebenen, die parallel zu der $(x, y, 0)$-, $(0, y, z)$- oder $(x, 0, z)$-Ebene verlaufen. Datenpunkte links neben der Ebene werden im linken Teilbaum gespeichert, während Punkte rechts neben der Ebene im zweiten Teilbaum abgelegt werden. Abbildung 3.9 zeigt zwei Schnitte durch einen kD-Baum. Dabei wurden die 3D-Scandaten aus Abbildung 3.7 verwendet.

Suche in einem kD-Baum. Der Suchalgorithmus für kD-Bäume ist eine rekursive Prozedur. Das Argument der Prozedur ist der aktuelle zu analysierende Knoten, also wird die Wurzel als Startwert übergeben. Die Suche nach dem nächsten Punkt für einen gegebenen Punkt $\mathbf{p}_q$ besteht

aus dem Vergleich von $\mathbf{p}_q$ mit der Teilerdimension und dem Teiler, d.h. im Fall $k = 3$ mit der entsprechenden Separierungsebene. Dadurch bestimmt der Suchalgorithmus, in welchem Nachfolgerknoten weitergesucht werden muss und setzt die Suche dort fort. Dies wird solange wiederholt, bis ein Blatt, das den nächsten Datenpunkt $\mathbf{p}_b$ enthält, erreicht ist. In den Blättern muss der Algorithmus alle Punkte betrachten. Es ist allerdings möglich, dass der nächste Punkt in einem anderen Blatt liegt. Dieser Fall ist zu untersuchen, falls der Abstand zwischen $\mathbf{p}_q$ und der Grenze der Region des Blattes kleiner ist als der Abstand $||\mathbf{p}_q - \mathbf{p}_b||$. Es tritt Backtracking auf, bis alle Blätter analysiert wurden, die im Radius $||\mathbf{p}_q - \mathbf{p}_b||$ liegen. Diese Bedingung ist als Ball-Within-Bounds bekannt [28, 83, 98]. Abbildung 3.8 zeigt den Fall des Backtrackings. Da der Abstand zu dem nächsten Punkt größer ist als der Abstand zum Rand des Blattes, müssen weitere Blätter untersucht werden. Im abgebildeten Beispiel muss bis zur Wurzel des kD-Baums zurückgegangen werden.

Der optimierte kD-Baum

Das Ziel der Optimierung ist, die erwartete Anzahl von Blättern, die untersucht werden muss, zu minimieren. Die veränderbaren Parameter der inneren Knoten sind die Teilerdimension und der Teiler sowie die Anzahl der Daten in den Blättern b.

Die Lösung einer Optimierung hängt in der Regel von der Verteilung der Punkte im kD-Baum und der Anfragepunkte ab. Für gewöhnlich hat man keine Information über die Anfrage und sucht daher nach einem Verfahren, das ausschließlich die Verteilung der gegebenen Punkte berücksichtigt. Für alle möglichen Anfragen funktioniert ein solches Verfahren gut; es wird aber nicht optimal für eine spezielle Anfrage sein [83]. Weiterhin kann man die Teilerdimension und den Teiler nur mit Hilfe des Punktmengenteiles bestimmen, der dem aktuellen Knoten zugeordnet ist. Dies ermöglicht erst den rekursiven Aufbau des kD-Baums und vermeidet eine allgemeine Optimierung des Binärbaums. Eine solche allgemeine Optimierung ist als NP-vollständig bekannt [111].

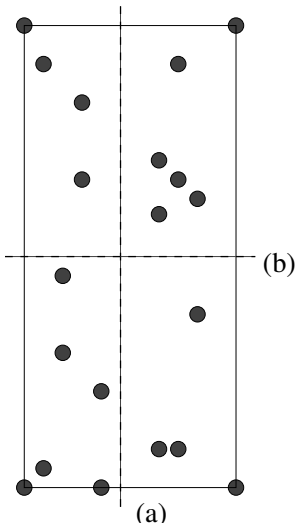


Abbildung 3.10: Partitionierung einer Punktmenge. Die Anwendung des Teilers (b) resultiert in einer kompakten Aufteilung.

Unter diesen beiden Restriktionen können die Teilerdimension und der Teiler an jedem inneren Knoten bestimmt werden. Die Information für den Suchalgorithmus besteht in der Identifizierung der Punkte, die den beiden Nachfolgerknoten zugeordnet sind. Sie ist in einem binären Suchbaum maximal, falls beide Alternativen gleich wahrscheinlich sind. Daher sollte jeder Eintrag die gleiche Wahrscheinlichkeit besitzen, auf einer Seite der Partition zu sein. Dieses Kriterium impliziert, dass die Partition am Median der Punktmenge verlaufen sollte.

Der Suchalgorithmus kann Teilbäume ausschließen, wenn der Ball-Within-Bounds-Test fehlschlägt, d.h. wenn die Distanz zur Partitionierungsebene größer ist als der Radius zum nächsten Punkt (vgl. Abbildung 3.8). Die Wahrscheinlichkeit einer Aufteilung, bei der Teilbäume ausgeschlossen werden können, ist maximal, wenn die Punktmenge parallel zu einem Schlüssel geteilt wird, der den größten Wertebereich aufweist. Abbildung 3.10 zeigt eine Punktmenge und zwei mögliche Partitionierungen: Im Fall der Aufteilung nach (b) tritt seltener Backtracking auf, da der durchschnittliche Punkt-Trennlinie-Abstand größer ist als bei einer Teilung nach (a).

Performanzanalyse. Der optimierte kD-Baum ist ein kD-Baum, bei dem die Teilerdimension anhand der größten Spannweite der Werte bestimmt und der Median als Teiler benutzt wird. Der Rechenaufwand für den Aufbau des optimierten kD-Baums lässt sich wie folgt abschätzen: In jeder Baumtiefe müssen alle Schlüssel untersucht werden. Dies benötigt Berechnungen proportional zu kN. Die Tiefe des Baums ist $\log N$; daher ist die Zeit für den Aufbau in $\mathcal{O}(kN \log N)$, wie sich leicht durch Lösen der Rekursionsformel $T_N = 2T_{N/2} + kN$ bestätigen lässt.

Der Erwartungswert der Suchzeit lässt sich in einem geometrischen Kontext bestimmen. Repräsentiere $\mathbf{p}_i = (p_i(1), p_i(2), \ldots, p_i(k))$ die Schlüsselwerte für den i-ten Datenpunkt in einem k-dimensionalen Raum. Die gesamte Menge von Daten besteht aus solchen Punkten, ebenso wie die Anfrage, die mit $\mathbf{p}_q$ bezeichnet sei. Die Performanz des optimierten kD-Baums hängt von der Anzahl der zu untersuchenden Blätter ab und diese wiederum könnte von der Gesamtzahl der Punkte im Baum N, der Dimensionalität k und der Anzahl der Punkte in den Blättern b abhängen. Friedman et al. beweisen, dass die ersten beiden Parameter keine Rolle spielen [83].

Satz 5. Die Zeit für eine Suche nach den m nächsten Punkten in einem optimierten kD-Baum ist logarithmisch zu der Anzahl der Punkte N im Baum [83].

Beweis: Sei $S_m(\mathbf{p}_q)$ die kleinste Region in dem Koordinatensystem mit Zentrum $\mathbf{p}_q$, die genau die m nächsten Punkte zu $\mathbf{p}_q$ enthält:

$$S_m(\mathbf{p}_q) \equiv \{\mathbf{p} \mid \; ||\mathbf{p} - \mathbf{p}_q|| \leq ||\mathbf{p}_q - \mathbf{p}_m||\},$$

wobei $\mathbf{p}_m$ der m-te nächste Nachbar von $\mathbf{p}_q$ ist [83]. Das k-dimensionale Volumen $v_m(\mathbf{p}_q)$ dieses Balls berechnet sich durch

$$v_m(\mathbf{p}_q) = \int_{S_m(\mathbf{p}_q)} d\mathbf{p}.$$

Die Wahrscheinlichkeit $u_m(\mathbf{p}_q)$ für einen Punkt in dieser Region ist definiert als

$$u_m(\mathbf{p}_q) \equiv \int_{S_m(\mathbf{p}_q)} p(\mathbf{p}) \; d\mathbf{p}, \qquad \text{mit} \quad 0 \leq u_m(\mathbf{p}_q) \leq 1. \tag{3.34}$$

Es kann bewiesen werden, dass die Wahrscheinlichkeitsverteilung von $u_m(\mathbf{p})$ einer Beta-Verteilung $B(m, N)$ entspricht [83, 91], was bedeutet, dass

$$p(u_m) = \frac{N}{(m-1)!(N-m)!}(u_m)^{m-1}(1-u_m)^{N-m}$$

unabhängig von der Wahrscheinlichkeitsdichte der Punkte $p(\mathbf{p})$ und der verwendeten Norm ist. Der Erwartungswert dieser Verteilung berechnet sich als

$$E[u_m] = \int_0^1 u_m \; p(u_m) \; du_m = \frac{m}{N+1}. \tag{3.35}$$

Dieses Ergebnis bedeutet, dass jedes kompakte Volumen, das genau m Punkte enthält, im Durchschnitt eine Wahrscheinlichkeit $m/(N+1)$ besitzt.

Für die weiteren Betrachtungen nimmt man an, dass N groß und $S_m(\mathbf{p}_q)$ klein ist. Daher kann die Wahrscheinlichkeitsverteilung $p(\mathbf{p})$ als ungefähr konstant innerhalb der Region $S_m(\mathbf{p}_q)$ angenommen werden [83]. Die Gleichung (3.34) lässt sich somit durch

$$u_m(\mathbf{p}_q) \approx \bar{p}(\mathbf{p}_q) v_m(\mathbf{p}_q)$$

und die Gleichung (3.35) durch

$$E[v_m(\mathbf{p}_q)] \approx \frac{m}{N+1}\ \frac{1}{\bar{p}(\mathbf{p}_q)} \tag{3.36}$$

approximieren. Hierbei ist $\bar{p}(\mathbf{p}_q)$ die konstante Wahrscheinlichkeitsdichte über der kleinen Region $S_m(\mathbf{p}_q)$, die immer ungleich Null ist. Nun berücksichtigt man die Art und Weise des Aufbaus des optimierten kD-Baums. Da der Algorithmus den Median als Teiler benutzt, kann angenommen werden, dass jedes Blatt annähernd gleich viele Datenpunkte enthält, und zwar b Punkte. Die Wahl der Teilerdimension mit der größten Spannweite stellt sicher, dass die Datenpunkte in den Blättern kompakt sind (vgl. Abbildung 3.10). Die erwartete geometrische Form der Blätter ist hyperkubisch mit der Kantenlänge gleich der k-ten Wurzel des Volumens der Blätter. Des Weiteren sind die Kanten parallel zu denen des Koordinatensystems. Aus Gleichung (3.36) lässt sich das zu erwartende Volumen eines Blattes bestimmen:

$$E[v_b(\mathbf{p}_b)] \approx \frac{b}{N+1}\ \frac{1}{\bar{p}(\mathbf{p}_b)}, \tag{3.37}$$

wobei $\mathbf{p}_b$ den Punkt darstellt, der das Blatt im Koordinatenraum lokalisiert.

Sei nun der kleinste Hyperkubus mit Kanten parallel zu den Koordinatensystemachsen gegeben, der die Region $S_m(\mathbf{p}_q)$ vollständig enthält. Das Volumen jenes Hyperkubus $V_m(\mathbf{p}_q)$ ist proportional zu $v_m(\mathbf{p}_q)$. Die Proportionalitätskonstante $g(k)$ hängt dabei von der Dimensionalität k ab [83]. Es gilt:

$$V_m(\mathbf{p}_q) = g(k)\ v_m(\mathbf{p}_q) \qquad \text{und} \tag{3.38}$$

$$E[V_m(\mathbf{p}_q)] = \frac{m}{N+1}\ \frac{g(k)}{\bar{p}(\mathbf{p}_q)}. \tag{3.39}$$

Um nun die erwartete Anzahl von Blättern zu bestimmen, die der Suchalgorithmus betrachtet, ist es notwendig, die durchschnittliche Anzahl der Blätter $\bar{l}$ zu berechnen, die sich mit der Region $S_m(\mathbf{p}_q)$ überlappen. Diese Anzahl wird nach oben durch die durchschnittliche Anzahl der Blätter $\bar{L}$ begrenzt, die den Hyperkubus $S_m(\mathbf{p}_q)$ umschließen. Letztere durchschnittliche Anzahl bestimmt sich als

$$\bar{L} = \left(\frac{e_m(\mathbf{p}_q)}{e_b(\mathbf{p}_q)+1}\right)^k. \tag{3.40}$$

$e_m(\mathbf{p}_q)$ ist hierbei die Kantenlänge des Hyperkubus, der $S_m(\mathbf{p}_q)$ enthält und $e_b(\mathbf{p}_q)$ die Kantenlänge der benachbarten hyperkubischen Blätter. Die Kantenlänge eines Hyperkubus errechnet sich als die k-te Wurzel des zugehörigen Volumens; bezieht man die Gleichungen (3.37) – (3.40) ein, so ergibt sich

$$\bar{l} \leq \bar{L} = \left(\left(\frac{m}{b} g(k)\right)^{1/k} + 1\right)^k$$

als obere Grenze für die durchschnittliche Anzahl der Blätter, die mit dem konstanten Ball $S_m(\mathbf{p}_q)$ überlappen. Da die Zahl der Daten in jedem Blatt b ist, folgt als obere Schranke $\bar{R}$ für die durchschnittliche Anzahl der Daten, die untersucht werden:

$$\bar{R} \leq b\bar{L} = b\left(\left(\frac{m}{b} g(k)\right)^{1/k} + 1\right)^k. \tag{3.41}$$

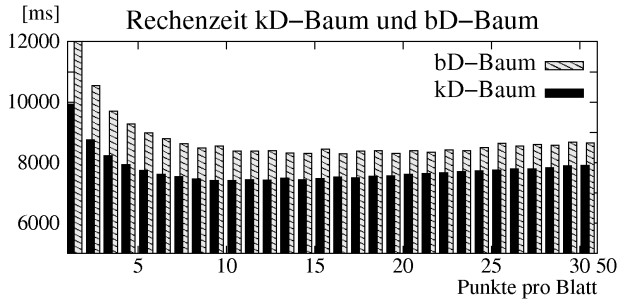


Abbildung 3.11: Zeiten für das Scanmatching in Abhängigkeit von der Anzahl der Datenpunkte im kD- und BD-Baum.

Gleichung (3.41) stellt zwei Ergebnisse zur Verfügung. Erstens: Wird mit Hilfe von b $\bar{R}$ minimiert, erhält man als Resultat $b = 1$. Dies bedeutet, dass die Blätter genau einen Datenpunkt enthalten sollten, um die Suche in einem optimierten kD-Baum möglichst schnell durchzuführen. In der Praxis zeigt sich jedoch, dass etwas größere Werte von b bessere Performanz liefern. Abbildung 3.11 zeigt bei einem kD-Baum ein Minimum bei etwa 10 Datenpunkten pro Blatt. Zweitens: Die erwartete Anzahl von Blättern, die untersucht werden müssen, ist sowohl unabhängig von der Anzahl der Daten N, die der kD-Baum speichert, als auch unabhängig von der Wahrscheinlichkeitsverteilung der Schlüssel $p(\mathbf{p})$.

Diese Ergebnisse lassen sich intuitiv verstehen: Der kD-Baum Algorithmus verfolgt das Ziel, den Überlappungsbereich mit einem Volumen zu minimieren. Also muss die Partitionierung so fein wie möglich sein. Dass kein Zusammenhang zwischen der Anzahl der Blätter, die untersucht werden müssen, und der Datenmenge N bzw. der Verteilung der Schlüssel besteht, ist eine direkte Konsequenz aus dem Aufbaualgorithmus für den optimierten kD-Baum. Die Partitionierung erfolgt nämlich so, dass jedes Blatt die gleiche Wahrscheinlichkeit hat, die Region $S_m(\mathbf{p}_q)$ mit den m nächsten Punkten zu enthalten. Genauer: Die Partitionierung erfolgt so, dass die geometrische Struktur kompakt wird und jedes Blatt gleich viele Daten umfasst. Die Abhängigkeit des Blattvolumens von der Menge der Daten und der Verteilung der Schlüssel ist identisch zu der für die Region $S_m(\mathbf{p}_q)$, die die m nächsten Punkte enthält. Wenn die gesamte Datenmenge vergrößert oder die lokale Schlüsseldichte erhöht wird, sinkt das Volumen der Blattregionen und das Volumen, das die m nächsten Punkte umschließt genau mit der selben Rate, so dass die Anzahl der überlappenden Blätter $\bar{l}$ konstant bleibt.

Da nur konstant viele Blätter untersucht werden müssen, können nächste Punkte in logarithmischer Zeit gefunden werden. Der kD-Baum ist ein balancierter binärer Baum. Die Zeit, die benötigt wird, um von der Wurzel zu den Blättern abzusteigen, verhält sich logarithmisch zu der Anzahl der Knoten, die direkt proportional zu der Datenmenge N ist. Der Aufwand für das Backtracking ist im Mittel proportional zu $\bar{l}$ und unabhängig von N. Daher können die m nächsten Punkte bzw. der nächste Punkt in logarithmischer Zeit gefunden werden. □

Bestimmung des Medians

Die Analyse des optimierten kD-Baums gelingt, weil die jeweiligen Punktmengen am Median gespalten werden. Dadurch ist sichergestellt, dass sich die Punkte immer gleichmäßig auf die Nachfolger-

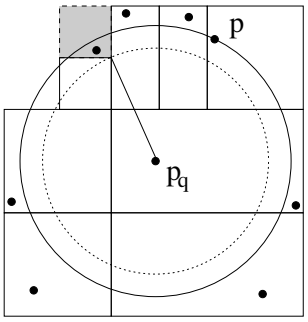


Abbildung 3.12: Der $(1+\varepsilon)$-approximierte nächste Nachbar. Die grau eingefärbte Zelle muss nicht untersucht werden, da der Punkt $\mathbf{p}$ bereits das Approximationskriterium erfüllt.

Knoten verteilen. Die Bestimmung des Medians muss in linearer Zeit geschehen, damit der Baum in $\mathcal{O}(N \log N)$ Zeit aufgebaut werden kann. Überraschenderweise löst folgender einfache Algorithmus das Problem randomisiert. `A[p,...,r]` ist dabei das Eingabefeld; das `i`-kleinste Element wird zurückgegeben [56]:

```
Randomized-Select(A,p,r,i)
if (p == r) return A[p];
q = Randomized-Partition(A,p,r)
k = q - p + 1;
if (i <= k) return Randomized-Select(A,p,q,i)
       else  return Randomized-Select(A,q+1,r,i-k)
```

Die Funktion `Randomized-Partition` teilt das Feld `A` in zwei nicht leere Teilfelder, so dass jedes Element des Feldes `A[p,...,q]` kleiner als jedes Element im Feld `A[q,...,r]` ist. Eine Analyse des Algorithmus zeigt, dass im schlechtesten Fall $\Theta(n^2)$ Zeit benötigt wird, im Mittel jedoch nur $\Theta(n)$ [56].

In [56] findet sich auch ein deterministischer $\Theta(n)$-Algorithmus für die Bestimmung des Medians. Der Algorithmus benutzt ein deterministisches Verfahren, um ein „gutes“ Pivot-Element zur Berechnung der Partitionierung zu finden. Er besitzt jedoch sehr große Konstanten, so dass er in der Praxis Vorteile nur bei sehr großen Punktmengen und ungünstigen Punktverteilungen liefert.

Approximative Bestimmung der nächsten Punkte

Greenspan und Yurick haben 2003 den approximierten kD-Baum (Apx-kD-Baum) für den ICP-Algorithmus benutzt [98]. Der Aufbau des Apx-kD-Baums erfolgt genau wie jener des optimierten kD-Baums. Die Idee der Approximation ist, bei der Suche den nächsten Punkt $\mathbf{p}_a$ durch den nächsten Punkt im Blatt $\mathbf{p}_b$ anzunähern, wo sich der Anfragepunkt $\mathbf{p}$ befindet. Da $\mathbf{p}_b$ nur durch Tiefensuche bestimmt wird, entfallen der Ball-Within-Bounds-Test und das Backtracking [98, 153]. Greenspan und Yurick haben gezeigt, dass der ICP-Algorithmus ein lokales Minimum erreicht, das in der Nähe von jenem liegt, das mit Backtracking erzielt wird.

Dieser Apx-kD-Baum wird in $\mathcal{O}(N \log N)$ Zeit aufgebaut. Eine Suche benötigt $\mathcal{O}(\log N)$ Zeit. Zusätzlich zu der beschriebenen Approximation gibt es die Möglichkeit, den Mittelwert der Punkte in einem Blatt als nächsten Punkt zurückzugeben. Dadurch wird die Approximation zwar gröber, aber auch die Laufzeit nochmals reduziert [153].

Ein Nachteil der beschriebenen Verfahren ist, dass keine Zusicherungen bezüglich der Qualität der Approximation gemacht werden. Arya et al. führen 1993 folgende Notation ein [22]: Gegeben sei ein $\varepsilon > 0$. Dann sei der Punkt $\mathbf{p} \in D$ der $(1+\varepsilon)$-approximierte nächste Nachbar von $\mathbf{p}_q$, falls

$$||\mathbf{p} - \mathbf{q}|| \leq (1+\varepsilon)\ ||\mathbf{p}^* - \mathbf{q}||\,,$$

wobei $\mathbf{p}^*$ den wahren nächsten Nachbarn zu $\mathbf{q}$ darstellt. Mit anderen Worten: $\mathbf{p}$ ist innerhalb eines relativen Fehlers von ε vom wahren nächsten Punkt. Die Suche in einem kD-Baum muss unter Verwendung dieser Notation den jeweils nächsten Punkt $\mathbf{p}$, der bisher gefunden wurde, mitprotokollieren. Die Suche wird erst beendet und $\mathbf{p}$ als nächster Punkt angegeben, wenn der Abstand der noch zu untersuchenden Blätter den Wert $||\mathbf{p}_q - \mathbf{p}||/(1+\varepsilon)$ übersteigt (vgl. Abbildung 3.12). Abbildung 3.15 zeigt das Laufzeitverhalten der Suche nach nächsten Punkten mittels approximativer Bestimmung in Abhängigkeit von ε und der Anzahl der Datenpunkte in den Blättern.

BD-Bäume zur Bestimmung der nächsten Punkte

Arya et al. [23] liefern 1998 einen optimalen Algorithmus für das Auffinden von approximierten nächsten Nachbarn für feste Dimensionen. Der Algorithmus benutzt einen balancierten Box-Dekompositions-Baum (engl.: *balanced box-decomposition tree*) (BD-Baum) als primäre Datenstruktur. Der BD-Baum kombiniert zwei wichtige Eigenschaften geometrischer Datenstrukturen. Erstens nimmt die *Kardinalität* der Punkte, die auf einem Pfad durch den Baum liegen, exponentiell ab. Dies ist auch beim optimierten kD-Baum der Fall. Zweitens wird das *Seitenverhältnis*, das aus der Länge der längsten Seite und der Länge der kürzesten Seite besteht, durch eine Konstante begrenzt. Ein Quadtree ist beispielsweise eine Datenstruktur, die auch diese Eigenschaft aufweist [23]. Diese Zusicherung macht der kD-Baum nicht.

Aufbau eines BD-Baums. Ein BD-Baum wird konstruiert durch die wiederholte Anwendung zweier Operationen: Teilen (engl.: *splits*) und Zusammenziehen (engl.: *shrinks*) [23]. Sie teilen die Datenmenge auf zwei Nachfolgeknoten auf. Es entsteht ein binärer Baum. Die Teilen-Operation benutzt eine Hyperebene, die entlang der Koordinatensystemachsen orientiert ist. Die Datenmenge wird also genau wie beim optimierten kD-Baum geteilt. Bei der Zusammenziehen-Operation entsteht die Partitionierung in zwei disjunkte Teile, indem eine Box bestimmt wird, die von dem Rest subtrahiert wird. Hierbei kommt eine mengentheoretische Differenz zum Einsatz. Abbildung 3.13 zeigt schematisch den Aufbau eines BD-Baums, Abbildung 3.14 gibt einen BD-Baum wieder, dessen korrespondierender kD-Baum in Abbildung 3.9 angegeben ist.

Suche in einem BD-Baum. Die Suche in einem BD-Baum erfolgt analog zu jener im kD-Baum, d.h. es wird zuerst das Blatt bestimmt, in dem sich der Anfragepunkt $\mathbf{p}_q$ befindet. Anschließend wird der Ball-Within-Bounds-Test angewandt und Backtracking ausgeführt, bis alle möglichen Blätter durchsucht wurden. Eine approximative Bestimmung ist ebenfalls möglich.

Für die Suche in einem BD-Baum beweisen S. Ayra et al. die Optimalität, d.h. den folgenden Satz [23]:

Satz 6. Sei S eine Menge mit n Datenpunkten aus $\mathbb{R}^k$. Es gibt eine Konstante $c_{k,\varepsilon} \leq k\lceil 1+6k/\varepsilon\rceil^k$, so dass es in $\mathcal{O}(kn\log n)$ Zeit möglich ist, eine Datenstruktur, nämlich den BD-Baum, zu konstruieren

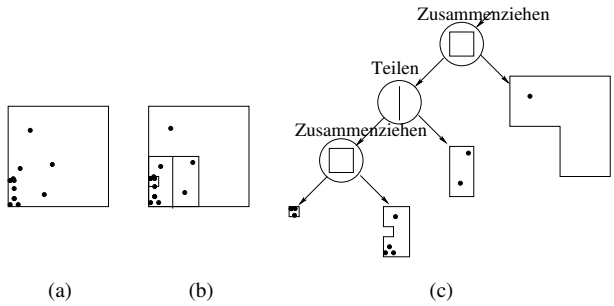


Abbildung 3.13: Schematische Darstellung eines BD-Baums [23]. (a) Gegebene Punktmenge in $\mathbb{R}^2$, (b) Aufteilung der Punktmenge. (c) BD-Baum mit zwei Zusammenziehen- und einer Teilen-Operation.

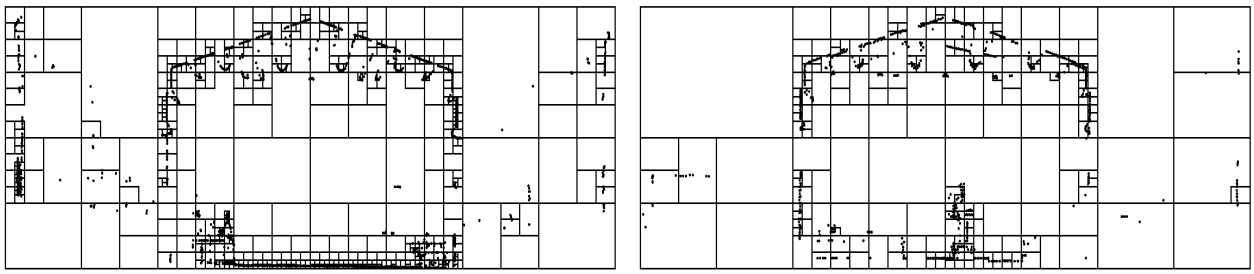


Abbildung 3.14: Visualisierung eines BD-Baums, $k = 3$. Die Abbildung korrespondiert mit dem 3D-Scan aus Abbildung 3.7. Aufgetragen ist die (x, y)-Projektion zweier Schnitte. Links: $z = 100$ cm. Rechts: $z = 550$ cm.

und folgende Bedingung erfüllt ist: Bei gegebenen $\varepsilon > 0$ und $\mathbf{q} \in \mathbb{R}^k$ kann ein $(1+\varepsilon)$-approximierter nächster Nachbar von $\mathbf{p}_q$ in Zeit $\mathcal{O}(c_{k,\varepsilon} \log n)$ bestimmt werden.

Beweis: Siehe [23]. □

Die Hoffnung bei einer Suche im BD-Baum ist, dass weniger Operationen nötig sind, um den nächsten Punkt zu bestimmen, als beim kD-Baum. Durch das Zusammenziehen sollte es möglich sein, sich schnell auf Bereiche mit hohen Punktdichten zu konzentrieren. Die Abbildungen 3.11 und 3.15 zeigen die Zeiten für das Scanmatching mit einem kD-Baum, einem BD-Baum und den approximierenden Varianten. Die approximierenden Versionen sind wesentlich schneller als die präzisen. Des Weiteren ist zu erkennen, dass mit einigen Ausnahmen der kD-Baum performanter ist als der BD-Baum. Abbildung 3.16 zeigt für einige ε-Approximationswerte und für verschiedene Datenanzahlen in den Blättern b diejenigen Startposen, die trotz der Approximation zu einem korrekten Scanmatching führen.

3.2 Matching mehrerer 3D-Scans

Das Digitalisieren von Umgebungen erfordert die Aufnahme und das Registrieren mehrerer 3D-Scans. Nach dem Registrieren muss die Szene konsistent sein. In der Robotik spricht man in diesem Zusammenhang vom Problem der gleichzeitigen Lokalisation und Kartierung (engl.: *simultaneous localization and mapping (SLAM)*). Die im Folgenden vorgestellten Algorithmen, die auf dem bereits

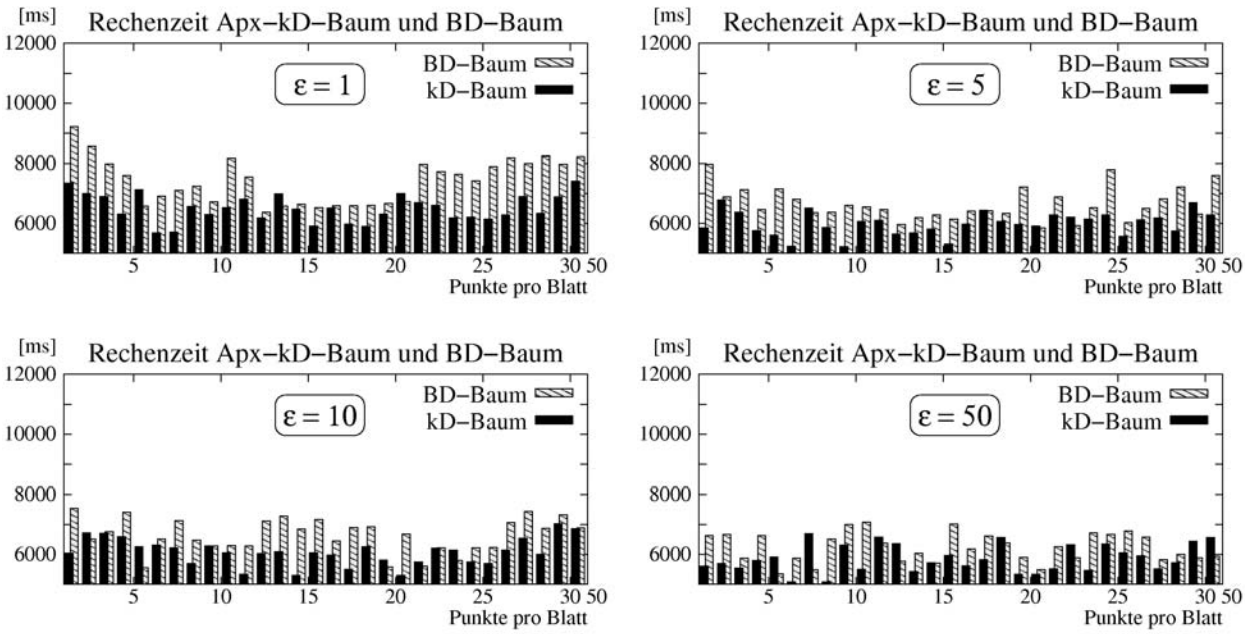


Abbildung 3.15: Zeiten für das Scanmatching in Abhängigkeit von der Anzahl der Datenpunkte im approximierenden kD- und BD-Baum. Ergebnisse für $\varepsilon = 1$, $\varepsilon = 5$, $\varepsilon = 10$, $\varepsilon = 50$.

beschriebenen Scanmatching basieren, erlauben das Erstellen einer 3D-Karte, wobei die Roboterpose $\mathbf{P}_{\text{Roboter}}$ in allen 6 Freiheitsgraden $(x, y, z, \theta_x, \theta_y, \theta_z)$ korrigiert wird. Der hieraus resultierende Ansatz wird 6D-SLAM genannt [153, 194, 195].

Beim Registrieren von n Mengen mit 3D-Messpunkten zu einem konsistenten Modell handelt es sich um ein aktuelles Forschungsgebiet, in dem bereits einige Lösungsansätze präsentiert wurden [27, 36, 59, 66, 167, 188]. Eine sehr einfache Methode für das Registrieren mehrerer 3D-Scans ist das *paarweise Matching*. Auf zwei nacheinander aufgenommene 3D-Scans wird jeweils der iterative Algorithmus der nächsten Punkte (ICP) angewendet. Der zweite Scan benutzt zum Registrieren die Daten des ersten Scans, der dritte Scan die Daten des zweiten Scans, u.s.w. Bei einem solchen paarweisen Matching kumulieren allerdings Fehler. Da das Scanmatching vorangegangener Scans niemals perfekt sein kann, wird dieser Fehler an die nachfolgenden Scans weitergegeben. Hinzu kommen außerdem neue Registrationsfehler.

Auch das *inkrementelle Matching* läuft nicht ohne Fehler ab [50]. Hierbei sind zuerst zwei 3D-Scans zu registrieren. Diese werden anschließend zu einer einzigen Datenmenge (Metascan) zusammengefasst [50]. Der nun folgende Scan wird mit diesem Metascan registriert, wodurch ein neuer Metascan entsteht. Auch bei dieser Matching Methode summieren sich die Fehler.

Die beiden Verfahren haben einen entscheidenden Nachteil: Wenn zusätzliche Scans hinzugefügt werden, besteht die Möglichkeit, dass diese neuen Scans Informationen mitbringen, die das bisherige Matching verbessern könnten [167]. Der folgende Algorithmus nutzt sämtliche Informationen aus, die zur Verfügung stehen, um global konsistente 3D-Karten zu erzeugen. Dabei werden geschlossene Kreise erkannt und der globale Fehler minimiert (engl.: *global relaxation*) [167, 195].

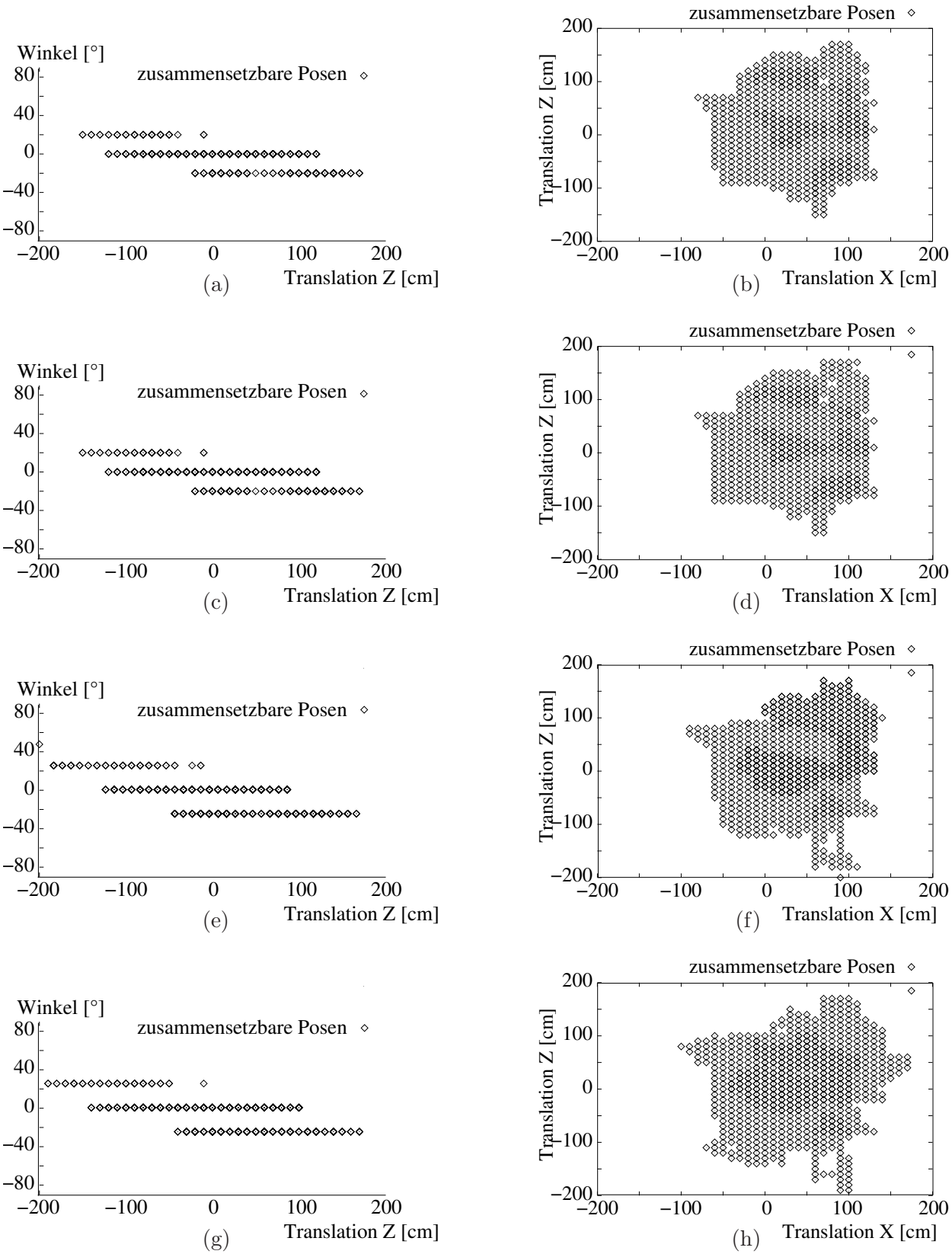


Abbildung 3.16: Startposen, die trotz ε-approximierter Bestimmung der nächsten Punkte zu einem korrekten Scanmatching führen, in Zweitafelprojektion (b Anzahl Datenpunkte pro Blatt). Posen, deren Fehler $e(x, z, \theta)$ nach dem Matching kleiner 100 ist, sind eingezeichnet. (a) und (b): $\varepsilon = 1$ und $b = 10$. (c) und (d): $\varepsilon = 1$ und $b = 20$. (e) und (f): $\varepsilon = 10$ und $b = 10$. (g) und (h): $\varepsilon = 50$ und $b = 5$.

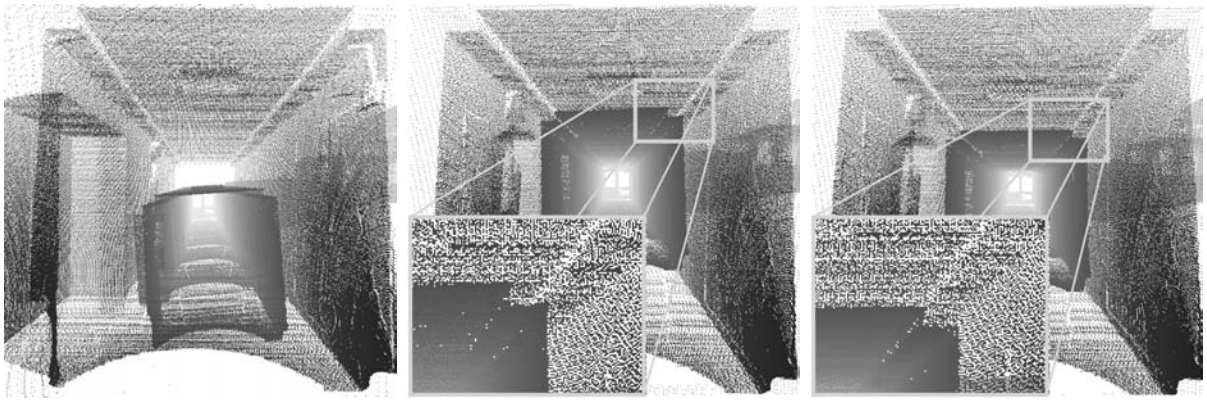


Abbildung 3.17: Links: Detektion eines geschlossenen Kreises und initiale Posen von 3D-Scans beim Schließen eines Kreises. Mitte: Die gleichmäßige Fehlerverteilung über alle Scans im Kreis ergibt einen kleineren Fehler. Rechts: Die globale Korrektur beseitigt die letzten Fehler. Ein konsistentes Modell entsteht.

1. In einem Vorverarbeitungsschritt benutze paarweises Matching, um alle 3D-Scans in etwa in die richtige Position zu bringen. Der zuerst aufgenommene 3D-Scan ist der Masterscan definiert das Koordinatensystem.
2. Detektiere geschlossene Kreise, registriere die beiden 3D-Scans, die den Kreis schließen, und verteile die Transformation $(\mathbf{R}, \mathbf{t})$ gleichmäßig über alle 3D-Scans des Kreises.
3. Initialisiere eine Liste mit dem zu registrierenden Scan.
4. Solange die Liste nicht leer ist, führe folgende Schritte aus:
   (a) Der aktuelle Scan ist das vorderste Element der Liste und wird aus der Liste entfernt.
   (b) Falls der aktuelle Scan nicht der Masterscan ist, bestimme die Nachbarn des Scans und registriere ihn anhand der Menge der Nachbar-Scans.
   (c) Falls der aktuelle Scan seine Lage ändert, füge die Nachbar-Scans am Ende der Liste hinzu.

Bemerkung: Ein Kreis gilt als detektiert, wenn es einen genügend großen Überlappungsbereich von 3D-Scans gibt und diese nicht direkt aufeinanderfolgen (hier: >5 3D-Scans). Der Überlappungsbereich drückt sich durch die Anzahl der möglichen nächsten Punktpaare unter Verwendung von $d_{\max}$ aus. Des Weiteren sind 3D-Scans benachbart, wenn ihr Überlappungsbereich groß ist (hier: > 500 reduzierte Punkte). Die Schritte 1 und 2 des obigen Algorithmus dienen hauptsächlich dazu, die globale Fehlerkorrektur zu beschleunigen.

Abbildung 3.17 zeigt einen Ausschnitt aus einem „closed loop“ Experiment. Dazu wurden 32 3D-Scans mit je 350000 Messpunkten auf dem Gelände des Fraunhofer Instituts AiS aufgenommen. Die anfängliche Lage der 3D-Scans beim Schließen des Kreises (links), ihre Lage, nachdem der Fehler gleichmäßig verteilt wurde (Mitte), sowie jene nach der globalen Fehlerminimierung (rechts) sind dargestellt.

Abbildung 3.18: Resultierendes 3D-Modell

Für das oben erwähnte Experiment war es notwendig, alle Freiheitsgrade in der Roboterpose zu berücksichtigen. Die obige Abbildung zeigt das komplette 3D-Modell in einer Ansicht von oben. Außerdem sind zwei detailliertere Ausschnitte angegeben. Sie zeigen eine Rampe – hier musste der Roboter einen Höhenunterschied von 120 cm überwinden – und einen Ausschnitt, der außerhalb des Gebäudes C2 und des Robotikpavillons liegt. Der Roboter befand sich auf unebenem Untergrund und scannte komplizierte Objekte, wie z.B. einen Baum. Ein Video, das das gesamte 3D-Modell als virtuellen Durchflug enthält, befindet sich unter `http://www.informatik.uni-osnabrueck.de/nuechter/vid`

Verteilung eines Fehlers über mehrere 3D-Scans. Der Algorithmus zum Matchen mehrerer 3D-Scans verteilt beim Schließen eines Kreises die Transformation, die aus der Rotation $\mathbf{R}$ und der Translation $\mathbf{t}$ besteht, über alle n 3D-Scans des Kreises. Die Transformation registriert den n-ten mit dem ersten 3D-Scan des Kreises. Die $(n-1)$ verbleibenden Roboterposen $\mathbf{P}_{\text{Roboter}_i}$, die den Kreis bilden, werden durch folgende Schritte aktualisiert:

1. Der jeweilige Anteil an der Translation wird berechnet:

$$\mathbf{t}_i = \frac{i}{n-1}\mathbf{t}. \tag{3.42}$$

2. Eine Interpolation zwischen zwei Rotationsmatrizen ist notwendig, um den anteiligen Wert für die Drehung zu berechnen. Sie ist möglich, wenn die Drehung durch eine Rotationsachse **a** und einen Winkel θ ausgedrückt wird (vgl. Seite 18). In diesem Fall lässt sich der Winkel in die gewünschten $(n-1)$-Schritte aufteilen. Für die Umrechnung einer Rotationsmatix in eine Rotationsachse **a** und den Winkel θ muss der Umweg über die Quaternionen genommen werden [60, 104, 159]. Analog zu Formel (3.18) lässt sich eine Matrix $\mathbf{R}$ in ein Quaternion $\dot{\mathbf{q}}$ überführen. Dazu wird folgende inverse Rechenvorschrift verwendet (vgl. auch [71, 180]):

$$\dot{\mathbf{q}} = \begin{pmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\sqrt{\text{Trace}\,(\mathbf{R})} \\ \frac{1}{2}\frac{r_{3,3}-r_{3,2}}{\sqrt{\text{Trace}(\mathbf{R})}} \\ \frac{1}{2}\frac{r_{2,1}-r_{2,3}}{\sqrt{\text{Trace}(\mathbf{R})}} \\ \frac{1}{2}\frac{r_{1,2}-r_{1,1}}{\sqrt{\text{Trace}(\mathbf{R})}} \end{pmatrix} \quad \text{, mit den Elementen } r_{i,j} \text{ von } \mathbf{R}. \tag{3.43}$$

Ist die Spur $\text{Trace}\,(\mathbf{R})$ jedoch Null, gilt die obige Rechnung (3.43) nicht. Es muss dann wie folgt gerechnet werden: Falls $r_{1,1} > r_{2,2}$ und $r_{1,1} > r_{3,3}$ ist, gilt

$$\dot{\mathbf{q}} = \begin{pmatrix} \frac{1}{2}\frac{r_{2,3}-r_{3,2}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}} \\ \frac{1}{2}\frac{r_{1,2}+r_{2,1}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\frac{r_{3,1}+r_{1,3}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \end{pmatrix},$$

falls $r_{2,2} > r_{3,3}$ ist, gilt

$$\dot{\mathbf{q}} = \begin{pmatrix} \frac{1}{2}\frac{r_{3,1}-r_{1,3}}{\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\frac{r_{1,2}+r_{2,1}}{\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}} \\ \frac{1}{2}\frac{r_{2,3}+r_{3,2}}{\sqrt{1-r_{1,1}+r_{2,2}-r_{3,3}}} \end{pmatrix},$$

ansonsten ergibt sich das Quaternion $\dot{\mathbf{q}}$ als

$$\dot{\mathbf{q}} = \begin{pmatrix} \frac{1}{2}\frac{r_{1,2}-r_{2,1}}{\sqrt{1-r_{1,1}-r_{2,2}+r_{3,3}}} \\ \frac{1}{2}\frac{r_{3,1}+r_{1,3}}{\sqrt{1+r_{1,1}-r_{2,2}-r_{3,3}}} \\ \frac{1}{2}\frac{r_{2,3}+r_{3,2}}{\sqrt{1-r_{1,1}-r_{2,2}+r_{3,3}}} \\ \frac{1}{2}\sqrt{1-r_{1,1}-r_{2,2}+r_{3,3}} \end{pmatrix}.$$

Nachdem das der Rotation entsprechende Quaternion bestimmt ist, wird es analog zu Formel (3.15) in eine Rotationsachse $\mathbf{a}$ und einen Drehwinkel θ entlang dieser Achse umgewandelt. Im Anschluss an das Normalisieren des Quaternions $\dot{\mathbf{q}}$ berechnet sich

$$\mathbf{a} = \begin{pmatrix} \frac{q_x}{\sqrt{1-q_0^2}} \\ \frac{q_y}{\sqrt{1-q_0^2}} \\ \frac{q_z}{\sqrt{1-q_0^2}} \end{pmatrix} \qquad \text{und} \qquad \theta = 2 \arccos q_o.$$

Der Winkel θ lässt sich nun auf die $(n-1)$ 3D-Scans des geschlossenen Kreises aufteilen und die gewünschten interpolierenden Rotationen können bestimmt werden:

$$\mathbf{R}_i = \begin{pmatrix} \cos\frac{i\theta}{n-1} + a_x^2(1-\cos\frac{i\theta}{n-1}) & a_z \sin\frac{i\theta}{n-1} + a_x a_y(1-\cos\frac{i\theta}{n-1}) \\ -a_z \sin\frac{i}{n-1}\theta + a_x a_y(1-\cos\frac{i}{n-1}\theta) & \cos\frac{i\theta}{n-1} + a_y^2(1-\cos\frac{i\theta}{n-1}) \\ a_y \sin\frac{i\theta}{n-1} + a_x a_z(1-\cos\frac{i\theta}{n-1}) & -a_x \sin\frac{i\theta}{n-1} + a_y a_z(1-\cos\frac{i\theta}{n-1}) \end{pmatrix}$$

$$\begin{pmatrix} -a_y \sin\frac{i\theta}{n-1} + a_x a_z(1-\cos\frac{i\theta}{n-1}) \\ -a_x \sin\frac{i\theta}{n-1} + a_y a_z(1-\cos\frac{i\theta}{n-1}) \\ \cos\frac{i\theta}{n-1} + a_z^2(1-\cos\frac{i\theta}{n-1}) \end{pmatrix}. \tag{3.44}$$

Mit den Matrizen $\mathbf{R}_i$ und den Vektoren $\mathbf{t}_i$ lassen sich die restlichen 3D-Scans und deren Roboterposen aktualisieren.

Fehlerfortschreibung und Korrektur der Roboterpose. Um 6D-SLAM auf dem Roboter Kurt3D einzusetzen, müssen initiale Poseschätzungen $\mathbf{P}$ als Startwerte für das Scanmatching erzeugt werden. Kurt3D kann sich während der Fahrt nur mit Winkelencoder (Odometrie) und HA-YAI lokalisieren (vgl. Kapitel 5.1.3). Diese Online-Lokalisierung auf ebenen Untergründen berücksichtigt nur 2D-Positionen (x, z) und einen Winkel θ_y. Dennoch lassen sich daraus Schätzungen für 6D-SLAM ableiten. Der erste aufgenommene 3D-Scan definiert das Koordinatensystem, befindet sich also im Nullpunkt. Für alle weiteren 3D-Scans erfolgt zuerst eine Berechnung der Poseänderung $\Delta\mathbf{P}$:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \\ \theta_{x,n+1} \\ \theta_{y,n+1} \\ \theta_{z,n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \\ z_n \\ \theta_{x,n} \\ \theta_{y,n} \\ \theta_{z,n} \end{pmatrix} + \left(\begin{array}{c|ccc} \mathbf{R}(\theta_{x,n}, \theta_{y,n}, \theta_{z,n}) & \multicolumn{3}{c}{\mathbf{0}} \\ \hline & 1 & 0 & 0 \\ \mathbf{0} & 0 & 1 & 0 \\ & 0 & 0 & 1 \end{array}\right) \cdot \underbrace{\begin{pmatrix} \Delta x_{n+1} \\ \Delta y_{n+1} \\ \Delta z_{n+1} \\ \Delta\theta_{x,n+1} \\ \Delta\theta_{y,n+1} \\ \Delta\theta_{z,n+1} \end{pmatrix}}_{\Delta\mathbf{P}}. \tag{3.45}$$

Der Übergang $\mathbf{P}_n \rightarrow \mathbf{P}_{n+1}$ erfordert folglich das Invertieren einer Rotationsmatrix, was dem Transponieren entspricht. Ist $\Delta\mathbf{P}$ berechnet, wird diese Poseänderung mit der 6D-Pose des vorangegangenen 3D-Scans multipliziert:

$$\mathbf{P}_{n+1} = \Delta\mathbf{P} \cdot \mathbf{P}_n. \tag{3.46}$$

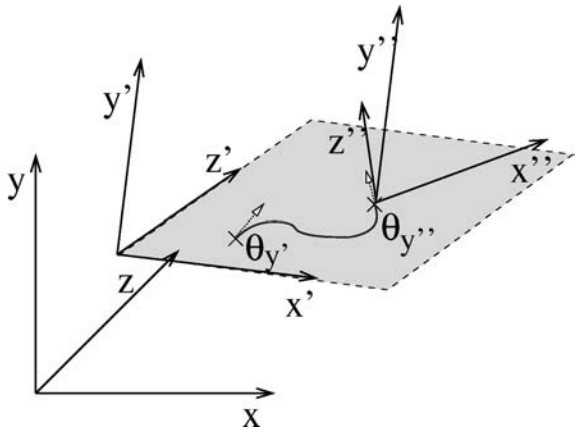


Abbildung 3.19: Korrektur der Roboterpose. Das globale Koordinatensysten sei durch (x, y, z) definiert. Ist ein 3D-Scan mit dem lokalen Koordinatensystem (x', y', z') bereits registriert, findet die Fortschreibung der Roboterbewegung, gemessen durch Odometrie und/oder HAYAI, in der (x', z')-Ebene statt. Diese so gewonnene 6D-Pose dient als Startwert für das Registrieren des 3D-Scans mit den lokalen Koordinaten (x'', y'', z'').

Benutzt man Formel (3.46) für die initiale, vor dem Scanmatching stattfindende Schätzung der Roboterpose, sind alle 6 Freiheitsgrade berücksichtigt (vgl. Abbildung 3.19). Zwar sind die Einträge für θ_{xn} und θ_{zn} wegen der planaren Lokalisierung mittels Odometrie und/oder HAYAI in Gleichung (3.45) Null, indem aber die letzte 6D-Pose herangezogen wird, die bereits völlig frei gedreht sein kann, ist eine Lokalisierung in 6D möglich.

Neigungssensoren. Neigungssensoren liefern zusätzliche Informationen, die für die Startposeschätzung nützlich sind. Sie messen die Orientierung des Gravitationsvektors (Vektor in Richtung Erdmittelpunkt) relativ zum Roboter. In der einfachsten Form kommen sie als Quecksilberschalter vor. Ausgefeilte Varianten, wie der auf Kurt3D eingesetzte Typ ADXL202, messen die Neigung kontinuierlich. Dabei stehen zwei Anordnungsmöglichkeiten zur Verfügung, so dass Roll- und Nickwinkel gemessen werden können. Bedingt durch das Messprinzip liefert der Neigungssensor ADXL202 nur brauchbare Werte, wenn der Roboter steht, dabei die Werte aufintegriert und mittelt. Da Kurt3D zur Aufnahme von 3D-Scans anhalten muss, werden an diesen diskreten Zeitpunkten die Neigungssensoren abgefragt. Werden Neigungssensoren benutzt, sind in Gleichung (3.45) alle Einträge besetzt. Im Gegensatz zu θ_y werden θ_x und θ_z nicht kontinuierlich fortgeschrieben, sondern nur an den Scanposen bestimmt. Ein systematischer Fehler entsteht, den das Scanmatching beheben muss.

3.3 Ergebnisse

Der Einsatz im Freien stellt besondere Anforderungen an Lokalisationsalgorithmen, da die Umgebung sehr unstrukturiert ist. Im Folgenden wird ein Experiment beschrieben, das die Allgemeingültigkeit und die Einsatzmöglichkeiten der Algorithmen für die Lösung des simultanen Lokalisations- und Kartierungsproblems in 6 Dimensionen demonstriert. Bereits die Kartierungsergebnisse, die zu den Abbildungen 3.17 und 3.18 führten, benötigten alle 6 Freiheitsgrade. Für eine weitere

Evaluation wurde ein noch schwierigeres Szenario gewählt: Kurt3D digitalisiert seine Umgebung auf einer Weglänge von zirka 285 Metern. Nach durchschnittlich 3.75 Metern stoppt der teleoperierte Roboter, um einen 3D-Scan aufzunehmen. Der Weg führt dabei in hügeligen Gelände über schlechten Asphalt, über eine Wiese, über Waldboden und über Verbundsteinpflaster. Abbildung 3.20 zeigt den Roboter auf dieser Messfahrt.

3.3.1 Heuristik zur Schätzung der Initialpose

Während des Experiments wurde zunächst versucht, die initiale Poseschätzung **P** für Formel (3.46) mit Hilfe des HAYAI Posetrackingverfahrens durchzuführen, das auf dem Matching von 2D-Scans basiert, die während der Fahrt aufgenommenen werden (siehe Kapitel 5.1.3). Dies gelang jedoch

Abbildung 3.20: Kurt3D auf Geländefahrt. Oben: Wiese. Unten: Waldboden und Verbundsteinpflaster.

.

nicht, da der Untergrund auf der Wiese und dem Waldboden zu uneben ist. Der Scanner wackelt beim Fahren so stark, dass kein Matching mehr möglich ist. Daher steht nur die Odometrie für die initiale Schätzung der Pose zur Verfügung.

Abbildung 3.25 rechts zeigt das Registrieren der 76 3D-Scans unter Verwendung der Odometrie. Die Poseschätzung über Odometrie ist außerdem als Startwert für das Scanmatching unzureichend, da die Bodenhaftung des Roboters zu gering ist. Folgende Heuristik wird dem globalen Registrierungsalgorithmus (vgl. Seite 41) vorweggeschaltet, um zu einem hinreichend guten Startwert zu gelangen:

0. (a) Erzeuge einen Octalbaum $\mathfrak{O}_D$ für den zu registrierenden 3D-Scan D.
   (b) Erzeuge einen Octalbaum $\mathfrak{O}_M$ für den zuletzt aufgenommenen 3D-Scan M.
   (c) Für Suchtiefen im Octalbaum $t \in [t_{\text{Start}}, \ldots, t_{\text{End}}]$ bestimme die beste Transformation $\Delta\mathbf{P}$ wie folgt:
   - Berechne maximal sinnvolle Verschiebungen in x-, y- und z-Richtung sowie Drehungen θ_y in Abhängigkeit der Suchtiefe t und der bisherigen Transformation der Pose $\Delta\mathbf{P}_{\text{best}}$.
   - Für diskrete 6-Tupel $\Delta\mathbf{P}_i \in [-\Delta\mathbf{P}, \Delta\mathbf{P}]$ mit dem Wertebereich $\mathbf{P}_\Delta = (x, y, z, 0, \theta_y, 0)$ verschiebe den Octalbaum $\mathfrak{O}_D$ um $\Delta\mathbf{P}_i$ und evaluiere das Matching der Octalbäume $\mathfrak{O}_M$ und $\mathfrak{O}_D$. Die Qualität des Matchings bestimmt sich durch die Anzahl überlappender Quader. Anschließend aktualisiere die beste Verschiebung $\Delta\mathbf{P}_{\text{best}} = (x_{\text{best}}, y_{\text{best}}, z_{\text{best}}, 0, \theta_{y,\text{best}}, 0)$.

Bemerkung: Im obigen Algorithmus ist für $t = t_{\text{Start}}$ die Pose $\Delta\mathbf{P}_{\text{best}}$ die Schätzung über Odometrie, d.h. $\Delta\mathbf{P}_{\text{best}} = (x_{\text{odo}}, 0, z_{\text{odo}}, 0, \theta_{y,\text{odo}}, 0)$. Des Weiteren hängt die Diskretisierung von der Suchtiefe im Octalbaum ab. Für geringe Baumtiefen werden nur wenige Diskretisierungsschritte benötigt. Die verwendeten Suchtiefen im Octalbaum entsprechen Quadergrößen von $(20\text{ cm})^3$, $(40\text{ cm})^3$ und $(80\text{ cm})^3$.

Obige Heuristik bildet Octalbäume aufeinander ab. Bei der Datenstruktur Octalbaum besitzt jeder innere Knoten des Baums acht Nachfolgerknoten. Die Blätter haben keine Nachfolger, enthalten aber die im Baum gespeicherten Daten. Alle Datenpunkte können von einem Quader umschlossen werden, der die Wurzel des Octalbaumes darstellt und entlang der Achsen des Koordinatensystems ausgerichtet ist. Durch das Einfügen von 3 Ebenen (horizontal, vertikal und in der Flucht) wird der Quader in kleinere, gleich große Quader zerteilt. Führt man dies rekursiv aus und schneidet dabei leere Quader ab, indem man keine Nachfolgerknoten bestimmt, ergibt sich eine Oberflächenrepräsentation des 3D-Scans. Abbildung 3.21 zeigt den beschriebenen Prozess.

Das Matchen von Octalbäumen hat drei Vorteile gegenüber jenem, das auf Punkten basiert: Erstens ist durch die Baumstruktur eine inhärente Grob-Fein-Granularität im Algorithmus gegeben. Durch die schrittweise Erhöhung der Baumtiefe t ist dies moduliert. Zweitens werden unterschiedlich hohe Punktdichten auf den gescannten Oberflächen ausgeglichen, deren Ursache in der kugelförmigen Abstrahlung der Laserstrahlen aus dem 3D-Scanner liegt. Drittens ermöglichen Octalbäume eine schnelle Suche.

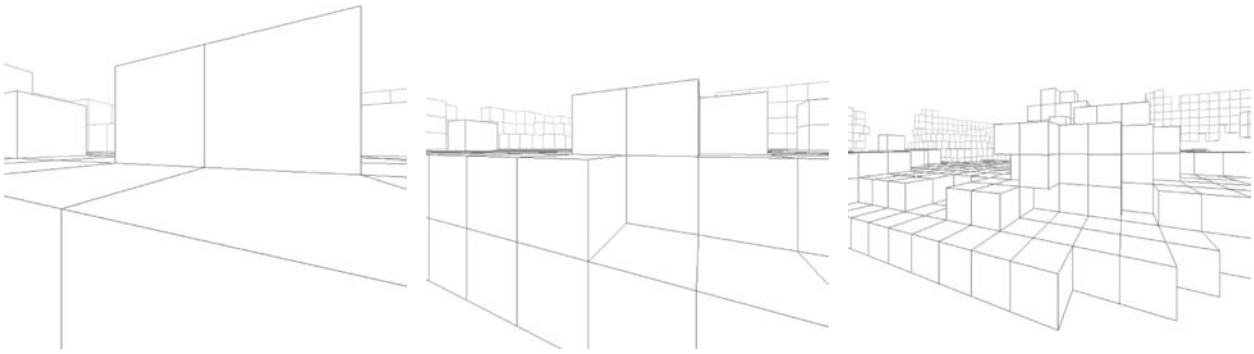


Abbildung 3.21: Aufbau eines Octalbaumes. Der Octalbaum entsteht durch das Aufteilen eines Quaders in 8 Nachfolger und durch das Abschneiden leerer Quader.

Performanz des Algorithmus zur initialen Poseschätzung. Der Octalbaum lässt sich wie beschrieben in $\mathcal{O}(n)$ aufbauen. In jeder Rekursionsstufe müssen alle Punkte betrachtet werden. Für einen gegebenen Punkt lassen sich Quader in logarithmischer Zeit finden. Die Suche ist allerdings nicht ganz so performant wie im kD-Baum. Die Evaluation des Matchings benötigt in etwa die gleiche Zeit wie eine Iteration des ICP Algorithmus. Der gesamte Zeitaufwand der Heuristik ist doppelt so groß wie die anschließende Registrierung der 3D-Scans.

Abbildung 3.22 zeigt zwei Octalbäume und eine Punktwolke, die aus zwei 3D-Scans besteht. Die Grafiken basieren auf dem 18. und 19. 3D-Scan des durchgeführten Experiments. Das Registrieren der beiden 3D-Scans gelingt erst nach dem Verschieben der Octalbäume.

3.3.2 Anwendung der Scanmatchingalgorithmen

Abbildungen 3.23 und 3.25 geben das Ergebnis der Registrierungsalgorithmen als 3D-Punktwolken in einer Ansicht von oben wieder. Für eine Größenvorstellung des gescannten Gebiets ist an der Startposition des Roboters ein 400 m^2 großes Raster eingezeichnet. Die Positionen, an denen 3D-Scans aufgenommen wurden, sind blau markiert. Der linke Teil von Abbildung 3.23 stellt die ersten 62 3D-Scans vor dem Detektieren des geschlossenen Kreises dar. In der rechten Abbildung wurde der Kreis geschlossen, der Fehler verteilt und global optimiert. Abbildung 3.24 zeigt einen vergrößerten Ausschnitt von der Stelle, an der sich der Kreis schließt. Dieser Punkt ist nach zirka 240 Metern

Abbildung 3.22: Heuristik zur Bestimmung einer initialen Poseschätzung. Links und rechts: Octalbäume zweier zu registrierender 3D-Scans. Mitte: Zwei 3D-Scans. Nur mit der Heuristik zur initialen Poseschätzung gelingt das korrekte Registrieren.

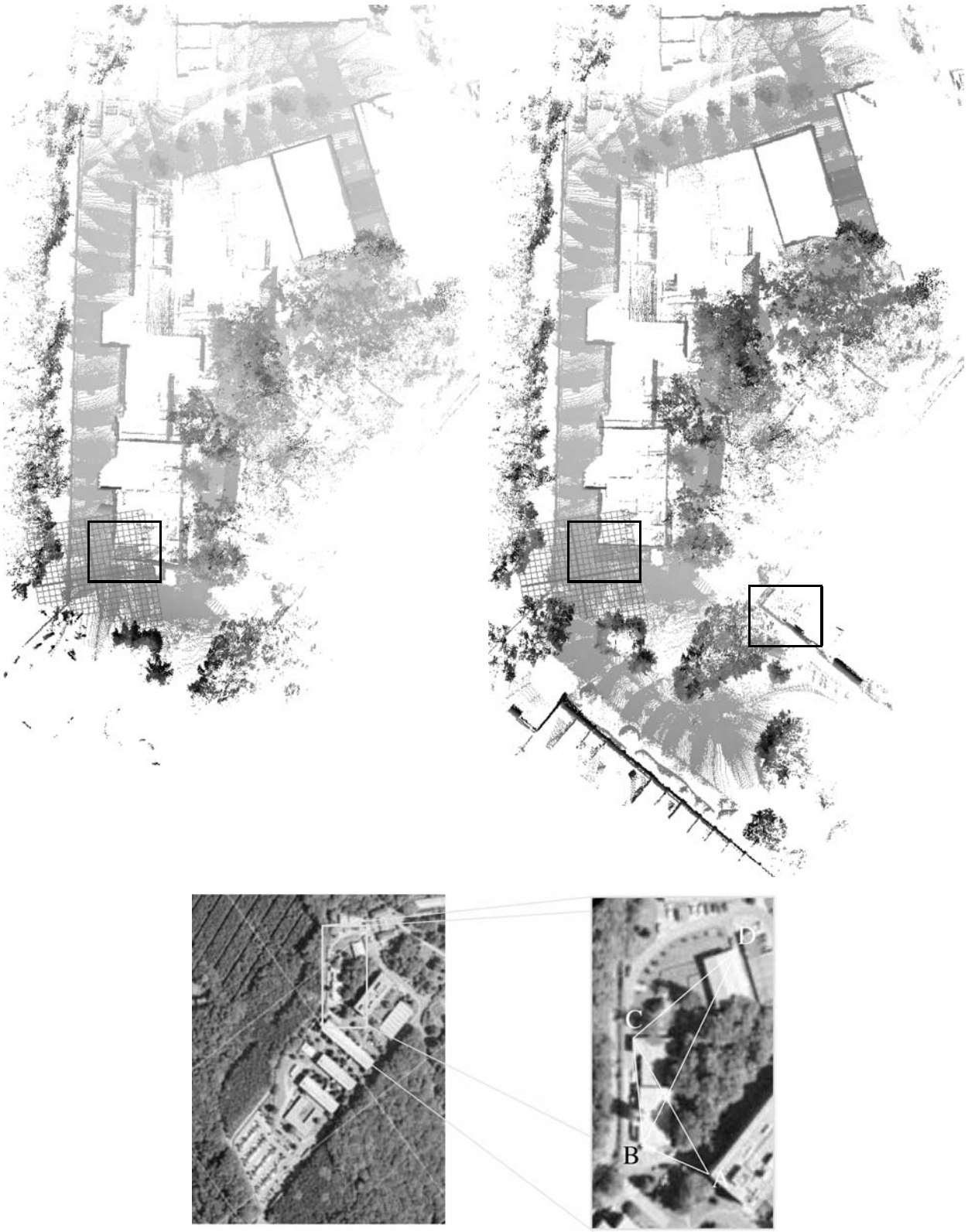


Abbildung 3.23: Oben: 3D-Modell einer Fahrt im Freien (Ansicht von oben). Oben links: Modell vor dem Schließen des Kreises. Oben rechts: Endgültiges Modell mit globaler Fehlerminimierung. Unten: Luftbilder der Szene. Die Gebäude auf dem Campus in Birlinghoven und ein Ausschnitt, der mit dem 3D-Modell korrespondiert.

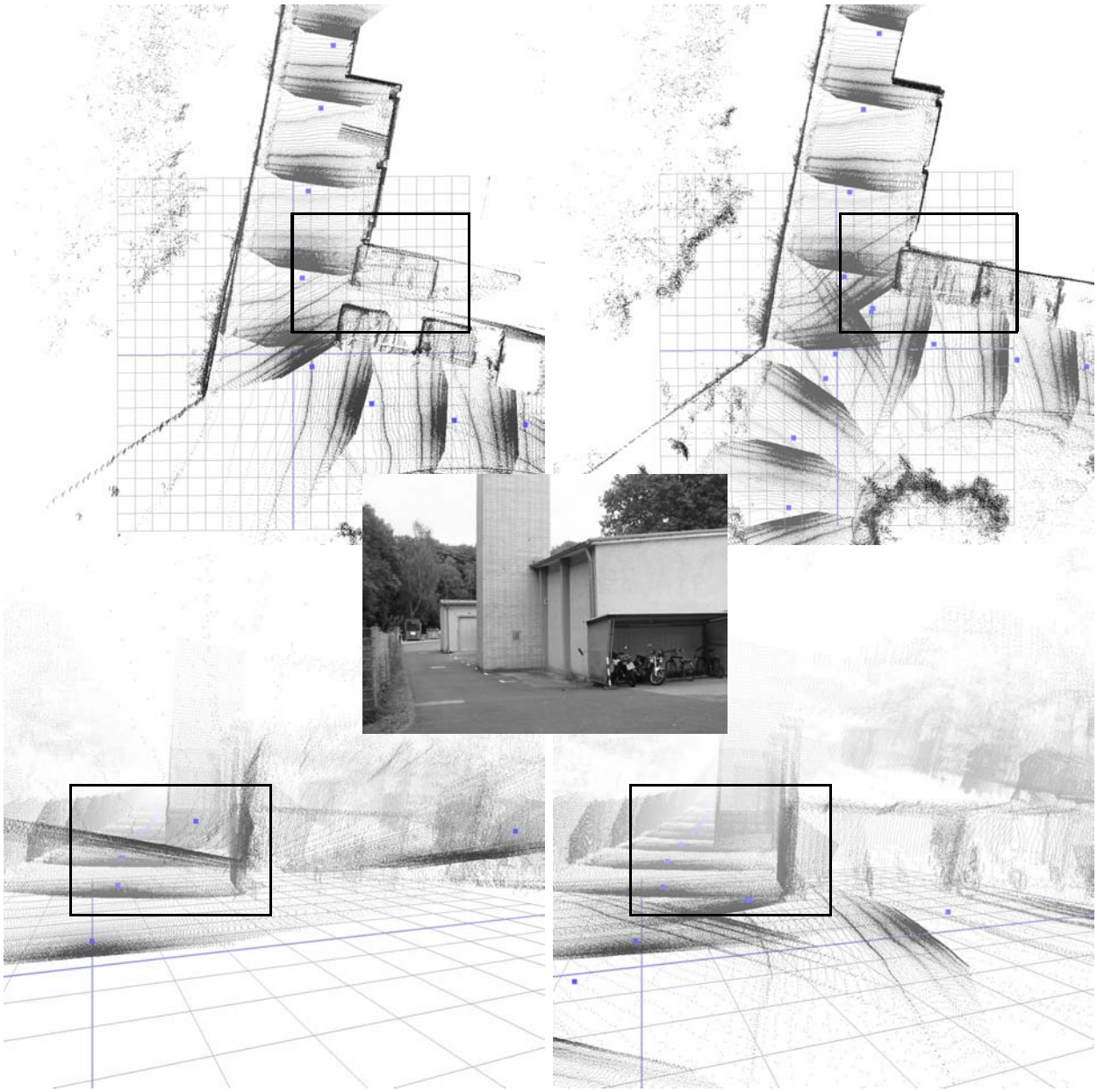


Abbildung 3.24: Vergrößerung eines Ausschnitts aus dem 3D-Modell von einer Fahrt im Freien. Links: Modell vor dem Schließen des Kreises. Rechts: Endgültiges Modell mit globaler Fehlerminimierung. Oben: Ansicht von oben. Unten: Ansicht von vorne.

Fahrt erreicht. Die Genauigkeit in der Position vor dem Schließen des Kreises liegt bei etwa 3 Metern. In Abbildung 3.24 links unten sieht man, dass die Orientierung der 3D-Scans bezüglich des Roll- und Nickwinkels stark abweicht. Der Gierwinkel ist bereits ziemlich exakt. Nach der globalen Relaxation des Modells sind alle Scans konsistent registriert. Tabelle 3.1 vergleicht Längen im 3D-Modell mit korrespondierenden Längen im Luftbild und gibt Auskunft über die erzielte Übereinstimmung.

Der linke Teil der Abbildung 3.25 zeigt das 3D-Modell, wenn nur der Fehler beim Schließen des Kreises verteilt wird, aber anschließend keine globale Optimierung stattfindet. Im unteren Teil, d.h. in dem Bereich, wo kein Kreis geschlossen wurde, sind kleinere Fehler zu erkennen. Offensichtlich

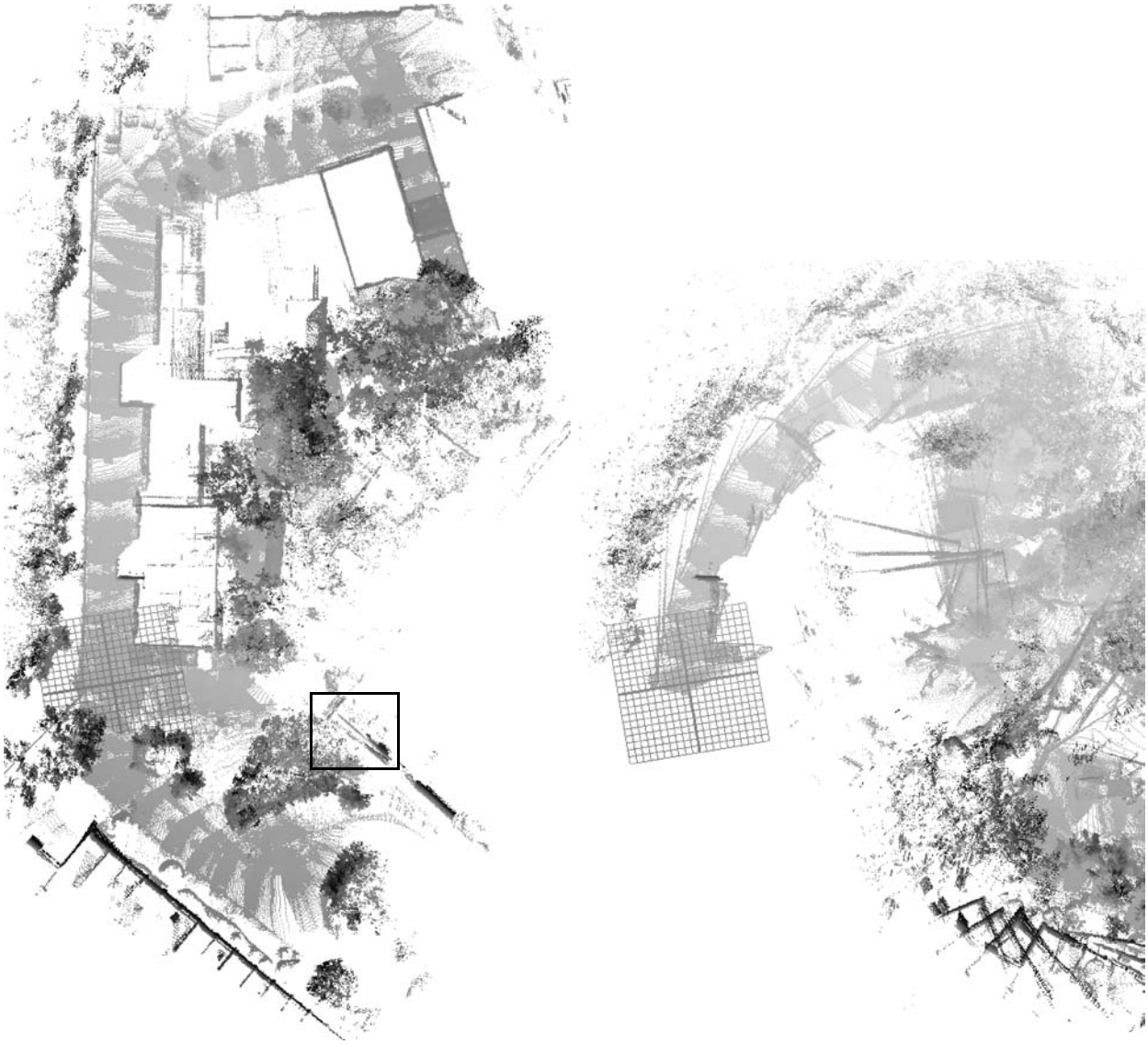


Abbildung 3.25: 3D-Modell einer geschlossenen Fahrt im Freien (Forts.). Links: 3D-Modell *ohne* Schritt 4 des globalen Registrierungsalgorithmus (vgl. Seite 41). Rechts: Registrierung von 3D-Scans unter ausschließlicher Verwendung der Odometrie.

ist die globale Relaxation in der Lage, solche kleinen Fehler auszugleichen.

Das 3D-Modell, das entsteht, wenn ausschließlich Odometrie-Informationen benutzt werden, ist im rechten Teil der Abbildung 3.25 wiedergegeben. Der anfängliche systematische Fehler, der auf einem asphaltierten Streckenabschnitt entsteht, weicht großen zufälligen Fehlern auf dem Wiesenabschnitt.

Die Abbildungen 3.26, 3.27 und 3.28 geben die Trajektorie des Roboters für das Experiment in Dreitafelprojektion wieder. Jeweils oben links befindet sich die (x, z)-, oben rechts die (y, z)- und unten die (x, y)-Karte. Abbildung 3.26 stellt den Weg vor dem Schließen des Kreises dar, während Abbildung 3.27 jenen nach der globalen Optimierung zeigt. Bemerkenswert ist, dass der Kreis auf-

Tabelle 3.1: Vergleich der Längenverhältnisse im Luftbild und in der 3D-Szene (vgl. Abbildung 3.23).

1. Linie	2. Linie	Verhältnis im Luftbild	Verhältnis im 3D-Modell	Abweichung
AB	BC	0.683	0.662	3.1%
AB	BD	0.645	0.670	3.8%
AC	CD	1.131	1.141	0.9%
CD	BD	1.088	1.082	0.5%

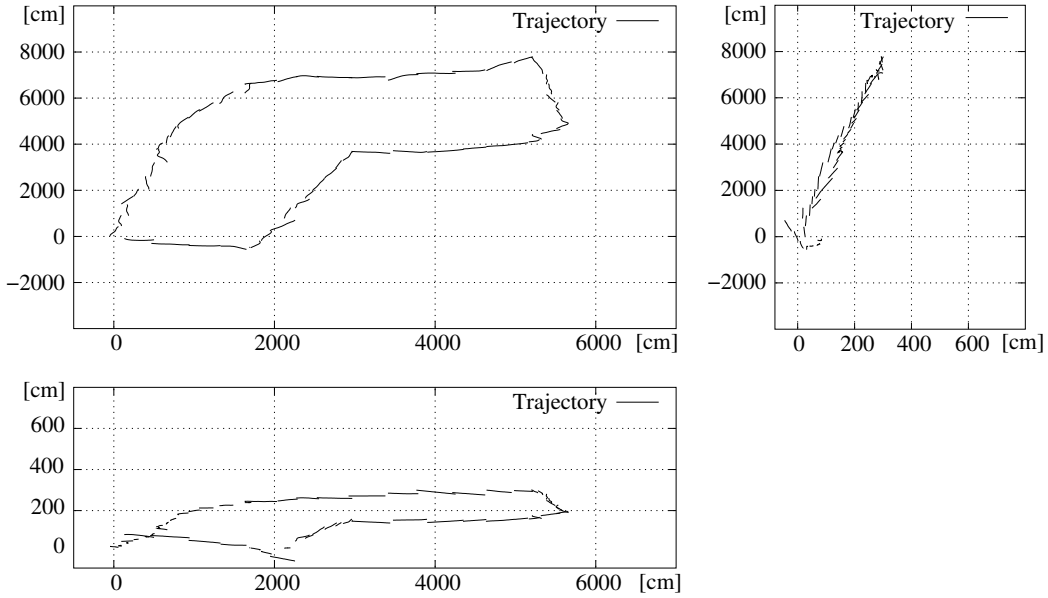


Abbildung 3.26: Trajektorie vor dem Schließen des Kreises in Dreitafelprojektion. Oben links: (x, z)-Koordinaten der Roboterpose (Ansicht von oben). Oben rechts: (z, y)-Koordinaten (Ansicht von rechts). Unten: (x, y)-Koordinaten (Ansicht von vorn).

geweitet und die Höhenkoordinaten korrigiert wurden. Die Trajektorie ist nicht geschlossen. Durch den Versatz an den Unterbrechungsstellen zeigt sich, wie die Pose durch das Scanmatching korrigiert wurde. Abbildung 3.28 visualisiert hingegen einen geschlossene Bahnkurve. Für die Erstellung der Grafik wird die Transformation $\Delta\mathbf{P}$, die den Endpunkt eines Wegstückes auf den Startpunkt des Nachfolgers abbildet, errechnet. D.h. für den $i-$ten Teilabschnitt ist

$$\Delta\mathbf{P} = \mathbf{P}_{i-1,\text{back}} \cdot \mathbf{P}^{-1}_{i,\text{front}},$$

wobei $\mathbf{P}$ ist die Matrixdarstellung der Roboterpose ist. Anschließend wird die errechnete Transformation $\Delta\mathbf{P}$ auf $\mathbf{P}_{i,\text{front}}$ angewandt. Danach verteilt der Algorithmus durch Interpolation den j-ten Anteil von $\Delta\mathbf{P}$ auf die verbleibenden n Posen ($1 \leq j \leq n$) des Teilstücks. Dabei berechnet sich jener Anteil analog zu den Formeln (3.42) und (3.44).

Abbildung 3.29 visualisiert vier weitere Ansichten des 3D-Modells. Dabei korrespondieren die Aufnahmeorte in der 3D-Szene in etwa mit jenen von Kurt3D in Abbildung 3.20. Neben den 3D-Daten sind die Scanposen und die geglättete Trajektorie eingezeichnet. Die Rechenzeiten für die beiden „closed-loop“-Modelle der Abbildungen 3.18 und 3.23 (rechts) sind in Tabelle 3.2 angegeben.

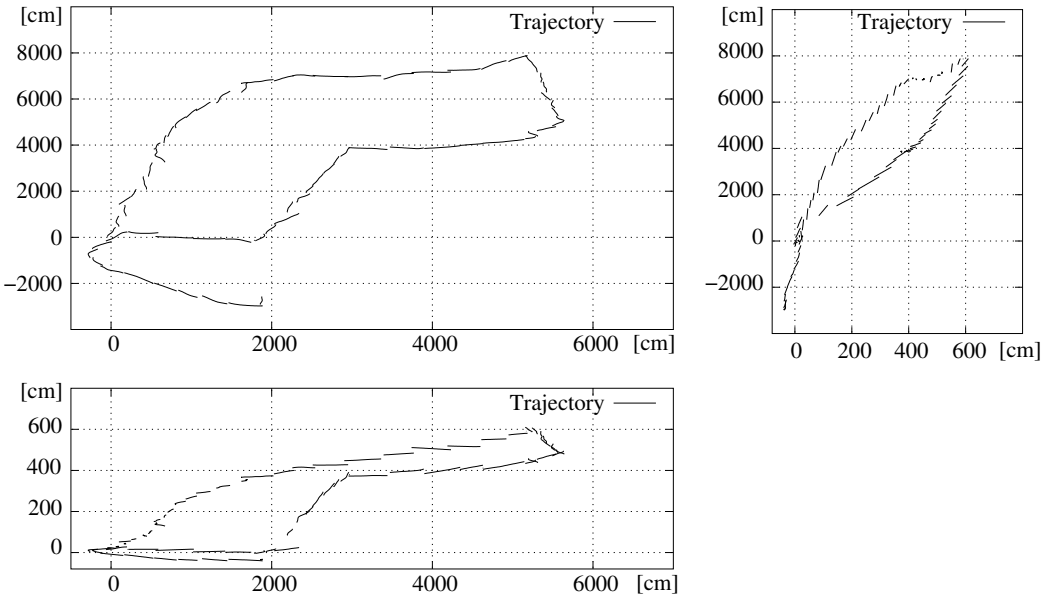


Abbildung 3.27: Trajektorie nach dem Schließen des Kreises in Dreitafelprojektion. Oben links: (x, z)-Koordinaten der Roboterpose. Oben rechts: (z, y)-Koordinaten. Unten: (x, y)-Koordinaten.

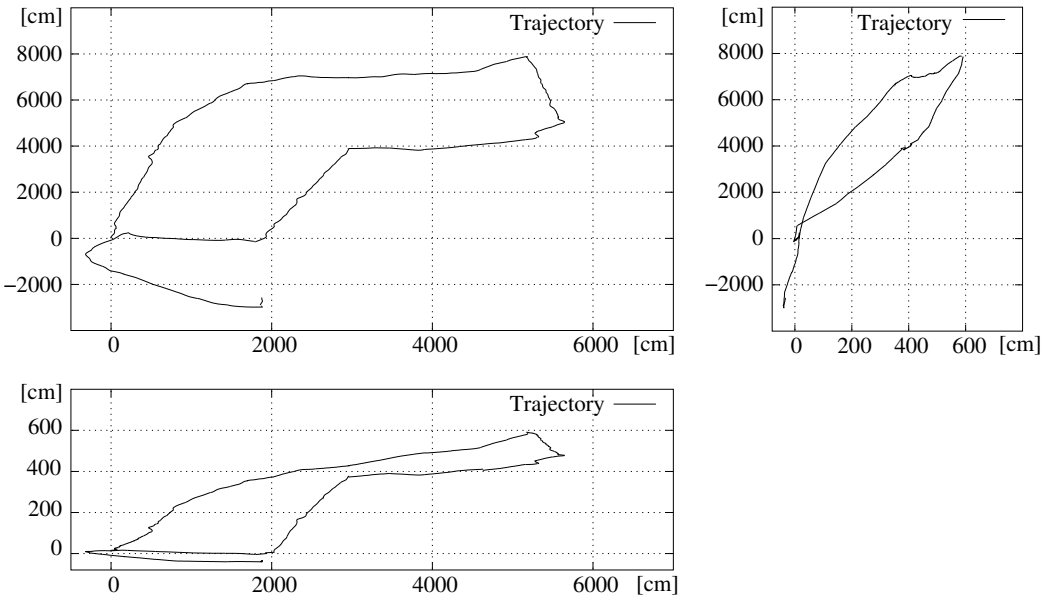


Abbildung 3.28: Trajektorie nach dem Schließen des Kreises und der Interpolation. Oben links: (x, z)-Koordinaten der Roboterpose. Oben rechts: (z, y)-Koordinaten. Unten: (x, y)-Koordinaten.

3.4 Diskussion

Das simultane Lokalisations- und Kartierungsproblem (SLAM) ist eines der wichtigen Probleme in der Robotik, dem in letzter Zeit viel Aufmerksamkeit gewidmet wurde [63,81,136,199,201]. Um eine

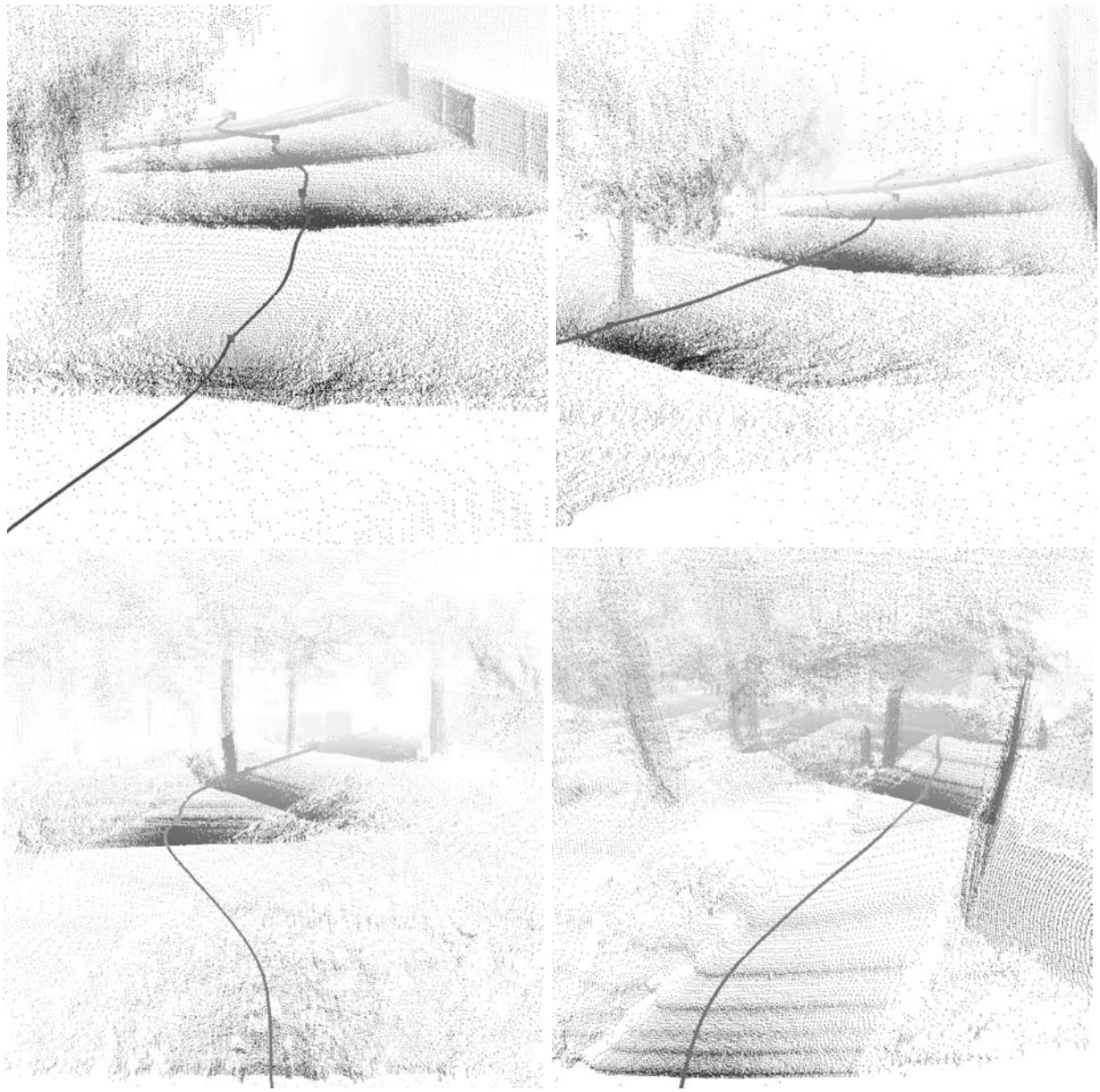


Abbildung 3.29: Detailansichten des resultierenden 3D-Modells. Die Abbildungen korrespondieren mit den Fotos aus Abbildung 3.20.

unbekannte Umgebung zu kartieren, verwenden herkömmliche Systeme probabilistische Methoden und punktförmige Landmarken. Dabei ist SLAM ein Schätzproblem: Die zu schätzenden Parameter sind die Roboterpose $\mathbf{P}$ zu verschiedenen Zeitpunkten sowie die Position der n Landmarken. Beide zu schätzenden Parameter werden zunächst durch Messungen bestimmt. Des Weiteren wird angenommen, dass ein *a-priori* Modell für die Messunsicherheit bekannt ist.

Bewegt sich ein Roboter durch ein bekanntes Gebiet, d.h. ist eine Karte gegeben, kann die Unsicherheit in der Roboterpose niedrig gehalten werden. Die Observation einer Landmarke ermöglicht es, die Roboterpose bis auf die Unsicherheiten der Landmarke und der Observation zu schätzen. Fährt der Roboter jedoch durch unbekanntes Gebiet, wird die Unsicherheit in der Roboterpose beliebig

Tabelle 3.2: Rechenzeit für die 3D-Kartierungen des Robotikpavillons (vgl. Abb. 3.18) und des Fraunhofer-Campus in Birlinghoven (vgl. Abb. 3.23, rechts). Angegeben sind die Gesamtrechenzeit in Abhängigkeit von der Methode zur Bestimmung der Punktpaare sowie die durchschnittliche Anzahl der ICP-Iterationen. Bemerkung: Das Robotikpavillonmodell wurde mit höherer Genauigkeit erzeugt, als jenes des FhG - Campus.

3D-Modell	Verwendete Punkte & Such Methode	Zeit	# Iter.
Robotikpavillon	Scanmatching mit allen Punkten & brute force	144 h 5 min	2080
Robotikpavillon	Scanmatching mit allen Punkten & kD-Baum	12 min 23 s	2080
Robotikpavillon	Scanmatching mit allen Punkten & Apx-kD-Baum	10 min 1 s	2080
Robotikpavillon	Scanmatching mit *reduzierten Punkten* & Apx-kD- Baum	<1 min 32 s	2176
Robotikpavillon	6D-SLAM (Schritte 1 - 4) mit *reduzierten Punkten* & Apx-kD-Baum	38 min	∼16000
FhG - Campus	6D-SLAM (Schritte 0 - 4) mit *reduzierten Punkten* & Apx-kD-Baum	1 h 54 min	∼48400

groß: Fehler in der Odometrie addieren sich. Sich vorbeibewegende Landmarken dienen lediglich dazu, die Unsicherheit in der Poseschätzung zu reduzieren. Für viele Sensoren konnte gezeigt werden, dass ein solches Tracking wesentlich besser funktioniert als Odometrie alleine [127, 198]. Aber das ändert nichts an dem prinzipiellen Problem der Fehlerakkumulation. Dadurch unterscheidet sich das SLAM-Problem von anderen Schätzproblemen. Abhilfe versprechende Sensoren, wie Kompass oder GPS/GLONASS, sind in sehr vielen Anwendungsszenarien, z.B. in Innenräumen, nicht einsetzbar.

Eine besondere Rolle in bekannten SLAM-Algorithmen spielen die geschlossenen Kreise. Wenn ein Roboter eine Position wieder erkennt und Landmarken richtig zuordnet, können die Fehler begrenzt werden. Dazu deformieren die Algorithmen die bereits kartierte Umgebung so, dass ein (topologisch) konsistentes Modell entsteht. Folgende drei probabilistische Verfahren lösen das SLAM-Problem:

Maximum Likelihood-Schätzung. Nimmt man unabhängige, gaußverteilte Messfehler mit bekannter Kovarianz an, lässt sich SLAM durch das Minimieren einer quadratischen Fehlerfunktion $Q(\mathbf{x})$ (nicht-lineares model fitting) lösen [81]:

$$\begin{aligned} Q(\mathbf{x}) &= \sum_i \frac{1}{2} \left(\mathbf{y}_i - \mathbf{f}_i(\mathbf{x})\right)^T \mathbf{C}_i^{-1} \left(\mathbf{y}_i - \mathbf{f}_i(\mathbf{x})\right) \\ \mathbf{x}_{\max} &= \arg\min_{\mathbf{x}} Q(\mathbf{x}). \end{aligned} \tag{3.47}$$

Damit ist $S_\gamma := \{\mathbf{x} | Q(\mathbf{x}) - Q(\mathbf{x}_{\max}) < \gamma\}$ die Region, die $\mathbf{x}_{\max}$ umgibt und mit dem Parameter γ die Unsicherheit in der Karte definiert. Die Posen des Roboters bilden den Vektor $\mathbf{x}$. $\mathbf{y}_i$ ist die i-te Messung mit der zugehörigen Kovarianzmatrix $\mathbf{C}_i$ und $\mathbf{f}_i(\mathbf{x})$ ist die korrespondierende Messfunktion, d.h. sie gibt denjenigen Wert wieder, den die Messung haben sollte, wenn die Landmarke und die Roboterpose bei $\mathbf{x}$ wären.

Für die Minimierung und demzufolge auch für das Finden einer guten Karte kann der Levenberg-Marquardt Algorithmus verwendet werden (vgl. Anhang B.4.2). Ein auf diesem Verfahren basierender inkrementeller Algorithmus mit p Roboterposen und n Landmarken benötigt jedoch eine Rechenzeit der Größenordnung $\mathcal{O}((n+p)^3)$ und ist daher nicht praktikabel.

Erweiterter Kalman Filter. Der meistgenutzte Algorithmus für SLAM ist der erweiterte Kalman Filter. Der Filter integriert alle Messungen in eine Kovarianzmatrix der Landmarkenpo-

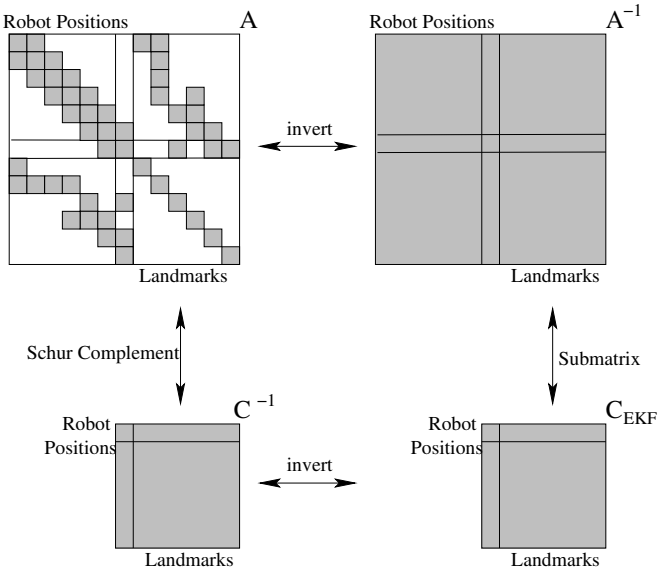


Abbildung 3.30: Zusammenhang zwischen der Informationsmatrix $\mathbf{A}$ und der Kovarianzmatrix $\mathbf{C}_{\text{EKF}}$ des erweiterten Kalman Filters [81].

sitionen und der Roboterposen. Für lineare Messmodelle entspricht dies genau der Maximum Likelihood-Schätzung, im nicht-linearen Fall nur, falls alle Messfunktionen linearisiert werden können. Die Linearisierung geschieht zumeist zum Zeitpunkt der Messung [201]. Allerdings kann der Punkt der Linearisierung signifikant falsch sein, wenn der Roboter durch noch nicht kartiertes Gebiet fährt [81]. Gerade der Fehler bezüglich der Orientierung ist kritisch, da Winkelfunktionen nur einen kleinen quasi-linearen Bereich aufweisen. Die Folge falscher Linearisierungen ist, dass kartierte, kreisförmige Umgebungen meistens zu groß sind [81].

Informationsfilter. Linearisiert man die Messfunktionen $\mathbf{f}_i$, ist die Zielfunktion Q quadratisch:

$$Q(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{b} + c. \tag{3.48}$$

Die Matrix $\mathbf{A}$ heißt Informationsmatrix. Wiederum wird hierbei zum Zeitpunkt der Messung linearisiert. $\mathbf{A}$ ist dünn besetzt (engl.: *sparse*) und hat nur von Null verschiedene Einträge an den Eintragspaaren, bei denen gemeinsame Messungen vorliegen. Die inverse Matrix $\mathbf{A}^{-1}$ ist die vollständige Kovarianzmatrix für alle Positionen der Landmarken und alle Roboterposen. Daher repräsentiert $\mathbf{A}^{-1}$ auch indirekte Zusammenhänge. Eine Teilmatrix davon ist die Kovarianzmatrix $\mathbf{C}$, die im erweiterten Kalman Filter vorkommt. Abbildung 3.30 stellt den Zusammenhang zwischen Informationsmatrix und der Kovarianzmatrix des erweiterten Kalman Filters dar.

Sparse Extended Information Filter (SEIFs) dienen dazu, Informationsmatrizen zu verarbeiten. Um lineare Rechenzeit zu gewährleisten, achten die Algorithmen darauf, dass stets dünn besetzte Ergebnismatrizen entstehen [201].

Die obigen SLAM-Algorithmen setzen eine Merkmalsextraktion voraus. Im Vergleich dazu ist der in dieser Arbeit entwickelte SLAM-Algorithmus unabhängig von Landmarken. Lu und Milios umgehen

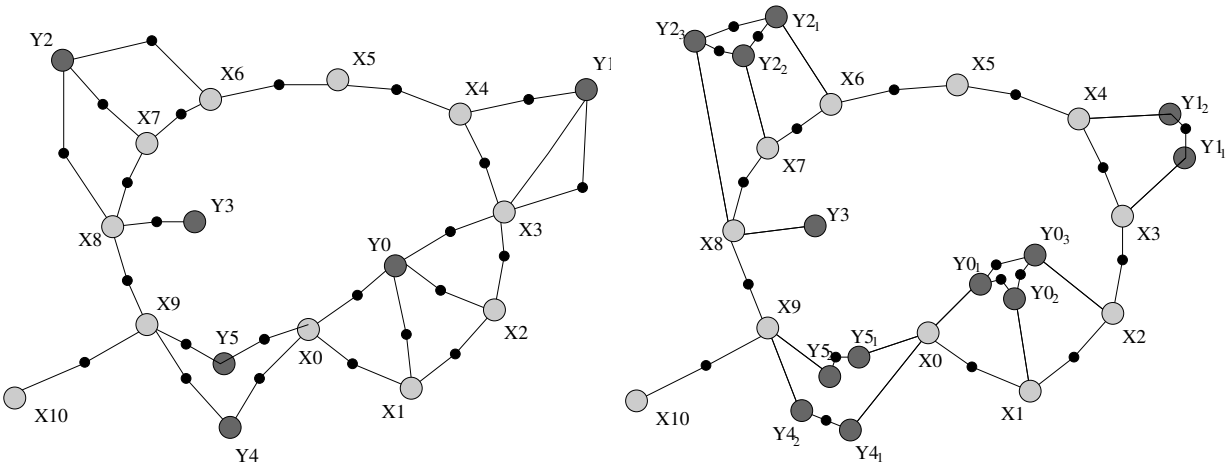


Abbildung 3.31: Vergleich der SLAM-Ansätze. Links: Probabilistisches Verfahren. Sowohl die Posen als auch die assoziierten Landmarken besitzen Wahrscheinlichkeitsverteilungen. Die globale Relaxation versucht nun, das Modell zu entspannen (die Landmarken sind fixiert und Linien mit kleinen schwarzen Punkten repräsentieren beweglichen Verbindungen/Federn) [96]. Rechts: Der hier vorgestellte Algorithmus hinterlegt den Messdaten keine Wahrscheinlichkeitsverteilung, betrachtet sie also als korrekt. Die Posen werden auf der Grundlage des Matchings verschoben, die Datenassoziation besteht aus der Suche der nächsten Nachbarn.

das Problem der Merkmalsextraktion, indem sie ganze 2D-Scans vergleichen [136]. Um dies zu erreichen, assoziieren sie mit jeder Roboterpose einen 2D-Scan. Wenn zwei Scans überlappende Regionen einer Umgebung abdecken, wird eine Relation der zugehörigen Roboterposen auf der Grundlage des Scanvergleichs abgeschätzt. Der resultierende Graph von relativen Posen kann dann genauso behandelt werden, wie Landmarken-basiertes SLAM. Eine Erweiterung dieses Verfahrens auf 3D-Scans und 6 Freiheitsgrade ist nicht bekannt.

Ein Schlüsselproblem der bekannten SLAM-Algorithmen ist das Datenassoziationsproblem, bei dem die Frage beantwortet werden muss, welche Landmarken miteinander korrespondieren. Das SLAM-Verfahren in dieser Arbeit basiert ausschließlich auf Nächste-Nachbar-Relationen und umgeht somit das Problem. Durch iterative Bestimmung der nächsten Nachbarn nimmt man an, dass die nächsten Nachbarn im letzten Iterationsschritt den tatsächlichen entsprechen.

Die probabilistischen SLAM-Algorithmen wurden bisher noch nicht erfolgreich zur Gewinnung von 3D-Modellen erweitert. Ein senkrecht zur Fahrtrichtung scannender Laser wird mitgeführt, um auf der Basis von 2D-Positionen 3D-Modelle zu erzeugen [89,90,199]. Dies dient aber hauptsächlich zur Evaluation der planaren Verfahren (vgl. Abschnit 2.2). Auch fehlt den probabilistischen Verfahren eine effiziente, geschlossene Berechnung der Transformation. Abbildung 3.31 zeigt einen schematischen Vergleich der SLAM-Methoden als dynamisches Netzwerk, das entspannt werden muss. Dabei werden die Verbindungen mit schwarzen Punkten als Federn betrachtet, die das Gebilde zu einem energetischen Minimum hin bewegen.

3.5 Weitere Anwendungsbeispiele

Die vorgestellten Algorithmen für 6D SLAM wurden in erster Linie entwickelt, um 3D-Karten mit dem Roboter Kurt3D zu bauen. Das Gesamtsystem wird in Kapitel 5 vorgestellt. Der folgende Abschnitt schildert Anwendungen der Algorithmen, bei denen die 3D-Daten mit anderen Systemen aufgenommen wurden. Zuerst wird die 3D-Kartierung von Minen beschrieben, bei denen die Daten von der CMU zur Verfügung gestellt wurden. Anschließend fügen die Algorithmes des Scanmatchings Gesichtsprofile zusammen, wobei die Daten vom Bonner Forschungsinstitut `caesar` stammen.

3.5.1 Scanmatching für die autonome Exploration von Minen

Die genaue Kartierung von Minen und anderen Objekten des Bergbaus ist von größter Bedeutung für die Gesellschaft [138], da von inaktiven Minen große Gefahren ausgehen können. Einem aktuellen Artikel zufolge [26] existieren weltweit zehntausende oder gar hunderttausende verlassene Minen. Das Fehlen akkurater Karten führt häufig zu Unfällen, beispielsweise zu jenem in Quecreek, PA, U.S.A. [158]. Strukturelle Veränderungen in verlassenen Minen können zu Erdeinbrüchen führen, wobei sowohl die Menschen als auch die Bebauung auf der Oberfläche Schaden nehmen können. Auch der Qualität des Grundwassers muss in Regionen mit Bergbau besondere Beachtung zukommen. Roboter könnten in kartierten Minen fahren und Proben entnehmen. Des Weiteren erlaubt eine genaue Kartierung von verlassenen Minen Rückschlüsse auf das entnommene Volumen. Dies ist von entscheidener Bedeutung, wenn es darum geht, eine bereits (teil-)ausgebeutete Mine zu reaktivieren.

Da die meisten Karten im Bergbau zweidimensional sind, können keine Aussagen bezüglich der genannten Fragestellungen gemacht werden. Genaue 3D-Karten von Bergbauobjekten werden daher dringend benötigt [153]. Die gefährlichen Arbeitsbedingungen in verlassenen Minen und der oftmals schwierige Zugang können durch den Einsatz von autonomen Kartierungsrobotern überwunden werden. Die Kartierung von Minen durch Roboter ist nicht neu, allerdings wurde sie bisher nur zur Navigation von Robotern in noch aktiven Minen benutzt. Zum Beispiel haben Corke et al. Maschinen gebaut, die akkurate 2D-Karten konstruieren und benutzen [55]. Auch Baily berichtet von 2D Kartierungen unterirdischer Gebiete [24].

Abbildung 3.32: Der Groundhog Roboter der CMU.

Der Roboter „Groundhog“ der CMU benutzt 2D-Scans und in diskreten Abständen aufgenommene 3D-Scans für die Exploration von Minen (vgl. Abbildung 3.32). Groundhog wurde 2003 von der CMU entwickelt [73,200] und besteht aus zwei Vorderteilen eines Geländefahrzeugs [153], so dass alle Räder Antrieb besitzen und gelenkt werden können. Durch diese zweiseitige Ackermann-Lenkung [70] ergibt sich ein Wendekreis von zirka 2.44 m. Ein hydraulischer Zylinder und ein Potentiometer-Feedback ermöglichen einen geschlossenen Regelkreis der Lenkung. Die maximale Geschwindigkeit des Roboters beträgt 0.145 m/sec, wobei eine Leistung von mehr als 1 kW konsumiert wird. Die Sensoren benötigen nur 25 W und der Boardcomputer 75 W. Daher hat die Zeit, die für die Auf-

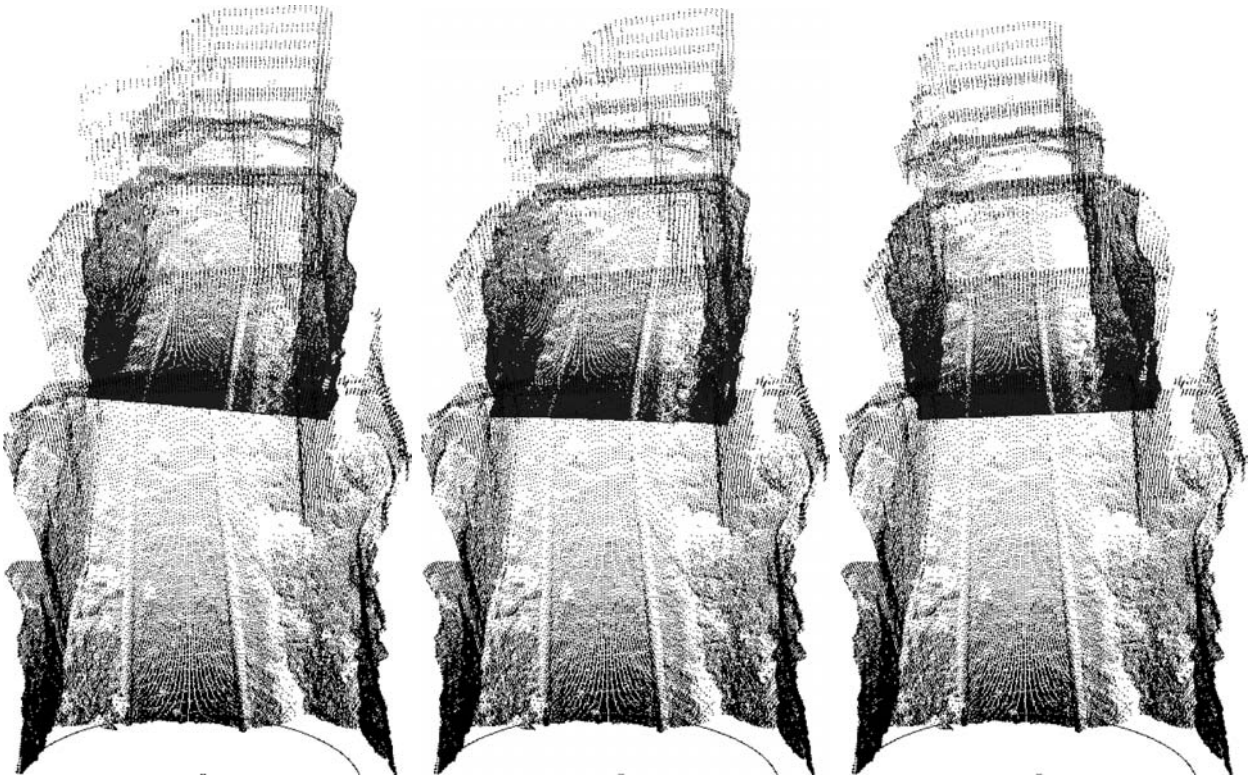


Abbildung 3.33: Links: Initiale, Odometrie-basierte Pose zweier 3D-Scans. Mitte: Pose nach fünf Iterationen des ICP-Algorithmus. Rechts: Registrierungsergebnis. Anhand vorhandener Merkmale (hier: Schienen) lässt sich die Qualität des Matchings überprüfen.

nahme und Verarbeitung von Sensordaten nötig ist, nur sehr geringen Einfluss auf die Betriebszeit des Systems, dessen Stromversorgung auf 6 großen Bleiakkumulatoren basiert. Die Maximalreichweite wird mit mehr als 3 Kilometern angegeben. Mit Hilfe der großen Räder und der daran anliegenden hohen Drehmomente kann Groundhog Schienen und kleinere Hindernisse in den zu explorierenden Minen überwinden. Zum Akquirieren von 3D-Scans ist dieser Minen-Roboter mit zwei 3D-Laserscannern ausgestattet. An diskreten Punkten stoppt der Roboter und nimmt 3D-Daten auf. Diese Daten nutzt er zur lokalen Wegeplanung und Hindernissvermeidung [25], aber es lassen sich darauf auch die Scanmatching-Algorithmen anwenden [153]. Abbildung 3.33 zeigt die anfängliche Lage zweier 3D-Scans (links), ein Zwischenergebnis und ihre Registrierung.

Am 30. Mai 2003 fuhr der Roboter Groundhog nach etlichen Tests in die Mathies Mine in der Nähe von Pittsburgh. Um genaue 3D-Karten aufzunehmen, wurde der Roboter so programmiert, dass er autonom dem Hauptgang folgt. 250 Meter tief in der Mine (nach einer Fahrstrecke von 350 m) entdeckte er einen eingestürzten Deckenbalken [25, 153]. Der Roboter traf die richtige Entscheidung und kehrte um.

Die Algorithmen des 6D SLAM (vgl. Seite 41) lassen sich auf die Datensätze des Groundhog Roboters anwenden. Dabei werden alle 6 Freiheitsgrade der Roboterpose bestimmt. Abbildung 3.34 zeigt die zx-Karte (Ansicht von oben) und die yx-Karte (Ansicht von rechts) in Parallelprojektion. Man sieht, dass der Roboter eine Höhenunterschied von zirka 4 m überwinden musste.

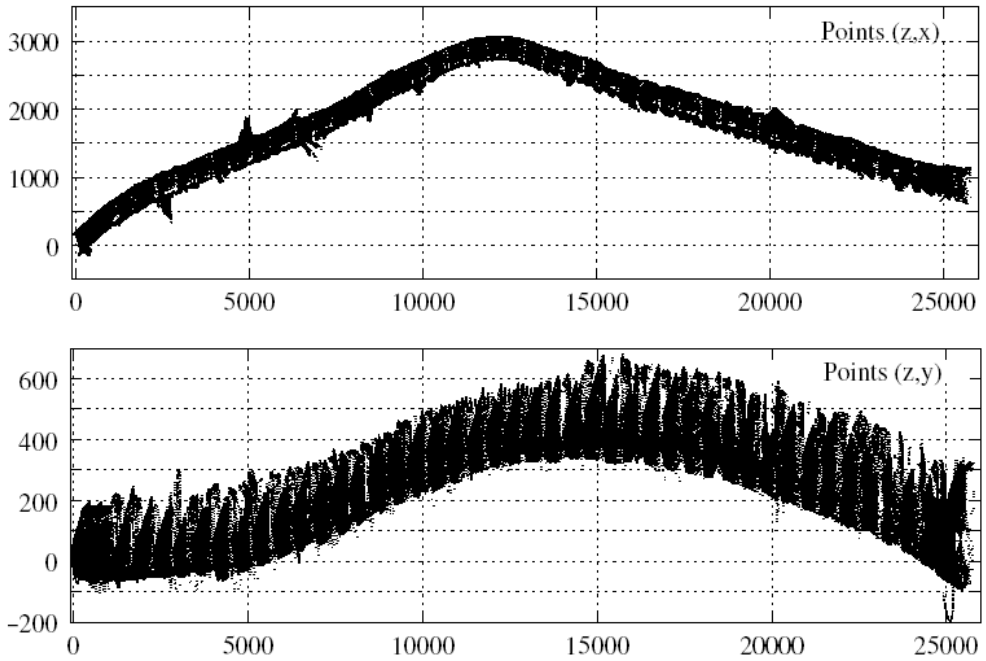


Abbildung 3.34: Vollständige 3D-Karte der Mathies Mine. Oben: zx-Karte (Ansicht von oben). Unten: yx-Karte (Ansicht von der Seite). z ist die Tiefenachse und y die Höhenachse. Einheiten in cm [153].

3.5.2 Matching von Gesichtsprofilen

Für die Behandlung von Krankheiten, Verletzungen und angeborenen Deformierungen an Kopf und Kinn müssen Kieferorthopäden komplexe Operationen ausführen. Zum Beispiel erfordert die Korrektur von entstellenden Behinderungen im Gesichtsbereich eine Manipulation von Gesichtsknochen mit maximaler Genauigkeit. Eine im Vorfeld der Behandlung stattfindende Simulation erlaubt eine genaue Planung des Eingriffs; sie erfordert allerdings ein exaktes 3D-Modell vom Gesicht des Patienten.

Dieser Abschnitt beschreibt einen Ansatz, ein solches 3D-Modell am Computer mit Hilfe eines schnellen Aufnahmeverfahrens und des 3D-Scanmatchings zu generieren [94,95]. Die Daten wurden mit einem holographischen Verfahren aufgenommen, das am Bonner Forschungsinstitut `caesar` entwickelt wurde. Abbildung 3.35 zeigt den Versuchsaufbau. Das gepulste Hologramm hält das Portrait des Patienten mit einem einzigen Laserimpuls fest (Impulsdauer 35 ns). Dieses so genannte Master-Hologramm umfasst die gesamte 3D-Information. Bewegungen des Patienten spielen wegen der extrem kurzen Aufnahmezeit keine Rolle.

Die Portrait-Hologramme werden mit einer holographischen Kamera des Typs GP-2J (Marke Geola) aufgenommen. Diese Kamera hat eine Wellenlänge von 526.5 nm, dringt nur gering in menschliche Haut ein und minimiert somit die Lichtdiffusion [95]. Durch das Auswählen der Moden wird eine Kohärenzlänge von ungefähr 6 Metern erreicht. Der Laser wird in drei Strahlen aufgespalten: Zwei sorgen fuer eine homogene Illumination des Objekts. Dazu werden sie durch konkave Linsen und Diffusorplatten am Ausgang des Lasers ausgeweitet. Der dritte Strahl dient als Referenz. Phasen-Hologramme entstehen aus der Hologrammplatte (30 cm $\times$ 40 cm, VRP-M Emulsion von

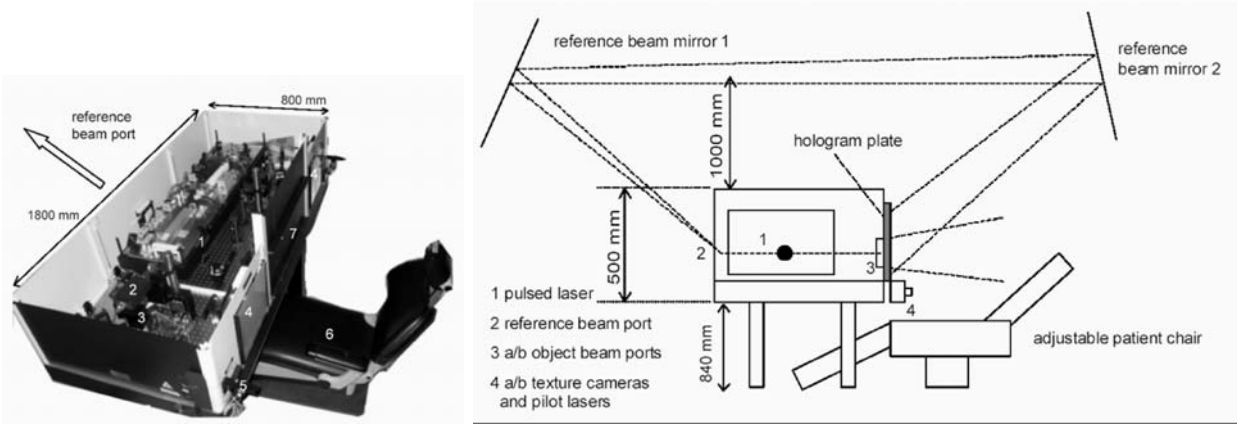


Abbildung 3.35: Versuchsaufbau zur Aufnahme von Hologrammen. Links: Vorrichtung für Gesichtsprofile. Rechts: Schematische Darstellung. Quelle: [94].

Slavich) durch die Entwicklung mit SM-6 und das Bleichen mit PBU-Amidol. Ein cw Nd:YAG Laser (COHERENT Verdi V-2) doppelter Frequenz dient zur Rekonstruktion des holographisch-realen Bildes. Ein Diffusor (Dicke 40 m, Durchmesser 380 mm) bewegt sich computergesteuert durch das holographisch-reale Bild (Positionierungsgerät PI M-531.DD). Dadurch können mit einer Digitalkamera (KODAK Megaplus ES 4.0) 2D-Bilder des Volumens angefertigt werden. Diesen Vorgang bezeichnet man als Hologramm-Tomographie [94], die in Abbildung 3.36 illustriert ist.

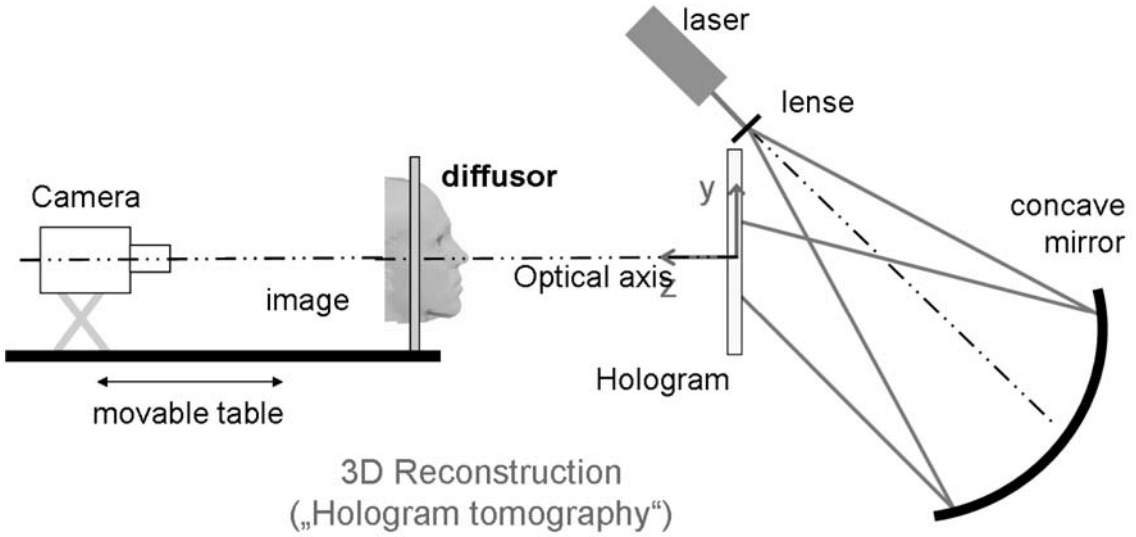


Abbildung 3.36: Hologramm-Tomographie bezeichnet den Vorgang des Abtastens eines Hologramms mit einem Diffusor. Anschließend können aus den Kamerabildern Tiefenbilder berechnet werden. Quelle: [94].

Der Ort des Fokus. In einem nächsten Verarbeitungsschritt wird aus den Bildern des Volumens die 3D-Information rekonstruiert. Durch die Bewegung des Diffusors durch das Hologramm zeigen die Bilder im Fokus die Oberflächenkontur des Patienten. Diese so genannte Struktur-durch-Fokus-Technik (engl.: *structure from focus* bzw. *locus of focus*) der Bildverarbeitung wurde zuerst von Stetson eingeführt [186].

Für die Analyse der Sequenz der 2D-Projektionen nimmt man an, dass das Objekt keine Hinterschneidungen besitzt. Daher kann es keine zwei Punkte geben, die bei gleicher (x, y)-Koordinate einen unterschiedlichen Tiefenwert besitzen. Wie bereits erwähnt, enthält jedes 2D-Bild die Information über die Objektform im Fokus und einen defokussierten Hintergrund. Nun muss zwischen den fokussierten und nicht-fokussierten Regionen unterschieden werden. Um die Schärfe eines Bildes zu beurteilen wird folgendes Verfahren verwendet: Die Schärfe wird mit der statistischen Varianz $V_{(x,y)}$ der Lichtintensität der Pixel in der Nachbarschaft von (x, y) bestimmt. Für jede Koordinate (x, y) ist das Maß der Schärfe $V_{(x,y)}(z)$ eine positive reelle Zahl. Die Tiefeninformation $z_{(x,y)}$ muss nun folgende Bedingung erfüllen:

$$V_{(x,y)}(z_{(x,y)}) \geq V_{(x,y)}(z) \qquad \forall\ z.$$

Mit dem Abtastverfahren der 2D-Bilder und der anschließenden Strukturbestimmung aus dem Fokus kann aus einem holographischen realen Bild ein Tiefenbild erzeugt werden [95] (vgl. Abbildung 3.36).

Simultanes Aufnehmen von Reliefs und 3D-Reliefmatching. Um Gesichtsprofile vollständig zu digitalisieren, müssen sie von mehreren Seiten gleichzeitig aufgenommen werden. Dazu werden beispielsweise, wie in Abbildung 3.37 (links) dargestellt, Spiegel verwendet. Aufgrund der großen Kohärenzlänge von 6 Metern kann das durch die Spiegel abgelenkte Laserlicht ebenfalls im Hologramm gespeichert werden. Diese demnach zeitgleich akquirierten Ansichten (vgl. Abbildung 3.37, rechts) können nun mit dem 3D-Scanmatching zu einer konsistenten Szene zusammengesetzt werden. Abbildung 3.38 zeigt zwei Gesichtsprofile (Punktmengen), die mit den Algorithmen aus

Abbildung 3.37: Virtuelles Bild eines Portraithologramms, wobei die Seitenansichten mit zwei Spiegeln erfasst wurden. Quelle: [94].

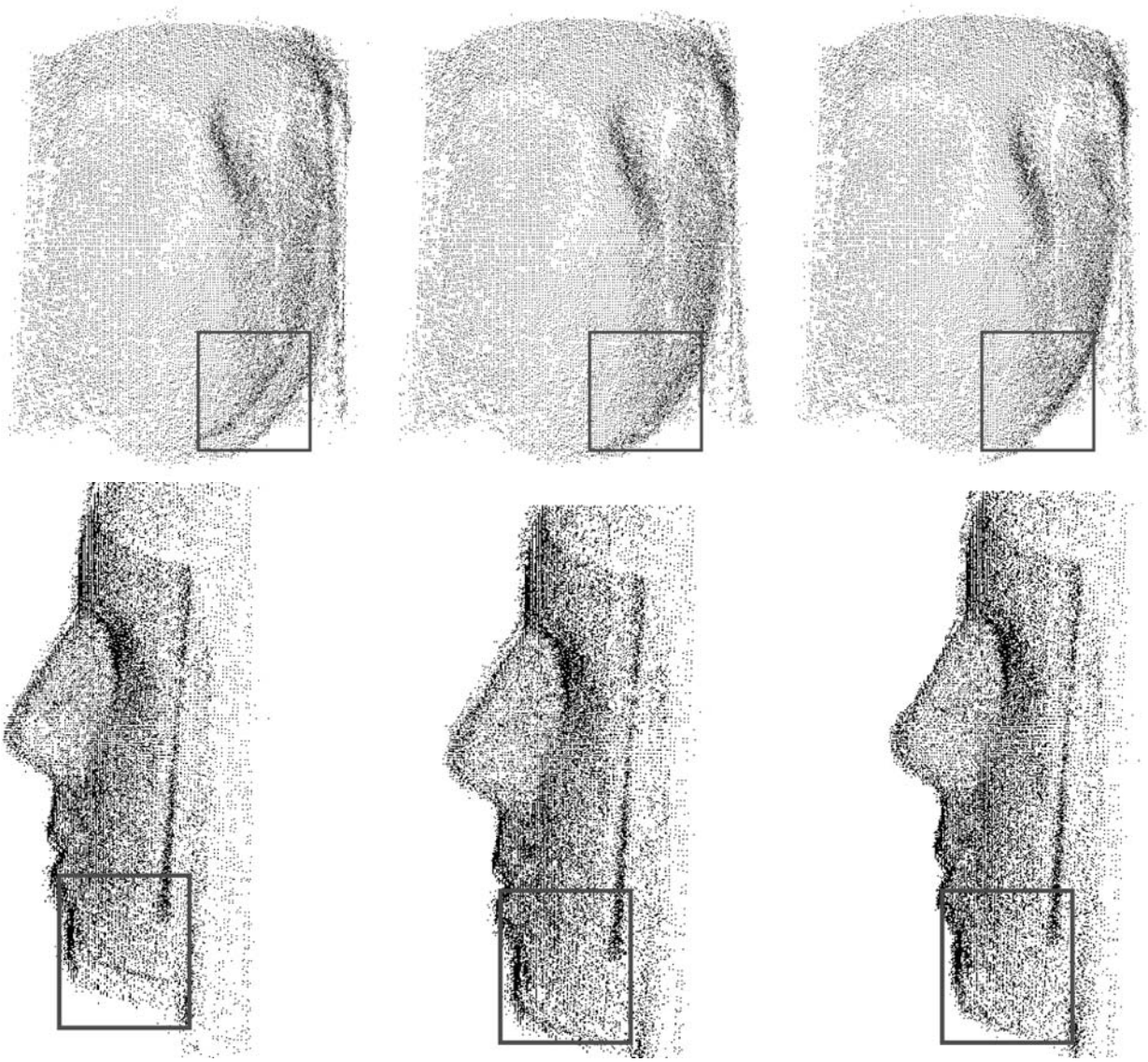


Abbildung 3.38: Registrieren von 3D-Gesichtsmodellen mit dem ICP-Algorithmus. Links: Initiale Ausrichtung. Mitte: Ausrichtung nach 4 Iterationen. Rechts: Abschließende Ausrichtung nach 85 Iterationen.

den vorangegangenen Abschnitten dieses Kapitels registriert werden. Die Startposeschätzung wird dabei fest vorgegeben und basiert auf einer Matrix, die eine Spiegelung ausführt.

Kapitel 4

Die Interpretation von 3D-Scans

Gemäß den Ausführungen in Kapitel 3 lassen sich 3D-Scans zu einer konsistenten Karte der Roboterumgebung zusammensetzen. Das Ergebnis ist eine große Menge von 3D-Punkten. Dieses Kapitel liefert Algorithmen zur Interpretation der Daten.

4.1 Formmatching

Für das Auffinden von Objekten in Bildern werden in der Bildverarbeitung häufig Korrespondenzen zwischen Objekt- und Bildpunkten hergestellt. Diese Korrespondenzen sind nicht frei. Ist eine kleine Menge von Korrespondenzen bestimmt, ist der Rest durch Objektzwänge definiert. Das Formmatching nutzt diese Tatsache aus und repräsentiert das zu matchende 3D-Objekt in einer abstrakten Beschreibung, d.h. durch wenige 3D-Punkte und Attribute.

Der in Abschnitt 3.1.1 vorgestellte Algorithmus zum Scanmatching kann auch auf das Formmatching angewandt werden. Für eine gegebene Form bestimmt sich die Datenmenge D durch die Berechnung des nächsten Punktes ausgehend von der abstrakten Beschreibung. Analog zur Minimierung des Fehlers zwischen zwei 3D-Punktmengen (vgl. Gleichung (3.1)) muss die abstrakte Formbeschreibung anschließend transformiert, d.h. rotiert und translatiert werden.

Bemerkung: Bei dieser Anwendung des ICP-Algorithmus sind die Rollen von M und D scheinbar vertauscht: Das zu matchende Modell bzw. die zu matchende Form liefert die Datenmenge D und wird in den 3D-Scan eingefügt.

4.1.1 Extraktion von 3D-Flächen

Umgebungen, die von Menschenhand geschaffen wurden, so wie Wände, Fußböden und Straßen, weisen in der Regel große ebene Flächen auf. Ein sinnvoller Schritt zur Datenreduktion ist daher die Extraktion von 3D-Flächen aus den aufgenommenen 3D-Scans. Aus der Literatur sind bereits einige Verfahren bekannt [46, 47, 102, 132, 143, 191].

Eine weit verbreitete Technik für das Generieren von 3D-Flächen ist die Regions-Vergrößerung (engl.: *region growing*), wie sie beispielsweise von Hähnel et al. verwendet wird [102]. Diese geht von einem initialen Gittermodell der gescannten Oberfläche aus; dabei sind die Messpunkte durch Dreiecke miteinander verbunden. Der Regions-Vergrößerungsalgorithmus vereinigt iterativ planare,

zusammenhängende Dreiecke. Da vielfach über die Messdaten und Dreiecke iteriert werden muss, ist der Algorithmus sehr rechenaufwändig [102]. Eine Möglichkeit, die Vergrößerungen von planaren Regionen in einzelnen 3D-Scans bereits während der Scanaufnahme durchzuführen, wurde in [191, 192] präsentiert. Der Zeitaufwand zur Vereinigung von Regionen aus mehreren 3D-Scans bleibt davon jedoch unberührt.

Eine weitere bekannte Technik zur Extraktion von Merkmalen aus Messdaten ist der RANSAC Algorithmus, wie er z.B. von Cantzler et al. benutzt wird [46, 47]. RANSAC (Random Sample Consensus) ist ein einfacher Algorithmus zur robusten Einpassung eines Modells in Messdaten [17]. Er gehört zur Kategorie der robusten Schätzverfahren und wird häufig im Bereich des maschinellen Sehens angewandt. Zur Bestimmung einer 3D-Fläche führt RANSAC folgende Schritte aus:

1. Wähle n Messdaten aus M zufällig aus.
2. Schätze damit die Parameter der 3D-Fläche ab.
3. Bestimmte die Anzahl der Datenpunkte $k \in M$, die die 3D-Fläche mit einer gewissen Toleranz repräsentieren.
4. Falls $k > k_0$ ist, dann akzeptiere die Schätzung und gib die 3D-Fläche zurück.
5. Wiederhole die Schritte 1 bis 4 l-mal.
6. Das Verfahren schlägt fehl, wenn dieser 6. Punkt erreicht wird. (Keine 3D-Fläche gefunden)

Der Wert k_0 bestimmt sich aus der Anwendung heraus und ist als Konstante fest vorgegeben. Werden mehrere 3D-Flächen in den Messdaten vermutet, werden diejenigen, die bereits zu einer Fläche gehören, gelöscht und der Algorithmus kann erneut starten. Um l abzuschätzen geht man wie folgt vor: Sei p_{fail} die Fehlerwahrscheinlichkeit des Algorithmus.

$$
\begin{aligned}
p_{\text{fail}} &= \text{Wahrscheinlichkeit des } l\text{-maligen Nichtakzeptierens eines Versuchs} \\
&= (\text{Wahrscheinlichkeit eines nicht akzeptierenden Versuchs})^l \\
&= (1 - \text{Wahrscheinlichkeit, dass ein Versuch akzeptiert wird})^l \\
&= \Big(1 - (\text{Wahrscheinlichkeit, dass ein Datenelement zum Modell passt.})^{|M|}\Big)^l \\
&= \Big(1 - (p_g)^{|M|}\Big)^l .
\end{aligned}
$$

Somit ergibt sich für $l = \frac{\log p_{\text{fail}}}{\log(1-(p_g)^{|M|})}$ [17].

Liu et al. schlagen wiederum eine andere Technik zur Bestimmung von 3D-Flächen aus Entfernungsdaten vor. Dabei benutzen sie die Maximierung des Erwartungswerts (engl.: *expectation maximization (EM)*) zum Erzeugen eines Oberflächenmodells [132]. Ihr Algorithmus schätzt die Anzahl der 3D-Flächen und gleichzeitig deren Position und Ausrichtung durch die Maximierung des Erwartungswerts einer logarithmischen Likelihood-Funktion. Die Parameter der Fläche werden während

des Maximierungsprozesses effizient durch das Einführen von Lagrange-Faktoren und durch die Berechnung von Eigenwerten bestimmt [132]. Der Nachteil dieses Verfahrens liegt darin, dass die Anzahl der 3D-Flächen nicht inhärent, d.h. durch das Funktionsprinzip des Algorithmus selbst, ermittelt werden kann [102].

Eine neue Technik zur Extraktion von 3D-Flächen stammt von Nevado et al. [143]. Ihr Verfahren betrachtet nacheinander alle 3D-Punkte. In einer lokalen Umgebung der Punkte wird eine 3D-Fläche durch die Analyse einer räumlichen Varianzmatrix (engl.: *spatial variance matrix*) abgeschätzt, die für eine gegebene Menge $X = \{\mathbf{x}_i \in \mathbb{R}^3\}_{i=1,\ldots,N}$ folgende Struktur besitzt:

$$\mathrm{Var}[X] = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 \end{pmatrix}.$$

σ_ξ^2 ist die Varianz in ξ-Richtung und σ_{xy} ist die x, y-Kovarianz. Bekannt ist, dass die grundlegenden Achsen, repräsentiert durch die Eigenvektoren, zusammen mit dem Schwerpunkt von X eng mit der räumlichen Verteilung der Punkte im Raum verbunden sind: Die Region, die von den 3D-Punkten belegt ist, lässt sich durch die Eigenwerte entlang der Eigenvektoren annähern. Eine 3D-Fläche zeichnet sich also durch genau einen kleinen Eigenwert aus. Auf der Basis dieser Überlegung haben Nevado et al. einen Algorithmus entworfen, der Flächen aus 3D-Punktwolken heraus konstruiert [143].

Der folgende Abschnitt stellt einen neuen Algorithmus zur Extraktion von 3D-Flächen vor. Dabei werden zwei bereits bekannte Algorithmen kombiniert.

Kombinierter RANSAC-ICP Algorithmus zur 3D-Flächenextraktion

Ein kombinierter RANSAC-ICP Algorithmus bietet die Möglichkeit zur schnellen Bestimmung von 3D-Flächen in einer 3D-Punktmenge. Eine 3D-Fläche $\mathcal{E}$ ist durch drei 3D-Punkte

$$\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in \mathbb{R}^3 \tag{4.1}$$

oder äquivalent durch den 3D-Punkt $\mathbf{p}_1 \in \mathbb{R}^3$ und den Normalen- bzw. Lotvektor $\mathbf{n} \in \mathbb{R}^3$ ($||\mathbf{n}|| = 1$) gegeben. Letztere Form wird auch Hesse'sche Normalform genannt. Um eine 3D-Fläche in den Szenendaten zu detektieren, wählt der Algorithmus zunächst analog zum RANSAC-Verfahrens einen beliebigen Messpunkt aus und schätzt eine Ebene mittels zweier benachbarter Punkte. Anschließend berechnet er alle Punkte, die die Ungleichung

$$|(\mathbf{x} - \mathbf{p}_1) \cdot \mathbf{n}| < \varepsilon \tag{4.2}$$

erfüllen. Falls die Menge dieser Punkte ein bestimmtes Limit überschreitet (hier: 50 Punkte bei ε=2.5cm), wird eine ICP-basierte Optimierung gestartet. In jedem Iterationsschritt bilden alle Datenpunkte, die die Gleichung (4.2) erfüllen, die Menge M. Ihre Projektionen auf die 3D-Fläche bilden die Menge D als Eingabe für den Algorithmus. Die Minimierung der ICP-Fehlerfunktion geschieht durch eine Transformation der 3D-Fläche. Das Verfahren benötigt nur wenige Iterationsschritte, um die Fläche an den Daten auszurichten. Die zeitaufwändige Suche zur Bestimmung

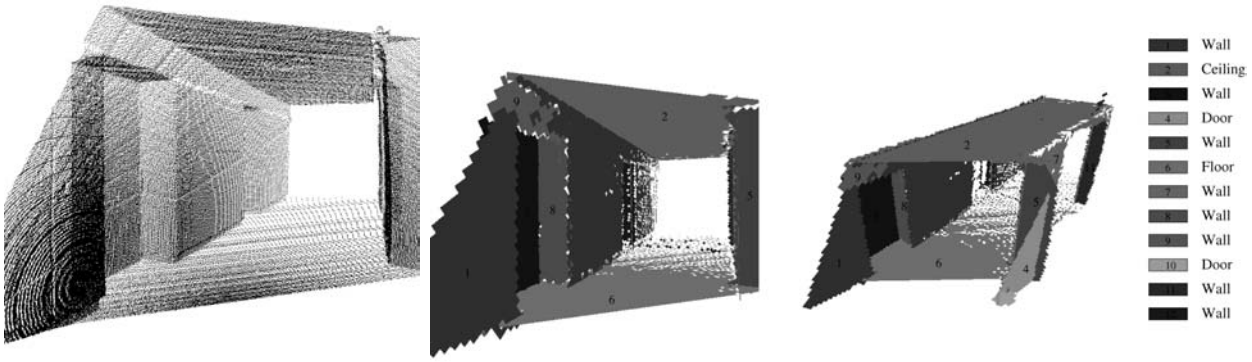


Abbildung 4.1: Flächenextraktion aus einer 3D-Punktwolke. Links: 3D-Punktwolke. Mitte und rechts: Extrahierte 3D-Flächen und ihre semantische Interpretation.

der nächsten Punkte $\mathbf{p}$ innerhalb des ICP-Algorithmus ist durch eine direkte Berechnung aus dem Anfragepunkt $\mathbf{p}_q$ ersetzt, d.h.

$$\mathbf{p} = \mathbf{p}_q - ||\mathbf{p} - \mathbf{p}_q|| \, \mathbf{n}. \tag{4.3}$$

Dabei heißt die Metrik *Punkte-Fläche Metrik*. Nachdem eine 3D-Fläche in die Daten eingepasst worden ist, werden alle Punkte, die zu der Fläche gehören, markiert. Der Algorithmus wird erneut mit den noch nicht markierten Punkten gestartet, um weitere 3D-Flächen zu finden. Das Verfahren terminiert, wenn alle Punkte nach Gleichung (4.2) getestet wurden:

1. Wähle einen 3D-Messpunkt $\mathbf{p}_0$ aus der Datenmenge X aus.
2. Bestimme die beiden nächsten Punkte $\mathbf{p}_1, \mathbf{p}_2$ und berechne die Ebene $\mathcal{E}$ durch $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$.
3. Starte ICP mit $M = \{\mathbf{p} | \mathbf{p} \in X(\mathcal{E}, \varepsilon)\}$ und $D = \text{Proj}(M, \mathcal{E})$.
4. Akzeptiere die Fläche, falls $|M|$ hinreichend groß und berechne $X := X \setminus M$.
5. Wiederhole die Schritte 1 bis 4, bis alle Punkte in X getestet wurden.

Die extrahierten 3D-Flächen $\mathcal{E}$ haben zunächst keine Größe, es handelt sich um Ebenen. Diese werden zu 3D-Flächen, indem zunächst die Messpunkte auf sie projiziert werden. Ein Quadtree generiert anschließend die 3D-Fläche [29] (vgl. Abbildung 4.1, Mitte und rechts).

In bestimmten Fällen ist es sinnvoll, begrenzte 3D-Flächen zu extrahieren. Die Begrenzung führt dazu, dass die 3 Punkte (vgl. (4.1)) die 3 Eckpunkte repräsentieren. In diesem Fall rechnet der

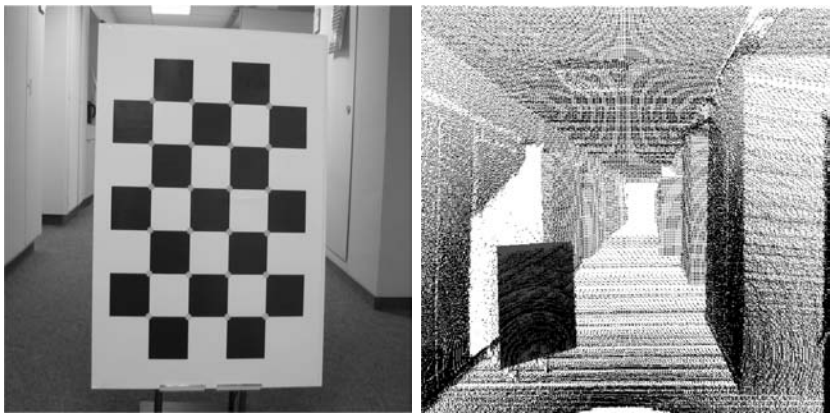


Abbildung 4.2: Extraktion eines Kamerakalibrierungsbretts aus einer 3D-Punktwolke. Links: Foto einer Szene, die eine Tafel mit einem Schachbrettmuster enthält. Rechts: Extrahierte 3D-Fläche.

Algorithmus statt mit Gleichung (4.3) wie folgt:

$$\mathbf{c} = \frac{1}{2}(\mathbf{p}_2 + \mathbf{p}_3) \tag{4.4a}$$

$$\mathbf{p} = \mathbf{c} + \Xi\left(\frac{1}{2}, (\mathbf{c}\cdot\mathbf{p}_q)/\,||\mathbf{p}_2 - \mathbf{p}_1||\right)(\mathbf{p}_2 - \mathbf{p}_1) + \Xi\left(\frac{1}{2}, (\mathbf{c}\cdot\mathbf{p}_q)/\,||\mathbf{p}_3 - \mathbf{p}_1||\right)(\mathbf{p}_3 - \mathbf{p}_1). \tag{4.4b}$$

Die Funktion $\Xi(x)$ gibt $-\frac{1}{2}$ oder $\frac{1}{2}$ zurück, falls der Punkt $\mathbf{x}$ außerhalb des Intervalls $[-0.5, \ldots, 0.5]$ liegt, d.h.

$$\Xi(x) = \begin{cases} -\frac{1}{2} & \text{, falls } x < -\frac{1}{2} \text{ ist,} \\ \frac{1}{2} & \text{, falls } x > \frac{1}{2} \text{ ist,} \\ x & \text{, sonst.} \end{cases}$$

Der erste Teil der Gleichung (4.4a) repräsentiert den Mittelpunkt der Fläche, der zweite (4.4b) den jeweiligen Anteil entlang der x- bzw. y-Achse der begrenzten Ebene. Abbildung 4.2 zeigt einen 3D-Scan, aus dem eine begrenzte Ebene extrahiert wurde.

4.1.2 Anwendung: 3D-Scanmatching für Abwasserkanäle

Die Inspektion von kommunalen Abwassersystemen ist eine potentielle Verwendung von autonomen mobilen Robotern [30, 170]. Die Selbstlokalisierung und die möglichst genaue Rekonstruktion des Kanalsystems stellen zwei grundlegende Probleme einer solchen Inspektion dar. Kann ein Zylinder in die 3D-Daten eingepasst werden, lassen sich sowohl Abweichungen von der Idealform bestimmen (vgl. Abbildung 4.4 (a) und (b)) als auch die Poseschätzungen des Roboters verbessern.

Ein Zylinder ist definiert durch zwei 3D-Punkte $\mathbf{a}, \mathbf{b}$ und den Radius r (vgl. Abbildung 4.3). Der nächste Punkt auf der Röhrenoberfläche $\mathbf{c}$ zu einem gegebenen Datenpunkt $\mathbf{x}$ ergibt sich als:

$$\mathbf{n} = \frac{\mathbf{a} - \mathbf{b}}{\|\mathbf{a} - \mathbf{b}\|}, \qquad s = \frac{(\mathbf{x} - \mathbf{a}) \cdot \mathbf{n}}{(\mathbf{n} \cdot \mathbf{n})} \tag{4.5}$$

$$\mathbf{c} = s \cdot \mathbf{n} + r \cdot \frac{\mathbf{x} - s \cdot \mathbf{n}}{\|\mathbf{x} - s \cdot \mathbf{n}\|}, \tag{4.6}$$

mit dem Einheitsvektor $\mathbf{n}$ zwischen $\mathbf{a}$ und $\mathbf{b}$ sowie der Projektion s von $\mathbf{x}$ auf $\mathbf{n}$.

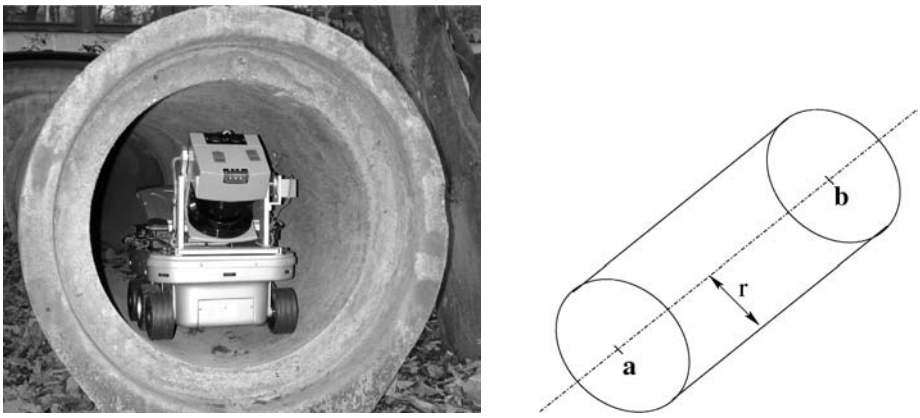


Abbildung 4.3: Links: Kurt3D im Testkanal auf dem Fraunhoferinstitutscampus in Birlinghoven. Rechts: Eine Röhre wird durch zwei Punkte $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ und einen Radius r beschrieben.

Die folgenden Schritte passen ein Röhrenmodell in eine gescannte Röhre ein [151]:

1. Füge eine initiale Röhre in den 3D-Scan ein, um den Matchingprozess zu initialisieren (vgl. Abbildung 4.4 (c)).
2. Berechne iterativ eine optimale Rotations- und Translationsmatrix mit dem Algorithmus der nächsten Punkte (vgl. Abbildung 4.4 (d) und (e)).
3. Vergrößere bzw. verkleinere sukzessive den Radius der Modellröhre bis die Röhre die 3D-Daten matcht, d.h., bis die Fehlerfunktion (3.1) ein Minimum erreicht (vgl. Abbildung 4.4 (f)).

4.1.3 Das Einpassen von 3D-Modellen

Analog zu den Beispielen für das Einpassen von Flächen und Röhren lassen sich beliebige Formen matchen. Dazu müssen sie lediglich als Drahtgittermodell vorliegen. Für jeden Datenpunkt in der Messdatenmenge und für jedes Teilstück des Gittermodells, z. B. jedes Dreieck, wird der nächste Punkt berechnet. Aus diesen Punkten wird der globale nächste Punkt bestimmt. Nachdem die Punktpaare gebildet wurden, minimiert der ICP-Algorithmus die Fehlerfunktion (3.1). Dadurch lassen sich beliebige Formen einpassen.

Ein Problem stellt die Auswahl derjenigen Punkte im 3D-Scan dar, die für das Matching benutzt werden. Für das Beispiel der Röhre wurden alle Scanpunkte verwendet, was für Anwendungen *im* Kanal Sinn macht. Für die Extraktion der Ebenen aus 3D-Punktwolken wurde das Problem umgangen, indem der Algorithmus zunächst randomisiert Flächen erzeugt und die folgenden Schritte mit ε-Umgebungen arbeiten.

Als Fazit für das Einpassen von beliebigen 3D-Formen in Punktwolken lässt sich festhalten, dass es im Prinzip möglich ist, mittels ICP-Algorithmus Modelle zu matchen. Leider ist vorab nicht bekannt, wo man anfangen soll. In der Praxis benötigen die Algorithmen also Zusatzinformationen in Form von Startschätzungen. Kapitel 4.3 zeigt eine Möglichkeit auf, wie eine solche Initalschätzung generiert werden kann.

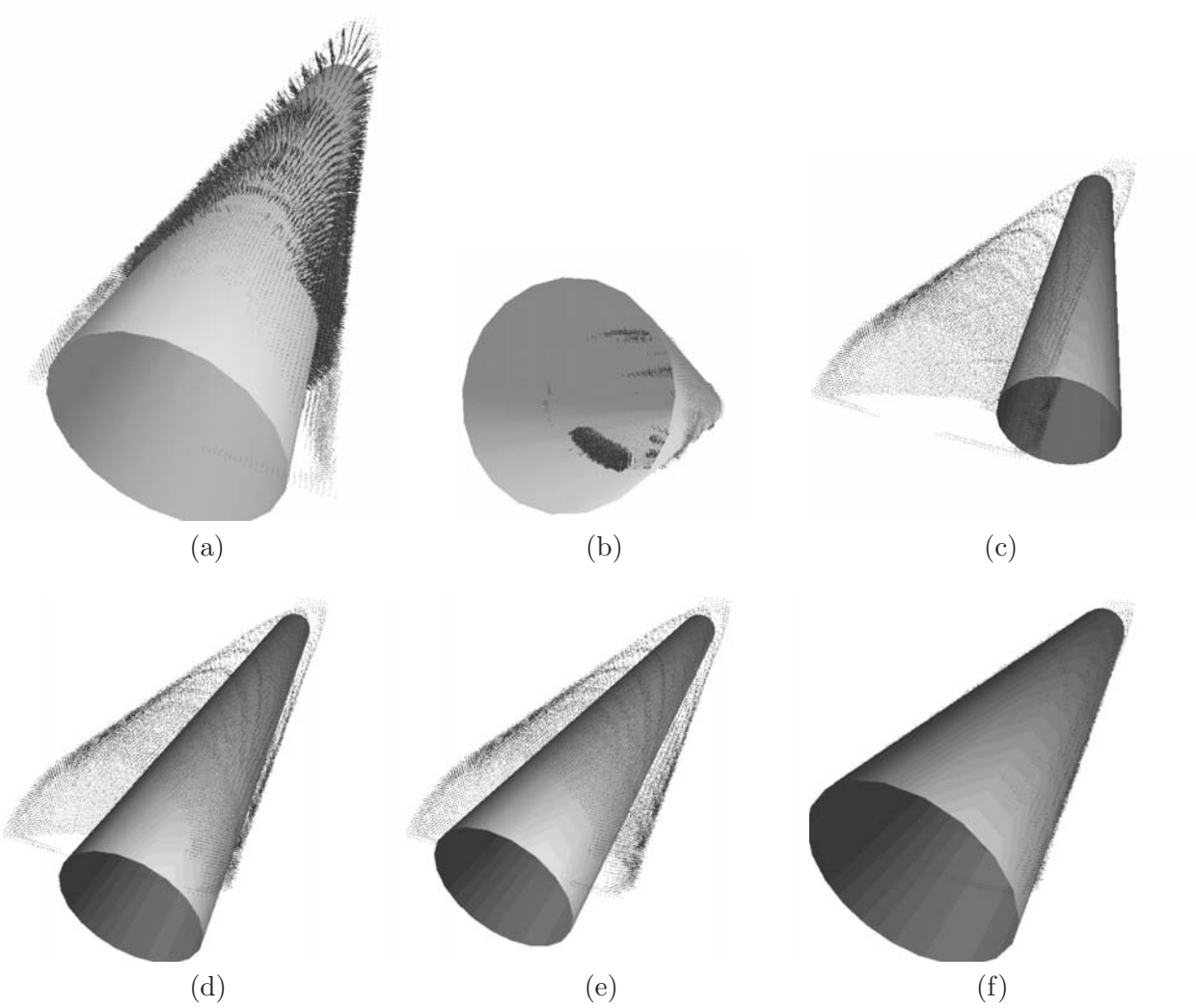


Abbildung 4.4: Scanmatching einer Röhre in 3D-Daten aus einem Kanalsystem. (a) Abweichungen von der Röhrengeometrie sind als rote Linien dargestellt. (b) Ein Hindernis innerhalb der Röhre lässt sich als Abweichung gut detektieren. (c) Anfängliche Lage des Röhrenmodells in den 3D-Daten. (d) und (e) Zwischenergebnisse des ICP-Algorithmus. (f) Vollständig eingepasstes Röhrenmodell.

4.2 Automatische Interpretation von 3D-Szenen

Viele Innenräume weisen ähnliche Formen auf, da sich die eingesetzten Baustoffe und Verfahren gleichen [76]. Dadurch treten architektonische Merkmale, beispielsweise planare Wände, Decken und Böden oder rechte Winkel in vielen Umgebungen auf. Den Hintergrund der Szeneninterpretation bildet also generisches Wissen aus der Architektur. Der im Folgenden vorgestellte Algorithmus verwendet das durch die Architektur vorhandene allgemeine Wissen, um spezielles Wissen über die 3D-Szene zu erzeugen.

Die automatische Interpretation von 3D-Szenen besteht aus zwei Schritten. Zuerst werden geeignete Merkmale, d.h. 3D-Flächen, aus den aufgenommenen Szenen extrahiert. Anschließend beschriftet

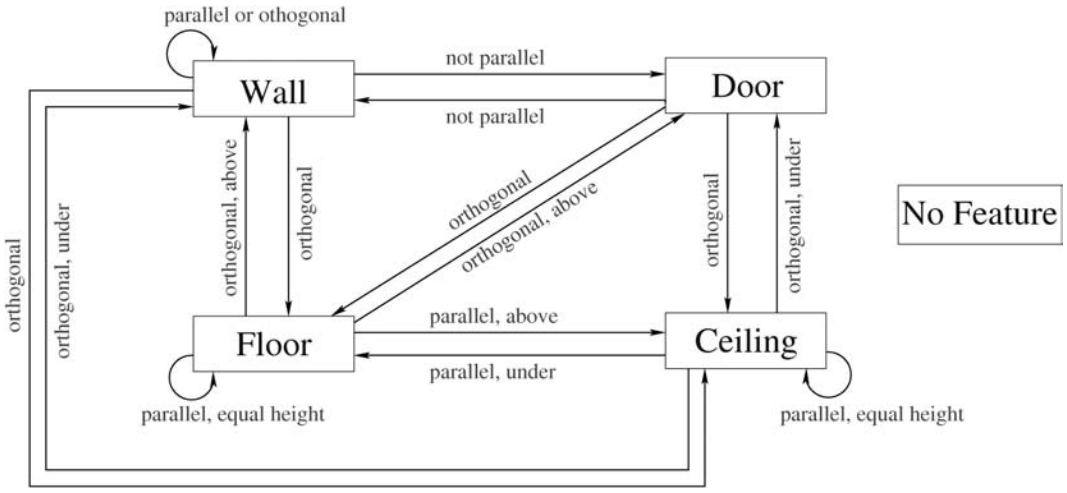


Abbildung 4.5: Das semantische Netz zur Szeneninterpretation

ein Backtracking-Algorithmus die Merkmale, so dass eine konsistente Interpretation entsteht. Aus allgemeinen Annahmen über den Aufbau von Innenräumen entsteht so spezielles Wissen über die erfasste Szene (vgl. [209]). Dieses Wissen wird schließlich zu Modellverbesserungen eingesetzt. Die Modellverbesserung gleicht Fehler aus, die während der Messung und der Scanregistrierung auftreten [150, 152]. Die folgenden Abschnitte beschreiben den Algorithmus im Detail.

Die Interpretation der Szene benutzt die 3D-Flächen, die mit dem Algorithmus aus dem vorigen Abschnitt gefunden wurden, als Merkmale. Die Menge der extrahierten 3D-Flächen sei mit $\mathcal{E}$ bezeichnet. Abbildung 4.1 zeigt einen 3D-Scan mit 184576 Messpunkten und 12 extrahierten 3D-Flächen. Ein allgemeines Modell von Innenräumen ist als ein semantisches Netz hinterlegt. Semantische Netze zur Interpretation von Bildszenen gehen auf Walz [209] zurück, und wurden erfolgreich von Grau et al. [97] sowie Cantzler et al. [46, 47] verwendet.

Ein semantisches Netz besteht aus Knoten und Kanten [173]. Die Knoten repräsentieren die verschiedenen Objektklassen der Welt bzw. des Modells. Die Kanten beschreiben die Beziehungen zwischen Objekten. Die Knoten haben folgende Bezeichnungen: $\mathcal{L} = \{$`Wall`, `Floor`, `Ceiling`, `Door`, `nofeature`$\}$, die mit den Beziehungen $\mathcal{R} = \{$`parallel`, `orthogonal`, `above`, `under`, `equalheight`$\}$ in Verbindung gebracht werden. Die Relationen `above` und `under` werden in Abhängigkeit zu ihrer Ebene verwendet, so dass sie nicht kommutativ sind. Abbildung 4.5 zeigt die Objektklassen und die Relationen. Zu beachten ist, dass der Knoten mit der Bezeichnung `door` ein *aufstehendes* Türblatt bezeichnet.

Das semantische Netz lässt sich leicht erweitern, wobei eine erweiterte Merkmalsextraktion notwendig wäre. Die hier vorgestellte Szeneninterpretation beschränkt sich auf einfache 3D-Flächen und lässt sich demzufolge in allen Innenräumen anwenden.

4.2.1 Tiefensuche zur Lösung des Zuordnungsproblems

Eine Tiefensuche (backtracking) implementiert die Zuweisung von Bezeichnungen aus der Menge $\mathcal{L}$ auf die 3D-Flächen $\mathcal{E}$. Diese Zuordnung muss so erfolgen, dass alle Relationen eingehalten werden.

Die Tiefensuche startet mit der Zuweisung der ersten Bezeichnung aus $\mathcal{L}$ auf die erste 3D-Fläche. Anschließend weist der Algorithmus der zweiten 3D-Fläche eine Bezeichnung zu und überprüft, ob alle durch die Relationen ausgeübten Zwänge erfüllt sind. Ist dies der Fall, wird mit der nächsten Fläche fortgefahren. Treten Inkonsistenten auf, startet das Backtracking mit der Zuweisung von anderen Bezeichnungen. Dieser Prozess ist beendet, wenn der gesamte Suchbaum abgearbeitet und alle konsistenten Belegungen gefunden sind. Die konsistenten Zuordnungen bezeichnen alle Flächen und erfüllen die Relationen im semantischen Netz. Aus diesen Zuweisungen wählt der Algorithmus schließlich diejenige aus, die

$$\sum_{p \in \mathcal{E}} f(p) \tag{4.7}$$

maximiert, wobei $f(p) = 0$, falls die 3D-Fläche p mit `nofeature` bezeichnet wurde und $f(p) = 1$, falls es sich bei p um eine `Wall, Door, Floor` oder `Ceiling` handelt. Die vollständige Suche und das Auswahlkriterium (4.7) stellen sicher, dass der Algorithmus eine korrekte Zuweisung mit einer minimalen Anzahl von `nofeature` Bezeichnungen erzeugt.

Die vollständige Suche mit Backtracking benötigt viel Rechenzeit. Um diese Zeit zu reduzieren, werden Teile des Suchbaums abgeschnitten, sobald festgestellt ist, dass die Belegung nicht zum Ziel führt. Des Weiteren werden Zwischenergebnisse gespeichert, um Berechnungen nicht doppelt durchzuführen. Beispielsweise benötigt die Verifikation der Relationen `under` und `above` die Auswertung der Distanzen aller Datenpunkte, die zu einer 3D-Fläche gehören.

Abbildung 4.1 zeigt die Szeneninterpretation der 3D-Flächen, die aus einer Punktwolke extrahiert wurden. Als Szene dient der so genannte GMD-Robobench, eine Büroumgebung zum Test von autonomen mobilen Robotern [193]. Die Ebene mit der Bezeichnung `door` stellt – auch wenn in der Abbildung nur schwer zu erkennen – eine leicht geöffnete Bürotür dar.

4.2.2 Ein Prolog-Programm zur Lösung des Zuordnungsproblems

Prolog bietet eine Möglichkeit, das semantische Netz zu externalisieren, d.h. außerhalb von compilierten Programmen zu repräsentieren. Dabei ist das Netz mit Hornklauseln kodiert [176]. Die Knoten des Netzes sind die Argumente der Relationen, die durch die Kanten definiert werden. Abbildung 4.6 zeigt das semantische Netz im Vergleich zu der Kodierung in Prolog. Hierbei sind alle Fakten für die Relation `parallel` angegeben. Um den Bezeichner `No Feature` auszudrücken, wird eine Bedingung verwendet, die verhindert, dass Prologs Unifikationsalgorithmus den 3D-Flächen den Bezeichner `nofeature` zuweist.

Zusätzlich zu dieser Repräsentation des semantischen Netzes stellt der Algorithmus basierend auf der Analyse der 3D-Flächen eine Klausel folgender Form zusammen. Dabei repräsentieren die Variablen `P0`, `P1`, ... die 3D-Flächen:

```
labeling(P0,P1,P2,P3,P4) :- parallel(P0,P1), under(P0,P1), orthogonal(P0,P2),
                            under(P0,P2),orthogonal(P0,P3), under(P0,P3),
                            parallel(P0,P4), above(P0,P4), ...
```

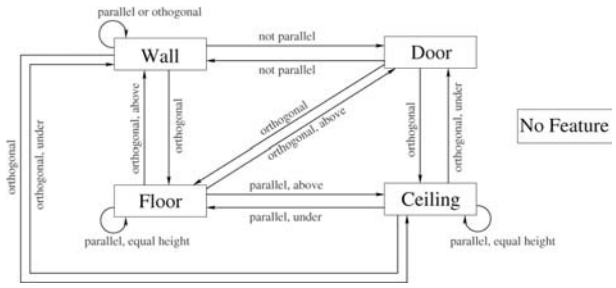
Der Unifikations- und Baktrackingalgorithmus von Prolog wird gestartet, indem die Interpretationssoftware die Klausel

```
consistent_labeling(P0,P1,P2,P3,P4) :- labeling(P0,P1,P2,P3,P4).
```

generiert. Die Anfrage

```
?- consistent_labeling(P0,P1,P2,P3,P4).
```

startet schließlich den Prozess.




```
parallel(floor,floor).
parallel(ceiling,floor).
parallel(ceiling,ceiling).
parallel(floor,ceiling).
parallel(wall,wall).
parallel(X,_) :- X == nofeature.
parallel(_,X) :- X == nofeature.
orthogonal(ceiling,door).
orthogonal(ceiling,wall).
...
equalheight(floor,floor).
equalheight(ceiling,ceiling).
equalheight(door,_).
...
```

Abbildung 4.6: Vergleich des semantischen Netzes mit der Prolog-Kodierung. Links: Semantisches Netz zur Szeneninterpretation. Rechts: Externalisierte Wissensrepräsentation in Prolog mit Fakten für jede Kante und einer Bedingung für die `No Feature` Bezeichnung.

Die Beschriftung einer 3D-Fläche mit `nofeature` wird nur dann in Erwägung gezogen, wenn die Unifikation fehlschlägt. In diesem Fall erzeugt das Programm eine weitere Hornklausel, um eine konsistente Belegung zu erreichen. Diese Hornklausel unifiziert explizit eine Variable mit der Bezeichnung `nofeature`. Die Berechnung aller Kombinationen ermöglicht es, alle Variablen zu berücksichtigen:

```
consistent_labeling(P0,P1,P2,P3,P4) :- comb([P0,P1,P2,P3,P4],[nofeature]),
                                       labeling(P0,P1,P2,P3,P4).
```

Der Prozess der expliziten Belegung von Variablen mit dem Bezeichner `nofeature` wird solange fortgesetzt, bis Prologs Unifikationsalgorithmus Erfolg hat:

```
consistent_labeling(P0,P1,P2,P3,P4) :- comb([P0,P1,P2,P3,P4],
                                            [nofeature,nofeature]),
                                       labeling(P0,P1,P2,P3,P4).
```

⋮

Die folgenden drei Klauseln berechnen alle Kombinationen:

```
comb(_,[]).
comb([X|T],[X|Comb]) :- comb(T,Comb).
comb([_|T],[X|Comb]) :- comb(T,[X|Comb]).
```

Bemerkung: Die Reihenfolge der Regeln innerhalb des Prolog-Programms ist für die korrekte Abarbeitung wichtig. Prolog arbeitet mit der SLD-Resolution (engl.: *linear resolution with selection function for definite clauses*), wobei die Auswahlfunktion durch die Reihenfolge der Klauseln gegeben ist. Lineare Resolution bedeutet, dass die Herleitung der leeren Klausel, also des Attributs *falsch*, als Binärbaum organisiert ist. Hierbei stellt die leere Klausel die Wurzel dar und es existiert nur ein Zweig, der durch Resolventenbildung entstanden ist. Eine definite Klausel steht gleichbedeutend für nicht-negative Hornklausel. [1]

[1] Prolog arbeitet mit der SLDNF-Resolution (engl.: *linear resolution with selection function for definite clauses with negation as failure*). Die dabei enthaltene Negation führt dazu, dass Nicht-Hornklauseln auftreten können. Damit ist diese Art der Resolution nicht mehr widerlegungsvollständig. Jedoch spielt eine Negation in dem erzeugten Prolog-Programm keine Rolle. Eine Einführung in die logische Programmierung und Prolog gibt Schöning [176].

Ein vollständiges Beispiel für ein aus einer 3D-Szene generiertes Prolog-Programm befindet sich in Anhang C.

4.2.3 Vergleich zwischen Tiefensuche und Prolog-basierter Beschriftung der 3D-Flächen

Tabelle 4.1 zeigt die Rechenzeit für die vollständige Tiefensuche und das Prolog-basierte Beschriftungsverfahren der 3D-Flächen in typischen Szenen, die mit dem 3D-Laserscanner aufgenommen wurden (Pentium IV-2400, SWI-Prolog [166]). Die Zeiten für die Prolog-Variante liegen unterhalb derer des Backtrackings. Während die Tiefensuche den gesamten Suchbaum erzeugt, um eine Maximierung nach Formel (4.7) zu berechnen, ist die durch das Prolog-Programm vorgegebene Tiefensuche optimistisch, d.h. es wird zuerst versucht, alle 3D-Flächen zu beschriften, *ohne* den Bezeichner `nofeature` zu verwenden. Dabei benutzt Prolog ebenfalls Tiefensuche. Wenn weitere Tiefensuchen notwendig sind, greift die verwendete Prolog-Implementation auf bereits erzeugte Zwischenergebnisse zurück [166].

Anzahl der Flächen	Backtracking	Prolog
5	93.51 ms	89.33 ms
7	155.14 ms	101.81 ms
13	589.11 ms	313.79 ms
25	3.7 sec	2.3 sec

Tabelle 4.1: Rechenzeit für das Matching des semantischen Netzes mit den 3D-Flächen.

4.2.4 Verbesserung des 3D-Modells

Bei jeder Messung treten Messfehler auf. Des Weiteren können auch beim Scanmatching Fehler auftreten (vgl. Abschnitt 3.1). Beides führt zu fehlerhaften 3D-Modellen. Nun hat aber die semantische Interpretation aus allgemeinem Wissen über Innenräume spezielles Wissen über die gescannte 3D-Szene erzeugt. Dieses wird im Folgenden dazu verwendet, das Modell zu verbessern. Dabei werden die extrahierten Ebenen so ausgerichtet, dass sie möglichst gut die 3D-Scandaten repräsentieren, gleichzeitig aber auch den semantischen Bedingungen, d.h. der Parallelität und Orthogonalität, gerecht werden.

Der erste Schritt der Modellverbesserung ist die Modellvereinfachung. Dieser Vorverarbeitungsschritt vereinigt benachbarte 3D-Flächen, wenn deren Beschriftungen gleich sind, z.B. bei zwei `ceiling` Flächen. Zunächst vergrößert sich der Abstand der 3D-Flächen zu den zu ihnen gehörenden Punkten. Dies spielt jedoch wegen der nachfolgenden Optimierung keine Rolle.

Der zweite Schritt ist die Verbesserung des 3D-Modells. Sie erfolgt durch das Aufstellen einer Fehlerfunktion und deren Optimierung. Die Fehlerfunktion besteht aus zwei Teilen: Der erste Teil summiert die Abstände aller Punkte zu ihren jeweiligen Flächen $\mathcal{E}$. Der zweite Teil besteht aus einer Summe der durch die Semantik induzierten Bedingungen Parallelität und Orthogonalität. Die Abweichungen der auftretenden Winkel werden aufsummiert. Die Fehlerfunktion lautet wie folgt:

$$E(\mathcal{P}) = \sum_{p_i \in \mathcal{E}} \sum_{\mathbf{x} \in X(\mathcal{E},\varepsilon)} ||(\mathbf{x} - \mathbf{p}_{i1}) \cdot \mathbf{n}_i|| + \gamma \sum_{p_i \in \mathcal{E}} \sum_{p_j \in \mathcal{E}} c_{i,j}, \tag{4.8}$$

mit $\mathbf{p}_{i1}$ einem Punkt der Ebene p_1 und mit Normalenvektor $\mathbf{n}_1$. $c_{i,j}$ beschreibt Parallelität (4.9) oder Orthogonalität (4.10). Die semantischen Bedingungen sind durch

$$c_{i,j} = \min\{|\arccos(\mathbf{n}_i \cdot \mathbf{n}_j)|, |\pi - \arccos(\mathbf{n}_i \cdot \mathbf{n}_j)|\} \tag{4.9}$$

bzw.

$$c_{i,j} = |\frac{\pi}{2} - \arccos(\mathbf{n}_i \cdot \mathbf{n}_j)| \tag{4.10}$$

gegeben. Die Fehlerfunktion kann sehr komplex werden, denn es gehen sämtliche Datenpunkte, 3D-Flächen und semantische Bedingungen in sie ein. Offensichtlich ist die Minimierung von (4.8) ein nicht-linearer Optimierungsprozess.

Die Zeit für mehrdimensionale Optimierungen steigt mit der Anzahl der Parameter n. Um den Prozess der Optimierung zu beschleunigen, werden die Ebenen kompakt repräsentiert (vgl. Formel 4.1). Der Normalenvektor $\mathbf{n}$ der 3D-Flächen wird in Kugelkoordinaten angegeben, d.h., durch zwei Winkel α und β. Außerdem reduziert man den Punkt $\mathbf{p}_1$ der Flächenbeschreibung auf einen konstanten Vektor in Richtung $\mathbf{p}_1$ und auf einen Abstand d zum Ursprung des Koordinatensystems. Demzufolge besteht die minimale Beschreibung einer 3D-Fläche aus einem 3-Tupel $p = (\alpha, \beta, d)$. Die Beschreibung der Menge aller 3D-Flächen $\mathcal{P}$ liegt nun in einem mehrdimensionalen Vektor vor, der durch das Zusammenfügen aller 3-Tupel entsteht.

Optimierungsverfahren für die Fehlerfunktion (4.8) können ausschließlich Funktionswerte berücksichtigen, da keine Ableitungen zur Verfügung stehen. Cantzler et al. benutzen genetische Algorithmen für die Optimierung [47]. Genetische Algorithmen ermitteln ein Minimum, indem sie für die Variablen eine Menge (Pool) von Lösungen (Genen) konstruieren. Anschließend werden die Gene randomisiert verändert, gekreuzt und vermehrt. Die zu optimierende Funktion übernimmt die Rolle einer Fitnessfunktion, die eine Auswahl unter den Genen, d.h. Variablenbelegungen, trifft. Die Minimierung von Funktionen mit genetischen Algorithmen ist sehr rechenaufwändig, bietet aber die Möglichkeit, ein globales Minimum zu finden.

Da die Fehlerfunktion (4.8) aus einer gescannten Szene mit definierter Genauigkeit hervorgeht, kann angenommen werden, dass der Startwert nahe dem globalen Minimum liegt. Daher wurden lokale Minimierungsverfahren zur Optimierung gewählt. Zwei weitere Optimierungsverfahren eignen sich: Eine Heuristik basierend auf Powells Minimierungsmethode (vgl. Anhang B.2) und die Downhill-Simplex Methode (vgl. Anhang B.3). Powells Methode versucht konjugierte Richtungen zu berechnen, entlang derer sie die Funktionswerte auswertet, bis ein lokales Minimum gefunden ist. Die Downhill-Simplex Methode benutzt $n + 1$ Werte der zu optimierenden n-dimensionalen Funktion, die sich aus dem Startwert generieren lassen. Anschließend wird das von den Werten aufgespannte Intervall schrittweise verkleinert, bis ein lokales Minimum erreicht ist.

Abbildung 4.7 zeigt den Verlauf der Optimierung mit beiden Verfahren. Die Downhill Funktion findet bei gleicher Anzahl von Funktionsauswertungen ein besseres Minimum, ist allerdings am Anfang des Optimierungsprozesses schlechter als Powells Methode. Die Ausschläge während der Minimierung mit dem Verfahren nach Powell haben ihre Ursache in sehr schlechten Suchrichtungen.

Der letzte Schritt für die Modellverbesserung ist die Koordinatensystemausrichtung der gescannten Szene auf der Grundlage der gefundenen Interpretation. Während des Scanmatchings definiert der erste Scan das globale Koordinatensystem. Dieses lässt sich nun so ausrichten, dass Fußböden/-Decken und Wände orthogonal und parallel zu den Achsen des Koordinatensystems sind.

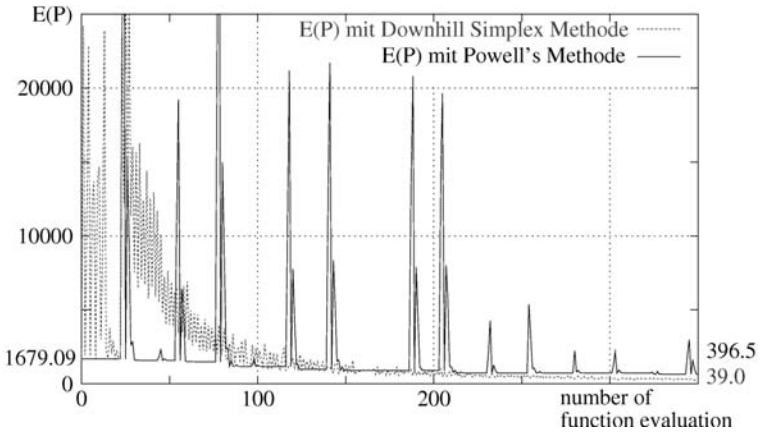


Abbildung 4.7: Vergleich zwischen der Minimierung mit der Downhill Simplex Methode und jener mit Powells Methode.

Die vorgestellten Verfahren wurden in vielen Experimenten im GMD-Robobench getestet. Abbildung 4.1 zeigt eine Punktwolke eines 3D-Scans mit 184576 Punkten, die extrahierten 3D-Flächen und deren Interpretation. Das Originalmodell und das verbesserte Modell sind in Abbildung 4.8 gegeben. Dabei wurde der Wert der Fehlerfunktion $E(P) = 11.76 + \gamma 16.68$ reduziert auf $E(P) = 15.61 + \gamma 0.23$. Der Parameter γ hat hierbei den Wert 100. Die Abbildung zeigt die Reduktion der Messfehler auf dem Boden und an der Decke. Um die Effekte zu visualisieren, wurde die Darstellung durch Octalbäume gewählt (vgl. Seite 47).

Abbildung 4.9 illustriert ein weiteres Beispiel für die Interpretation von 3D-Szenen und die Modellverbesserung. Hierbei wurden 8 3D-Scans mit dem autonomen Roboter Ariadne aufgenommen und zu einer konsistenten Szene zusammengefügt. Die Interpretation benutzt nur die flachen Objekte, sie filtert also gleichzeitig die Personen aus den Daten. Die Abbildung zeigt ein Foto der Szene, die mit Reflektionswerten gerenderte Octalbaumdarstellung, die semantische Interpretation sowie das Orginal und das verbesserte Modell.

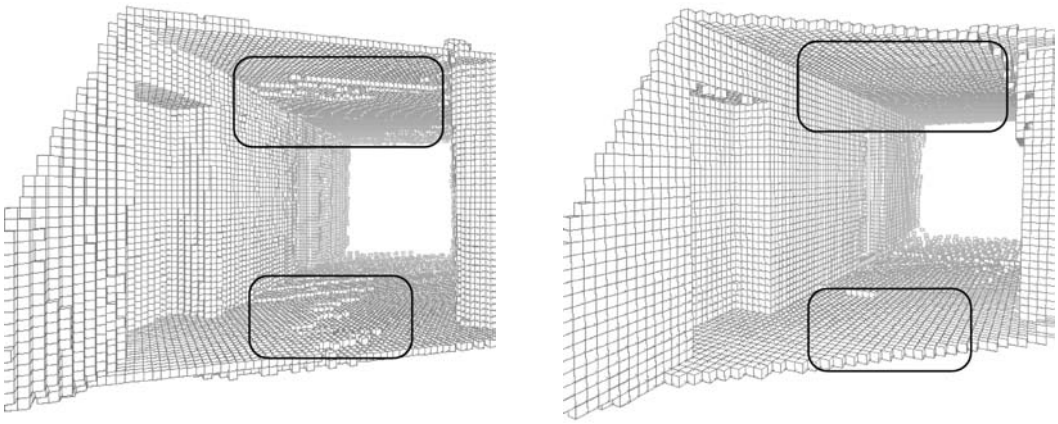


Abbildung 4.8: Modellverbesserung. Links: Orginalmodell. Rechts: Verbessertes Modell. Über weite Teile der Szene entsteht ein besseres Voxelmodell, allerdings treten auch lokale negative Effekte auf, deren Ursache in der Voxelbildung [144] liegen.

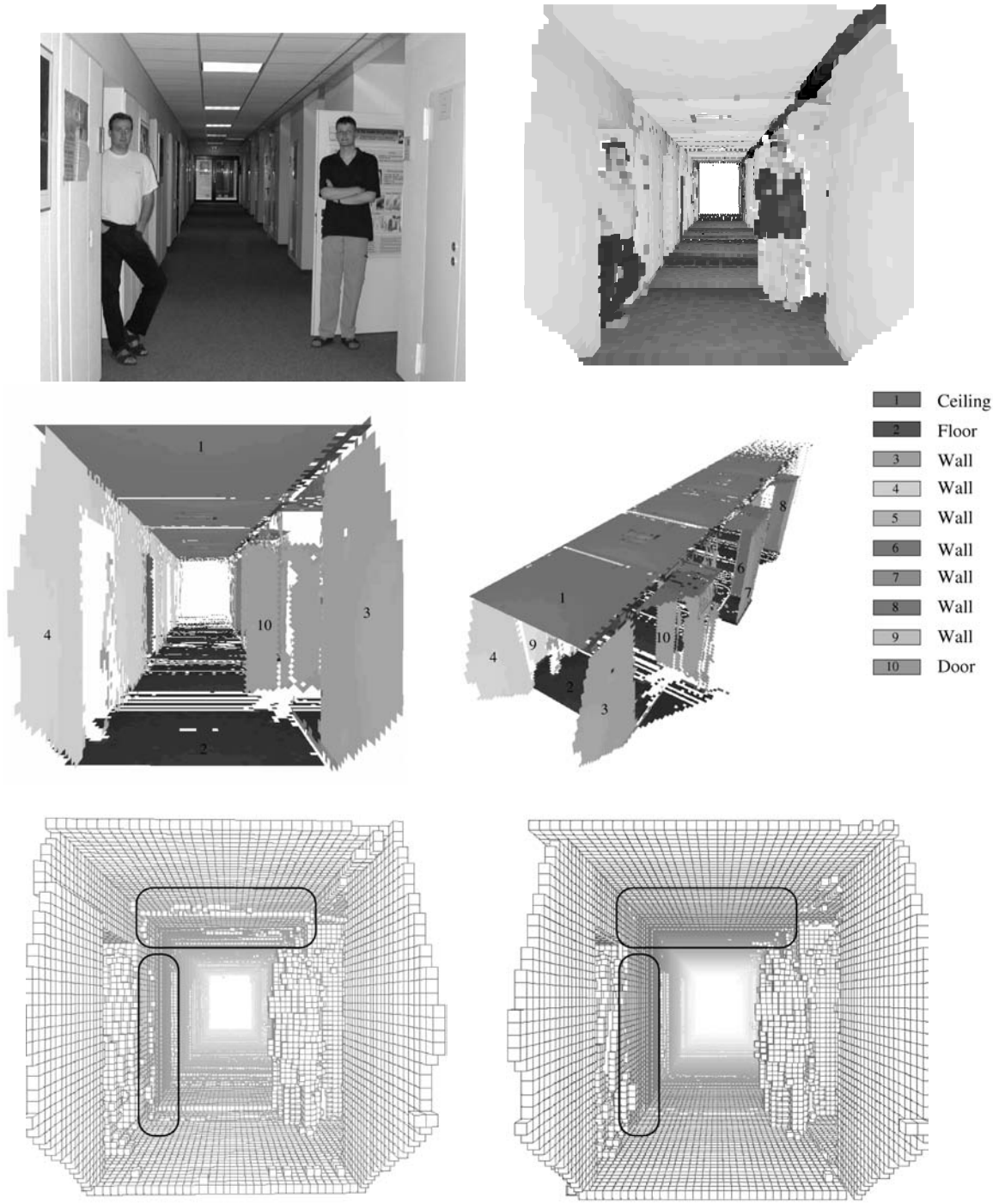


Abbildung 4.9: Durch einen autonomen Roboter rekonstruiertes Flächenmodell (8 vereinigte 3D-Scans). Die Personen in der Szene werden durch die 3D-Flächenerkennung herausgefiltert. Oben links: Foto der Szene. Oben rechts: Gerenderte Szene in Octalbaumdarstellung. Mitte: Extrahierte Flächen und semantische Interpretation. Unten links: Orginalmodell. Unten rechts: Verbessertes Modell.

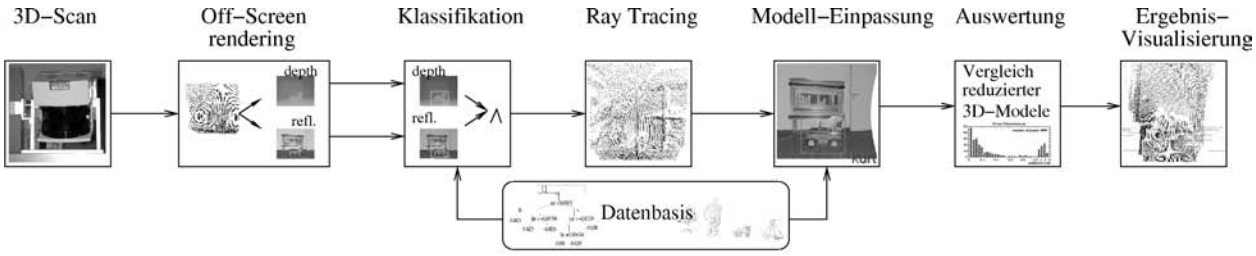


Abbildung 4.10: Das System zur automatischen Objektklassifikation und Objektlokalisierung.

4.3 Objektklassifikation und -lokalisierung in 3D-Scans

Bei der automatischen Objektklassifikation und -lokalisierung werden die Objekte nicht wie in den vorangegangen Abschnitten explizit repräsentiert, sondern von Algorithmen gelernt. Lernen bedeutet immer das Lernen von Parametern, so dass auf einem Lerndatensatz möglichst wenig Fehler gemacht werden. Des Weiteren sollen die gelernten Sachverhalte gut verallgemeinern, d.h. auch unbekannte Daten gut vorhersagen bzw. klassifizieren. Beim Lernen kommen N positive und negative Beispiele zum Einsatz. Diese werden als Paare $(x_1, y_1), \ldots, (x_N, y_N)$ angegeben, wobei $y_i \in \{-1, 1\}, i \in \{1, \ldots, N\}$ die klassifizierte Ausgabe ist.

Bevor jedoch die Lernalgorithmen vorgestellt werden, müssen die 3D-Scandaten vorverarbeitet werden. Um 3D-Objekte automatisch zu erkennen und zu lokalisieren, sind etliche Teilschritte notwendig: Bilderzeugung, Klassifikation, Ray Tracing, Modelleinpassung und Auswertung. Abbildung 4.10 gibt einen Überblick über das System.

4.3.1 Erzeugen von Bildern aus den 3D-Scandaten

Nach dem Scannen liegen die Daten als Menge von 3D-Punkten vor. Jeder 3D-Scanpunkt besitzt zusätzlich ein Attribut $c \in [0, 1]$, das die Reflektivität angibt. Die 3D-Daten können nun mittels `OpenGL`, auch ohne die Grafikkarte zu verwenden, auf eine Bildfläche projiziert werden. Befindet sich dabei die virtuelle Kamera genau an dem Ort, von dem die Laserstrahlen ausgesandt wurden, entsteht ein Bild, in dem die projizierten Punkte gleichförmig verteilt sind. Die Ursache hierfür liegt in der kugelförmigen (radialen) Abstrahlung des Lasers. Vergrößert man nun sukzessive die 3D-Punkte, bis sich die Zwischenräume auflösen, entsteht ein Bild vom Scan. Die Bildpunkte werden entweder mit Grauwerten in Abhängigkeit von der Tiefe oder von den Reflektionswerten dargestellt. Abbildung 4.11 zeigt einen 3D-Scan und zwei erzeugte Bilder. Da die verwendete virtuelle Kamera einen Öffnungswinkel von 60° besitzt, geben die generierten Bilder nur einen Ausschnitt des 3D-Scans wieder. Die gerenderten Bilder entstehen durch eine ideale Projektion, sind also nicht verzerrt.

4.3.2 Merkmalsdetektion mit Integralbildern

Für die Entscheidung, statt einzelner Bildpunkte Merkmale zu verwenden, gibt es viele Gründe im Informatikbereich „Computer-Sehen“. Bei Anwendungen auf mobilen Robotern ist die Rechenzeit mit den begrenzten Kapazitäten eines solchen Systems entscheidend: Diese ist bei merkmalbasierten Klassifikationssystemen oftmals besser [205]. Die hier verwendeten Merkmale haben die gleiche

Abbildung 4.11: Links: 3D-Scan als Punktwolke. Mitte: 3D-Szene als Tiefenbild, wobei dunkle Grautöne größeren Entfernungen entsprechen. Rechts: Reflektionsbild der Szene.

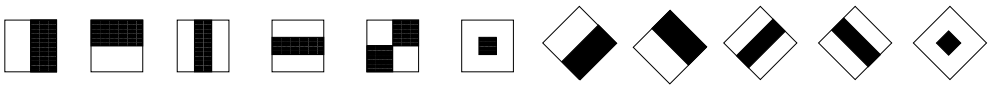


Abbildung 4.12: Merkmale zur Objekterkennung. Kanten-, Linien-, Diagonal-, und center-surround Merkmale. Die invertierten Merkmale können durch Einstellung des Schwellwertes in den Merkmalen entstehen.

Struktur wie die Haar-Funktionen [101] mit denen Wavelets dargestellt werden können: $f_{\text{Haar}} : \mathbb{R} \rightarrow [-1, 1]$, mit

$$f(x) = \begin{cases} -1 & 0 \leq x \leq 1/2 \\ 1 & 1/2 \leq x \leq 1 \\ 0 & \text{, sonst.} \end{cases}$$

Diese Haar-Funktionen sind Schrittfunktionen und werden auch in [126, 157, 205] verwendet.

Abbildung 4.12 zeigt die 11 Basisfunktionen, d.h. die Kanten-, Linien-, Diagonal-, und center surround Merkmale. In einem Basisdetektor, der z.B. die Größe von 30×30 Pixel besitzt, werden alle möglichen Merkmale generiert. Für einen 30×30 Detektor ergeben sich 642592 Merkmale (vgl. [126] für Details zur Anzahlberechnung).

Die ersten 6 Merkmale lassen sich effizient auswerten, indem Integralbilder verwendet werden [126,204,205] (engl.: *integral images* oder *summed area tables*). Ein Integralbild I ist eine Zwischenrepräsentation für das Bild mit einer Breite x und Höhe y und enthält die Summe der Pixelwerte N:

$$I(x, y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} N(x', y').$$

Das Integralbild wird rekursiv durch folgende Formel bestimmt:

$$I(x, y) = I(x, y-1) + I(x-1, y) + N(x, y) - I(x-1, y-1)$$

mit $I(-1, y) = I(x, -1) = I(-1, -1) = 0$. Demnach benötigt die Berechnung von I nur einen einzigen Zugriff auf die Eingabedaten. Das Zwischenergebnis Integralbild erlaubt die Berechnung

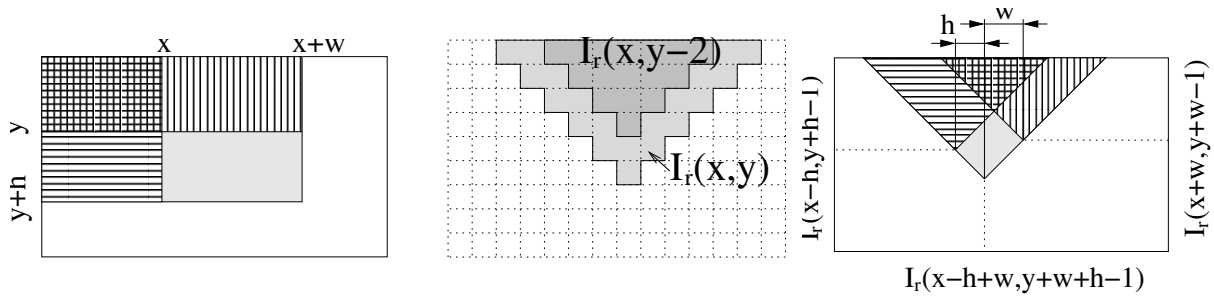


Abbildung 4.13: Links: Die Berechnung der Merkmalswerte F in der schattierten Region basiert auf den Integralwerten der 4 oberen Rechtecke. Mitte: Rotierte Integralbilder I_r. Rechts: 4 Referenzen werden benötigt, um ein rotiertes Merkmal F_r zu errechnen.

einzelner Rechteckmerkmale der Breite (h, w) an Pixel (x, y) durch vier Referenzen auf das Integralbild (vgl. Abbildung 4.13):

$$F(x, y, h, w) \quad = \quad I(x, y) + I(x + w, y + h) - I(x, y + h) - I(x + w, y).$$

Für die Bestimmung der rotierten Merkmale haben Lienhart et al. die rotierten Integralbilder eingeführt [126]. Diese enthalten die Summen über die Pixel eines Rechtecks, das um 45° gedreht wurde. Die untere Spitze befindet sich bei (x, y) (vgl. Abbildung 4.13):

$$I_r(x, y) = \sum_{x'=0}^{x} \sum_{y'=0}^{x-|x'-y|} N(x', y').$$

Auch die rotierten Integralbilder I_r werden rekursiv berechnet, und zwar durch

$$I_r(x, y) = I_r(x - 1, y - 1) + I_r(x + 1, y - 1) + -I_r(x, y - 1) + N(x, y) + N(x, y - 1),$$

mit den Startwerten $I_r(-1, y) = I_r(x, -1) = I_r(x, -2) = I_r(-1, -1) = I_r(-1, -2) = 0$. Wiederum benötigt man 4 Referenzen auf das Integralbild, um ein Rechteck auswerten zu können:

$$\begin{aligned} F_r(x, y, h, w) \quad = \quad & I_r(x + w - h, y + w + h - 1) + \\ & I_r(x, y - 1) - I_r(x - h, y + h - 1) - \\ & I_r(x + w, y + w - 1). \end{aligned}$$

Da alle Merkmale aus Rechtecken zusammengesetzt sind, lassen sie sich durch das Referenzieren des Integralbildes und gewichtete Subtraktionen bestimmen. Die Gewichtung geschieht dabei proportional zu den Flächen der weißen und schwarzen Bereiche.

Um ein Merkmal f_i zu detektieren, benötigt man einen Schwellwert. Ist die Auswertung eines Merkmals über einen Bildbereich größer als der Schwellwert, gilt das Merkmal als erkannt. Der Schwellwert wird so angepasst, dass möglichst wenige der positiven bzw. negativen Beispiele falsch klassifiziert werden. Tritt ein negativer Schwellwert auf, handelt es sich um ein inverses bzw. „negatives“ Merkmal. Des Weiteren stellt der Algorithmus die Rückgabewerte (α, β) so ein, dass der Fehler über den Beispielen am geringsten ist. Im diskreten Fall sind nur die Rückgabewerte -1 und 1 erlaubt, ansonsten Werte aus dem Bereich $[-1, 1]$.

4.3.3 Das Lernen von Klassifikatoren

Für das Lernen von Klassifikatoren bieten sich so genannte Boosting-Techniken an. Sie zielen darauf ab, eine möglichst kleine Teilmenge von Merkmalen bzw. einfachen Klassifikatoren, die nur ein Merkmal benutzen, auszuwählen. Man erhofft sich davon, dass die gewichtete Summe gute Klassifikationsergebnisse erzielt.

Für die Auswahl der Merkmale stehen verschiedene Ausprägungen des Boosting-Verfahrens zur Verfügung: Der Diskrete Ada Boost-, der Real Ada Boost- und der Gentle Ada Boost- Algorithmus [82]. Alle Ausprägungen verbindet die Gemeinsamkeit, dass zuerst ein Merkmal ausgewählt wird und anschließend die Trainingsbeispiele neu gewichtet werden. Danach startet der Auswahlprozess erneut, bis eine gegebene Klassifikations- und Fehlerrate erreicht ist. Die Ausprägungen des Lernverfahrens unterscheiden sich in der Art und Weise der Neuverteilung der Gewichte.

Im Folgenden wird eine Klassifikation als Treffer (engl.: *hit*) und ein Fehler als falscher Alarm (engl.: *false alarm*) bezeichnet. Das Lernen basiert auf N gewichteten Trainingsbeispielen $(x_1, y_1), \ldots, (x_N, y_N)$, wobei x_i die Bilder sind und $y_i \in \{-1, 1\}_{i=1,\ldots,N}$ die klassifizierte Ausgabe ist. Zu Beginn der Lernphase werden alle Gewichte konstant initialisiert, d.h. $w_i = 1/N$. Der diskrete Ada Boost-Algorithmus führt nun die folgenden Schritte aus:

1. Setze $m := 0$.
2. Erhöhe den Zähler m, d.h. $m := m + 1$.
   (a) Passe den Schwellwert in den Merkmalen f_j so an, dass eine minimale Anzahl von Trainingsbeispielen $(x_1, y_1), \ldots, (x_N, y_N)$ als falsch klassifiziert wird. Dabei sind die Beispiele mit w_i gewichtet.
   (b) Für jedes Merkmal f_j berechne
   $$e_j = \sum_{i=1}^{N} w_i \begin{pmatrix} 1 & ,\text{falls } (y_i \neq f_j(x_i)) \\ 0 & ,\text{sonst.} \end{pmatrix} \quad \text{und} \quad c_j = \frac{1}{2} \log \left(\frac{1 - e_j}{e_j} \right).$$
   (c) Wähle das Merkmal f_j, das den geringsten Fehler e_j aufweist, als Merkmal f'_m aus.
3. Aktualisiere alle Gewichte mit $w_i := w_i e^{c_j - 1_{(y_i \neq f'_m(x_i))}}$ unter Benutzung des gewählten Merkmals. Damit werden falsch klassifizierte Beispiele für den nächsten Schleifendurchlauf stärker gewichtet. Renormalisiere anschließend die Gewichte, so dass $\sum_i w_i = 1$ gilt.
4. Teste, ob der Klassifikator $\text{sign}\left(\sum_{t=1}^{m} c_m f'_m(x)\right)$ die gewünschte Trefferquote und Rate falscher Alarme bereits erfüllt. Wenn nicht, mache mit Schritt 2 weiter.

In den reellen Varianten des Algorithmus, Real Ada Boost und Gentle Ada Boost, dürfen die Merkmale (Merkmalsklassifikatoren) Rückgabewerte im Intervall $[-1, 1]$ annehmen. Für den Real Ada Boost-Algorithmus werden die Schritte 2b und 3 wie folgt angepasst:

2. (b') Für jedes Merkmal f_j berechne

$$e_j = \sum_{i=1}^{N} (w_i y_i f_j(x_i)) \quad \text{und} \quad c_j = \frac{1}{2} \log \left(\frac{1 - e_j}{e_j} \right).$$

3. Aktualisiere alle Gewichte mit $w_i := w_i e^{c_j - y_i f'_m(x_i)}$ unter Benutzung des gewählten Merkmals. Renormalisiere anschließend die Gewichte.

Der Gentle Ada Boost Algorithmus kommt ohne Schritt 2b aus, da der Log-Fehler c_j für die Merkmale nicht berechnet werden muss. Die Aktualisierung der Gewichte basiert nur auf dem auftretenden Fehler, d.h.

2. (b') entfällt.

3. Aktualisiere alle Gewichte mit $w_i := w_i e^{-y_i f'_m(x_i)}$ unter Benutzung des gewählten Merkmals. Anschließend Renormalisiere anschließend die Gewichte.

Die Ausgabe des Klassifikators über einer Bildregion x ist

$$\text{sign}(\sum_{m=1}^{M} f'_m(x)),$$

mit $f'_m(x) = \alpha$, falls $x \geq \text{thr.}$, und $f'_m(x) = \beta$ sonst. α und β sind die gewichteten Ausgaben der einfachen Merkmale.

Eine Kaskade von Klassifikatoren

Die Performanz eines Klassifikators reicht für die Klassifikation von Objekten nicht aus. Trotz hoher Trefferquote, z.B. $p_{\text{hit}} = 0.999$, ist auch die Wahrscheinlichkeit der falschen Alarme hoch, z.B. $p_{\text{falarm}} = 0.5$. Da die Trefferquote nahe 1 liegt und signifikant größer als die Fehlerrate ist, bietet sich die Konstruktion einer Kaskade von Klassifikatoren an. Dabei werden mehrere gelernte Klassifikatoren hintereinander angeordnet, so dass ein degenerierter Entscheidungsbaum entsteht. In jeder Etappe fällt der Algorithmus die Entscheidung, ob der entsprechende Bildauschnitt x das Objekt enthält. Nur wenn über alle Etappen eine positive Entscheidung getroffen wird, gilt das Bild als positiv klassifiziert und das Objekt als erkannt. Durch dieses Verfahren werden sowohl die Trefferquote p_{hit} als auch die Wahrscheinlichkeit der falschen Alarme $p_{\text{falarm}} = 0.5$ reduziert. Bei einer Kaskade mit k Schritten gilt: $p_{g,\text{hit}} = p_{\text{hit}}^k$ und $p_{g,\text{falarm}} = p_{\text{falarm}}^k$. Allerdings wird die Fehlerwahrscheinlichkeit viel schneller reduziert als diejenige der Treffer. Somit ist eine sichere Klassifikation möglich. Des Weiteren sinkt durch den Einsatz der Kaskade die Rechenzeit des Klassifikationsprozesses, da viele Bereiche, die nicht das gesuchte Objekt enthalten, schnell abgelehnt werden können.

Die Kaskade wird durch einen einfachen iterativen Prozess konstruiert. Für jede Etappe lernt der Algorithmus eine Ansammlung von Merkmalen, bis die Trefferquote und Fehlerrate erreicht ist.

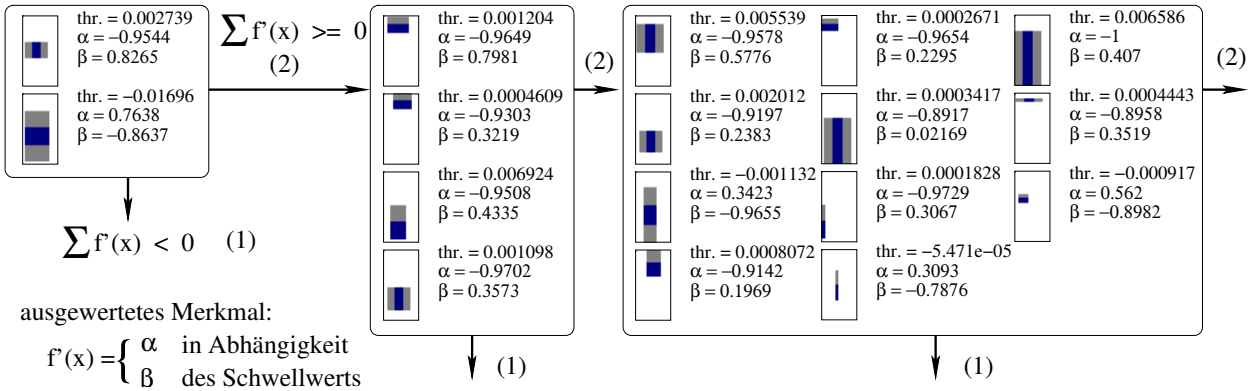


Abbildung 4.14: Die ersten drei Etappen einer Kaskade von Klassifikatoren zur Detektion eines Bürostuhls in Tiefenbildern. Jede Etappe enthält eine Ansammlung einfacher Haar-Merkmale, die wiederum mit dem Gentle Ada Boost Algorithmus gelernt wurden. Jedes Merkmal besitzt einen Schwellwert (thr.) und die Rückgabewerte α und β. In jeder Etappe wird die Summe $\sum f'(x)$ gebildet, anhand derer dann über die Fortsetzung entschieden wird.

Dieser Prozess wird mit den positiven Beispielen fortgesetzt, die die Kaskade korrekt passiert haben. Aus den negativen Bildbeispielen werden Ausschnitte gesucht, die ebenfalls die aktuelle Kaskade passieren konnten. Diese Bildausschnitte wurden dann falsch klassifiziert und werden für die nächste Etappe als Trainingsbilder benutzt. Dieser Erzeugungsprozess für die negativen Beispiele heißt *bootstrapping* und ist der zeitaufwändigste Teil des Lernverfahrens.

Beim Lernen der Kaskade erhöht sich in höheren Stufen für gewöhnlich die Anzahl der einfachen Merkmale im Klassifikator, da es immer schwerer wird, die gewünschte Trefferquote und die Rate falscher Alarme zu erreichen. Abbildung 4.14 zeigt die ersten drei Etappen der Kaskade für das Objekt Bürostuhl.

Die Anwendung der Kaskade

Um ein Objekt in einem Bild zu erkennen, startet der Detektionsprozeß mit der kleinsten Größe des Klassifikators und sucht das gesamte Bild von oben links nach unten rechts durch Anwendungen der Kaskade nach dem Objekt ab. Sollen Objekte unterschiedlicher Größe erkannt werden, wird der Klassifikator neu skaliert. Die Darstellung des Bildes als Integralbild bleibt davon unberührt und aufwändiges Neu-Skalieren des Bildes wird vermieden. Der große Vorteil der Haar-Merkmale ist, dass sie einfach skalierbar sind. Dadurch ist die Objekterkennung größen-invariant.

Eine Kombination der Kaskaden für die Tiefenbilder und Reflektionsbilder reduziert die Anzahl der falschen Alarme. Es gibt zwei Kombinationsmöglichkeiten: Die beiden Kaskanden werden abwechselnd oder nacheinander ausgewertet (vgl. Abbildung 4.15). In beiden Möglichkeiten entspricht die erfolgreiche Objekterkennung einer UND-Verknüpfung. Um zu vermeiden, dass sich auch die Trefferquote verringert, werden aus verschiedenen Blickwinkeln und mit unterschiedlichen Öffnungswinkeln Bilder aus dem 3D-Scan generiert [149].

Die Geschwindigkeit der kombinierten Kaskade ist höher, wenn die Kaskaden nacheinander ausgewertet werden. Im seriellen Fall werden also weniger Merkmale evaluiert, was darauf hindeutet, dass viele Bildausschnitte schneller abgelehnt werden können.

Abbildung 4.15: Die Vereinigung der Kaskaden. Links: Abwechselnde Kaskadenschritte. Rechts: Serialisierte Kaskaden.

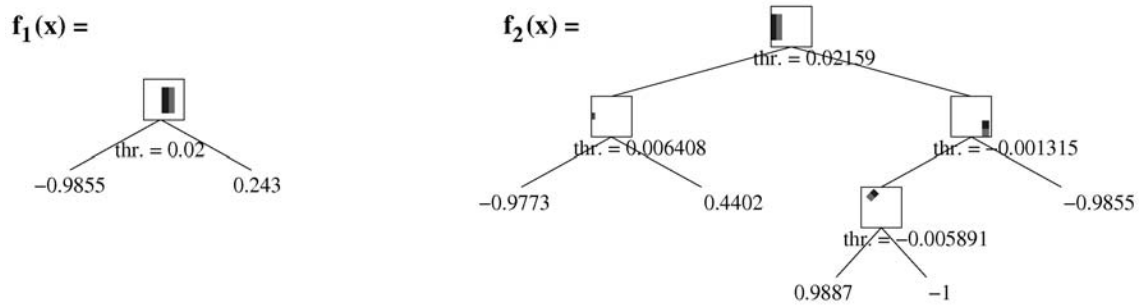


Abbildung 4.16: CARTs zur Auswertung von Bildregionen. Links: Ein einfaches Merkmal f_1 zur Klassifikation. Es besitzt einen Schwellwert und die Rückgabewerte α und β. Rechts: Ein Klassifikations- und Regressionsbaum (CART) f_2 mit 4 Knoten. In Abhängigkeit von den Merkmalen und den Schwellwerten wird für eine Bildregion x über den Weg durch den Baum entschieden. Die Blätter enthalten die Rückgabewerte.

Klassifikations- und Regressionsbäume

Der Klassifikator ist in der Lage, ein gelerntes Objekt in einem Bild zu lokalisieren. Für jedes weitere Objekt muss ein neuer Klassifikator gelernt werden. Ebenso müsste man für jede mögliche Ansicht eines Objekts einen neuen Klassifikator bzw. eine neue Kaskade lernen. Klassifikations- und Regressionsbäume (engl.: *Classification and Regression Tree, CART*) versprechen hierbei Abhilfe.

Statt einfacher Merkmale werden nun binäre Bäume verwendet. Jeder Knoten besitzt ein Merkmal und der Auswertungsalgorithmus entscheidet anhand eines gelernten Schwellwerts, ob der rechte oder linke Nachfolgerknoten evaluiert wird. In den Blättern befinden sich Rückgabewerte, die analog zu den einfachen Merkmalen so gewählt sind, dass ein minimaler Fehler entsteht. Abbildung 4.16 zeigt links ein einfaches Merkmal und rechts einen Klassifikations- und Regressionsbaum.

Für den Aufbau der Klassifikations- und Regressionsbäume werden für eine Basisdetektorgröße, z.B. 30×30 Pixel, alle möglichen Merkmale erzeugt. Jedes Merkmal ist Ausgangspunkt für einen Baum, indem für den aktuellen Knoten zwei Nachfolgerknoten angelegt werden. Das Aufteilkriterium wird durch den Schwellwert repräsentiert; dieser ist so gewählt, dass möglichst unvermischte Teilmengen entstehen (engl.: *reduce impurity*) [173]. Das Fehlermaß sind also die falsch klassifizierten Bilder, die es zu reduzieren gilt. Der Aufteilungsprozess setzt sich dann rekursiv fort.

Der Gentle Ada Boost-Algorithmus für Klassifikations- und Regressionsbäume

Die Ada Boost-Algorithmen können, wie auf Seite 82 dargestellt, dazu benutzt werden, Entscheidungsbäume zu lernen. Im diskreten Fall gibt es wiederum nur die Rückgabewerte $\{-1, 1\}$, beim Real Ada Boost wird eine logarithmische Fehlerfunktion verwendet und beim Gentle Ada Boost-Algorithmus der einfache Fehler. Wie Lienhart und Kollegen herausgefunden haben, ist der Gentle Ada Boost-Algorithmus das beste Lernverfahren für das Detektieren von Gesichtern [126]. Für die

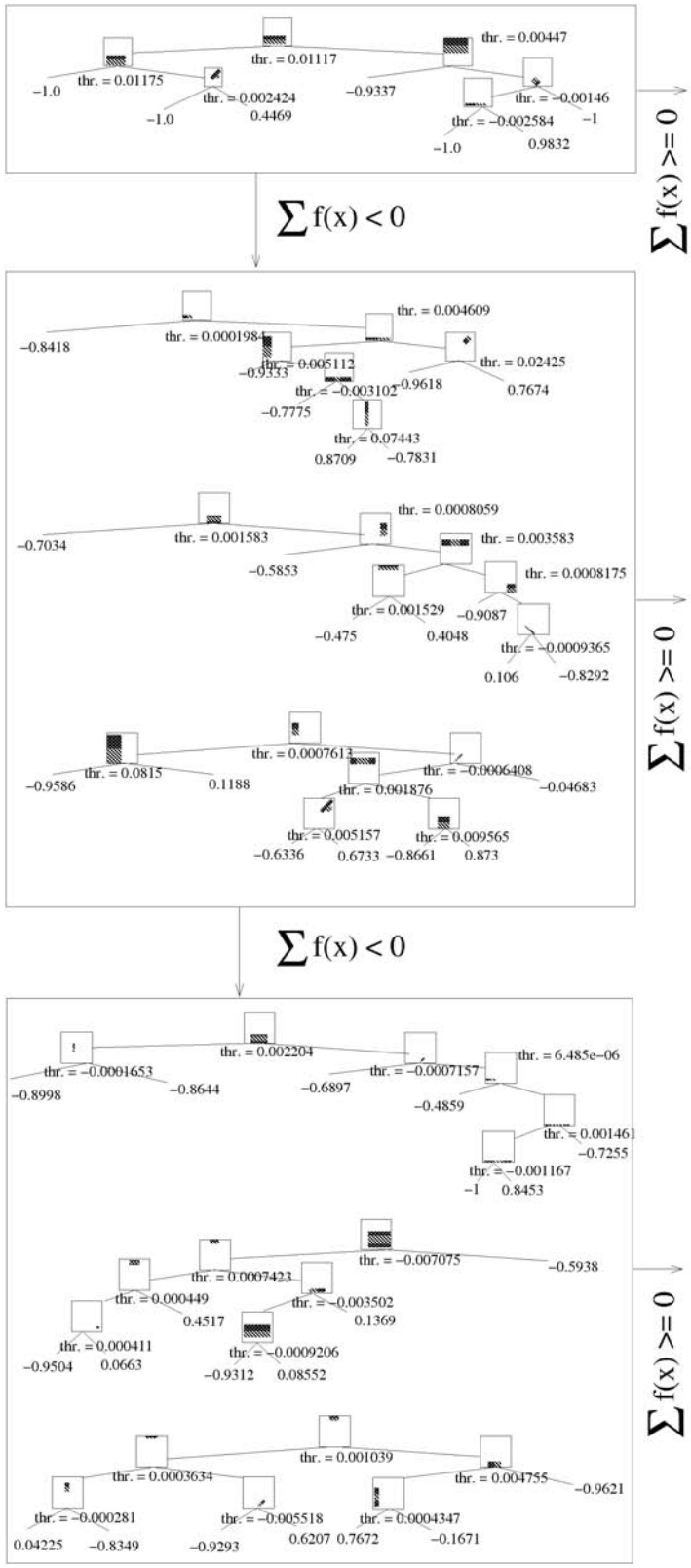


Abbildung 4.17: Die ersten drei Etappen einer Kaskade von Klassifikatoren, um das Objekt „Volksbot“ zu detektieren. Jede Etappe enthält ein Ensemble von Klassifikations- und Regressionsbäumen, die wiederum Haar-Merkmale verwenden. Die Entscheidung in jeder Etappe wird bezüglich $\sum f(x)$ gefällt, wobei $f(x)$ vom Weg durch die Bäume abhängt.

Zahl der Aufteilungen im Baum zeigen sie ebenfalls die Existenz eines Optimums. Für das Objekt „Volksbot“ [16] wurden z.B. 6 innere Knoten benutzt, um gute Ergebnisse zu erzielen. Auch die gelernten Bäume lassen sich in einer Kaskade anordnen, da gute Raten falscher Alarme und eine hohe Auswertungsgeschwindigkeit erreichbar sind. Abbildung 4.17 gibt die ersten 3 Etappen der Kaskade für den Volksbot wieder.

4.3.4 Lokalisierung von Objekten

Bestimmung potentieller Objektpunkte

Nach der Klassifikation von Objekten in 2D-Projektionen müssen diejenigen 3D-Punkte gefunden werden, die innerhalb der klassifizierten Bildfläche liegen. Dazu werden von allen Messpunkten aus Strahlen zur virtuellen Kamera und der Projektionsebene zurückverfolgt und dabei bestimmt, ob die Punkte innerhalb des Klassifikators liegen (engl.: *ray tracing*). Hierbei wird eine spezielle `OpenGL` Projektionsmatrix angewendet. Abbildung 4.18 zeigt einen klassifizierten Bildausschnitt (rechts) und die zugehörigen hellgrau eingefärbten 3D-Punkte (Mitte und rechts).

Das Einpassen von 3D-Modellen

Nachdem diejenigen 3D-Punkte bestimmt sind, die das Objekt enthalten, gilt es, seine Lage möglichst exakt zu bestimmen. Dabei kommt wiederum eine auf Punkten basierende Einpasstechnik zum Einsatz.

Die potentiellen Objektpunkte werden in einer Menge M gespeichert, die Modelle liegen als Punktmenge D vor. Mit Hilfe des ICP-Algorithmus (vgl. Seite 12) lassen sich diese beiden Mengen in Übereinstimmung bringen. Da hier kein Schwellwert $d_{\max}$ verwendet wird, ist garantiert, dass der Algorithmus monoton den Wert der Fehlerfunktion minimiert und der Algorithmus terminiert (vgl. Seite 12). Nach dem Matching sind diejenigen Punkte, die zum Objekt gehören, genau bestimmt.

Evaluation der Einpassung

Ein weiterer Schritt der Objekterkennung ist die Evaluation der Einpassung. Für viele Applikationen sind Informationen über die Qualität des Matchings wichtig, beispielsweise bei schwierigen Navigationsaufgaben. Der Wert der Fehlerfunktion des ICP-Algorithmus (3.1) liefert diese Information nicht, denn die Dichte der Punkte in den Mengen M und D tragen zum Wert bei. Unterschiedliche Punktdichten haben ihre Ursache in der radialen Abstrahlung der Laserstrahlen und somit werden Objekte, die sich nahe am Scanner befinden, mit einer höheren Auflösung abgetastet. Ein kompetitiver Lernalgorithmus dient zur Reduktion der 3D-Daten.

Kompetitives Lernen von Objekten. Das Ziel des kompetitivien Lernens von Objekten ist die Reduzierung von 3D-Daten auf eine Menge mit einer festen Anzahl von Elementen. Dabei muss sowohl der zu erwartende Quantisierungsfehler minimiert, als auch die Entropie maximiert werden. Dies bedeutet, eine endliche Menge D von 3D-Datenpunkten wird auf eine Menge $A = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_S\}$ reduziert und der Fehler

$$E(D, A) = \frac{1}{|D|} \sum_{\mathbf{w}_i \in A} \sum_{\boldsymbol{\xi} \in R_c} ||\boldsymbol{\xi} - \mathbf{w}_i|| \tag{4.11}$$

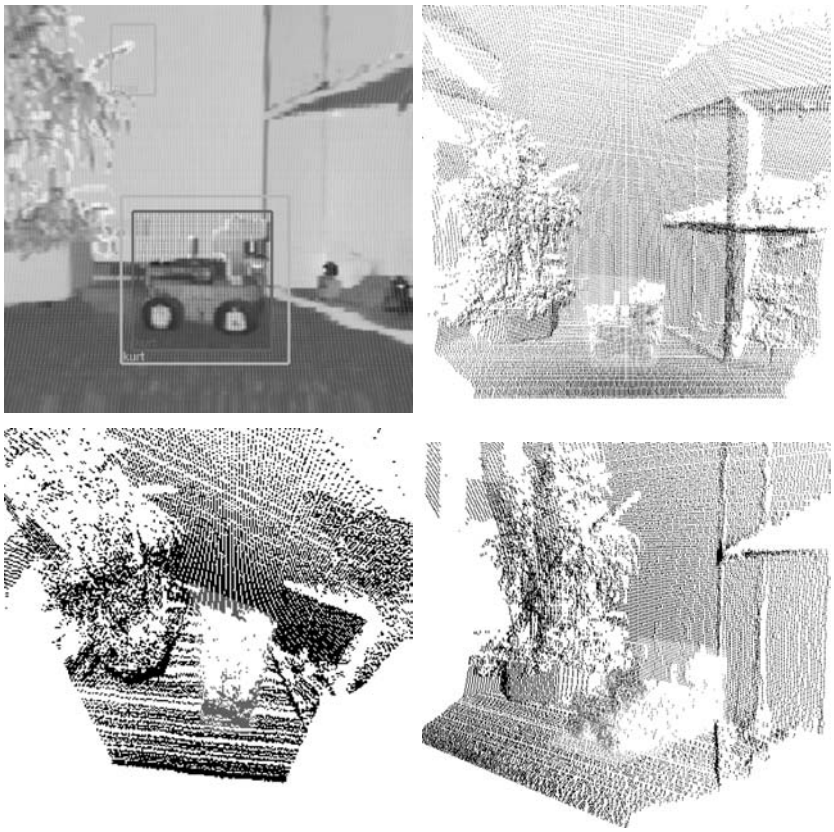


Abbildung 4.18: Bestimmung potentieller Objektpunkte durch Strahlenverfolgung. Links oben: Alle Punkte innerhalb des positiv klassifizierten Bildausschnitts werden extrahiert. Rechts oben und unten: 3D-Ansicht. Die gefundenen 3D-Punkte (innerhalb der Sichtbarkeitspyramide) sind hellgrau dargestellt.

minimiert. Hierbei ist $|A| = S = 250$ die Anzahl der auszuwählenden Samples und R_c die Voronoi-Menge der Signale c, d.h. $R_c = \{\boldsymbol{\xi} \in D | s(\boldsymbol{\xi}) = c\}$ und $s(\boldsymbol{\xi}) = \arg\min_{c \in A} ||\boldsymbol{\xi} - \mathbf{w}_c||$. Die Anzahl der Signale (hier: 250) muss so gewählt werden, dass sich alle Objekte der Datenbasis damit darstellen lassen. Abgesehen davon hat sie keinen Einfluss auf die Ergebnisse, sie beeinflusst lediglich die Rechenzeit und sollte daher klein gewählt werden. Die Entropiemaximierung garantiert eine inhärente Robustheit. Das Fehlen eines Messwertes, also eines Referenzpunktes, übt lediglich Effekte auf eine kleine Menge von Signalen aus. Interpretiert man die Erzeugung eines Eingabesignals und die nachfolgende Zuordnung zum nächsten Punkt in A als ein Zufallsexperiment, das den Wert $\mathbf{x} \in A$ einer Zufallsvariablen X zuordnet, dann ist die Maximierung der Entropiefunktion

$$H(X) = -\sum_{\mathbf{x} \in A} P(\mathbf{x}) \log(P(\mathbf{x})) \tag{4.12}$$

gleichbedeutend mit einer Gleichverteilung der Samples.

Der folgende Algorithmus wird als neuronales Gas bezeichnet (engl.: *neural gas*). Er minimiert den Fehler (4.11) und maximiert die Entropie (4.12) [88]:

1. Initialisiere die Menge A mit S Vektoren zufällig mit Hilfe der Eingabemenge D. Setze $t := 0$.

2. Ziehe zufällig ein Element $\boldsymbol{\xi}$, d.h. bestimme einen 3D-Punkt aus der Menge D.

3. Ordne alle Elemente von A bezüglich Ihres Abstandes zu $\boldsymbol{\xi}$. Dieses Sortieren ist gleichbedeutend mit dem Festlegen der Reihenfolge von Indizies $(i_0, i_1, \ldots, i_{N-1})$, so dass $\mathbf{w}_{i_0}$ derjenige Referenzvektor ist, der sich am nächsten zu $\boldsymbol{\xi}$ befindet, dass $\mathbf{w}_{i_1}$ der zweitnächste Referenzvektor ist, etc. $\mathbf{w}_{i_k}$, $k = 0, \ldots, S-1$, ist der Referenzvektor, für den k Vektoren $\mathbf{w}_j$ existieren, die näher an $\boldsymbol{\xi}$ liegen. $k_i(\boldsymbol{\xi}, A)$ bezeichnet die Zahl k, die mit $\mathbf{w}_i$ assoziiert ist.

4. Passe die Referenzvektoren nach folgendem Schema an:

$$\Delta \mathbf{w}_i = \varepsilon(t) h_\lambda(k_i(\boldsymbol{\xi}, \mathcal{A})) \cdot (\boldsymbol{\xi} - \mathbf{w}_i),$$

   und benutze dabei die Zeitabhängigkeiten:

$$\begin{aligned} \lambda(t) &= \lambda_i(\lambda_f/\lambda_i)^{t/t_{\max}}, \\ \varepsilon(t) &= \varepsilon_i(\varepsilon_f/\varepsilon_i)^{t/t_{\max}}, \\ h_\lambda(k) &= \exp(-k/\lambda(t)). \end{aligned}$$

5. Erhöhe den Zeitparameter t.

Der Algorithmus des neuronalen Gases wird mit den Parametern $\lambda_f = 0.01$, $\lambda_i = 10.0$, $\varepsilon_i = 0.5$, $\varepsilon_f = 0.005$, $t_{\max} = 10000$, $S = 250$ gestartet. Die Rechenzeit kann durch den Parameter $t_{\max}$ kontrolliert werden. Da in jedem Schritt die Samples S sortiert werden müssen, hat $|S|$ wesentlichen Einfluss auf die Rechenzeit. Der verwendete Sortieralgorithmus ist der Introsort-Algorithmus [142]. Introsort ist eine Variation von QuickSort, die bei pathologischen Fällen auf ein anderes Sortierverfahren mit $\mathcal{O}(n \log n)$-Worst Case Komplexität (z.B. Heapsort) zurückfällt. Auf diese Weise wird die Geschwindigkeit von Quicksort mit einem $\mathcal{O}(n \log n)$-Worst-Case Verfahren gekoppelt [142]. Abbildung 4.19 zeigt die 3D-Modelle der Datenbasis (links) und die reduzierten Versionen (rechts) mit je 250 Punkten.

Qualitätsbestimmung der Einpassung. Die Qualität der Einpassung lässt sich für zwei Punktmengen ermitteln, die die gleiche Punktanzahl aufweisen und die aus einem randomisierten Reduzierungsprozess hervorgegangen sind, der die Prinzipien Fehlerminimierung und Maximierung der Entropie befolgt. Für die Qualitätsbestimmung werden die kürzesten Distanzen d_{ij} zwischen dem i-ten Punkt der Datenmenge D und dem j-ten Punkt der Modellmenge M bestimmt. Letztere Menge ist Bestandteil der Datenbasis (vgl. Abbildung 4.10). Eine Analyse zeigt, dass die Distanzen eine sehr typische Verteilung aufweisen (vgl. Abbildung 4.20). Viele sind sehr klein, d.h. kleiner als 0.3 cm, ebenso sind sehr viele Distanzen größer als 1 cm. Daher lässt sich immer ein guter Schwellwert finden. Nach dem Teilen der Abstandsmenge d_l ($l \in \{0, 1, \ldots, 250\}$) berechnet der

Abbildung 4.19: Links: 3D-Modelle (Punktwolken) der Datenbasis. Rechts: Reduzierte Modelle mit je 250 3D-Punkten.

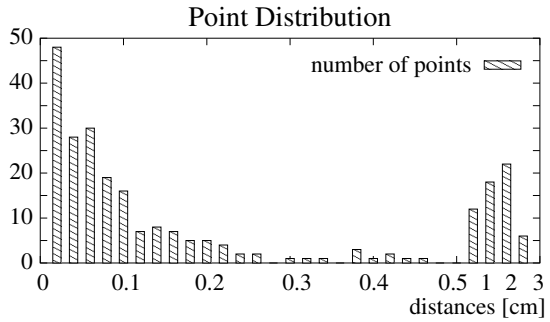


Abbildung 4.20: Eine typische Verteilung der Distanzen zwischen den nächsten Punkten bei fester Punkteanzahl.

Algorithmus den Mittelwert und die Standardabweichung des ersten Teiles:

$$\mu = \frac{1}{N'} \sum_{l=1}^{N'} d_l \qquad \sigma = \sqrt{\frac{1}{N'} \sum_{l=1}^{N'} (d_l - \mu)^2}.$$

Basierend auf diesen Werten bestimmt der Evaluationsalgorithmus die Qualität des Matchings Q als Funktion von μ und σ wie folgt: $Q = \mu + 3\sigma$. Kleine Werte von Q korrespondieren mit qualitativ guten Objekteinpassungen, wohingegen größere Werte schlechte Matchings darstellen.

Die Abbildungen 4.21 und 4.22 zeigen einige Ergebnisse der Objekterkennung. Tabelle 4.2 gibt die Detektionsraten, die Rate der falschen Alarme sowie die benötigte Rechenzeit wieder.

Abbildung 4.21: Objektdetektion und -lokalisierung. Von links nach rechts pro Zeile: (1) Detektion mit Hilfe einer einzelnen Kaskade von Klassifikatoren. Hellgrau: Detektion im Reflektionsbild. Grau: Detektion im Tiefenbild. (2) Detektion mit der kombinierten Reflektions- und Tiefenbildkaskade. (3) In das Tiefenbild wurde das 3D-Modell (hellgrau) eingezeichnet. (4) Darstellung der lokalisierten Objekte in den 3D-Scandaten.

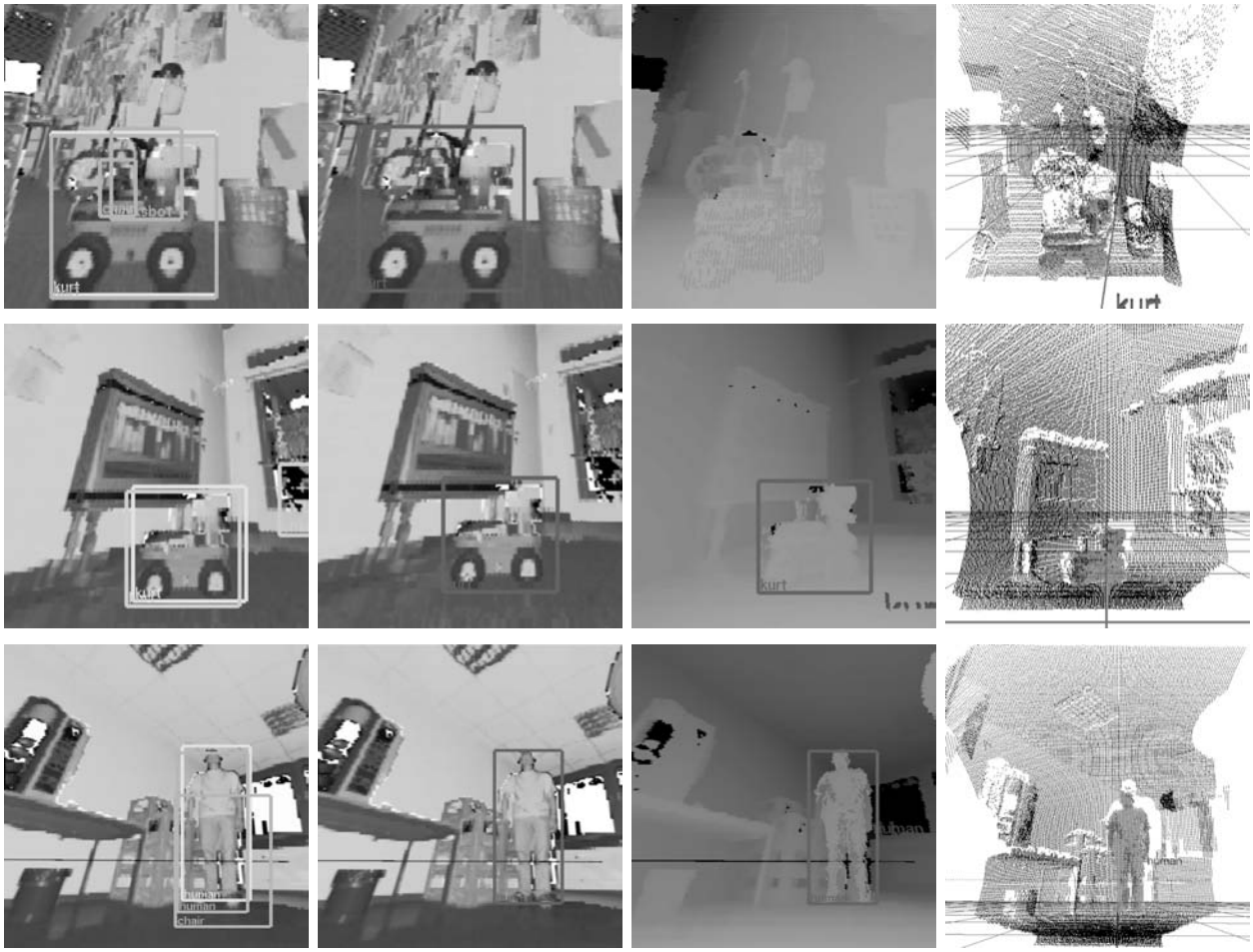


Abbildung 4.22: Objektdetektion und -lokalisierung (Forts.).

Tabelle 4.2: Objektname, Anzahl der gelernten Etappen für die Objektklassifikation und die Detektionsrate bzw. Rate falscher Alarme aufgeschlüsselt nach Tiefen- und Reflektionsbild. Durch die UND-Verküpfung ist das Minimum für die erzielten Raten ausschlaggebend. Des Weiteren ist die Rechenzeit für die Objekterkennung, die Einpassung und die Evaluation angegeben.

Objektklasse	Anzahl der Etappen	Detektionsrate (Reflektions- / Tiefenbild)	Rate falscher Alarme (Reflektions- / Tiefenbild)	Auswertungs-zeit
Bürostuhl	15	0.767 (0.867 / 0.767)	12 (47 / 33)	1.9 sec
Roboter Kurt3D	19	0.912 (0.912 / 0.947)	0 (5 / 7)	1.7 sec
Volksbot	13	0.844 (0.844 / 0.851)	5 (42 / 23)	2.3 sec
Mensch	8	0.961 (0.963 / 0.961)	1 (13 / 17)	1.6 sec

4.4 Automatisches Lernen von 3D-Objekten

Die Eingabe der Lernalgorithmen zur Objektklassifikation und -lokalisierung bilden Beispielpaare der Form $(x_1, y_1), \ldots, (x_N, y_N)$. Die klassifizierte Ausgabe ist $y_i \in \{-1, 1\}, i \in \{1, \ldots, N\}$. Sie wird manuell vorgegeben. Um nach der Lernphase Objekte sicher zu detektieren, sind zirka 1000 positive Beispiele notwendig. Die negativen Beispiele können beliebige Bilder sein, die nicht das Objekt enthalten.

Die erforderliche Anzahl positiver Beispiele generiert ein OpenGL-basiertes Programm aus zirka 200 3D-Scans. Dafür rotiert das Programm die aufgenommene 3D-Szene und verändert den Öffnungswinkel der virtuellen Kamera [149]. Abbildung 4.23 zeigt einen 3D-Scan und die aus ihm erzeugten Eingabebilder, in denen jeweils die Objekte manuell markiert werden.

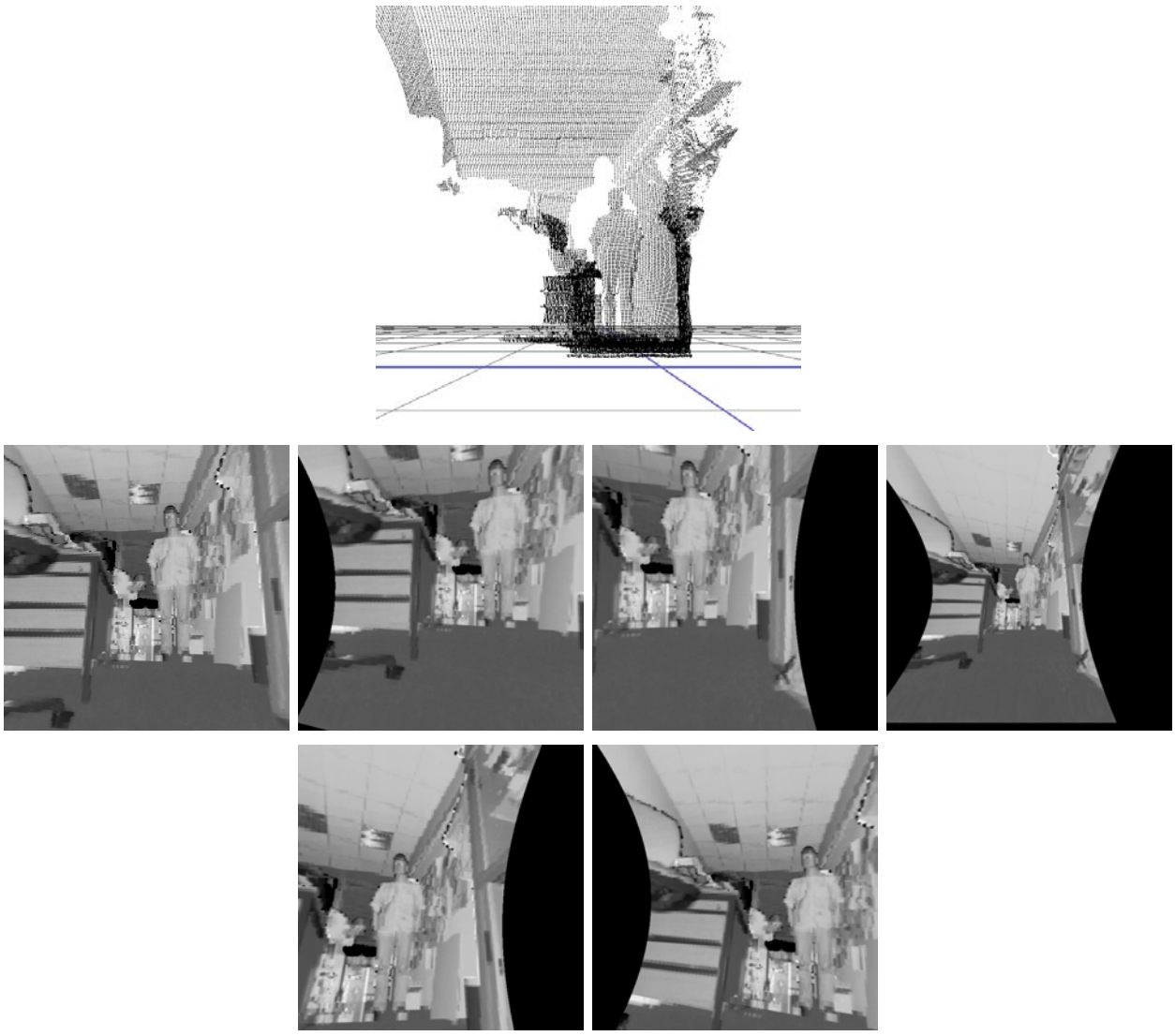


Abbildung 4.23: Oben: 3D-Punktwolke für das Lernen von Objekten. Mitte und unten: Bilder für positive Beispiele mit gedrehten Kamerapositionen. Die Ursache der schwarzen gewölbten Ränder liegt im Scanmodus. Es wurde mit einem Öffnungswinkel von 45 Grad gescannt.

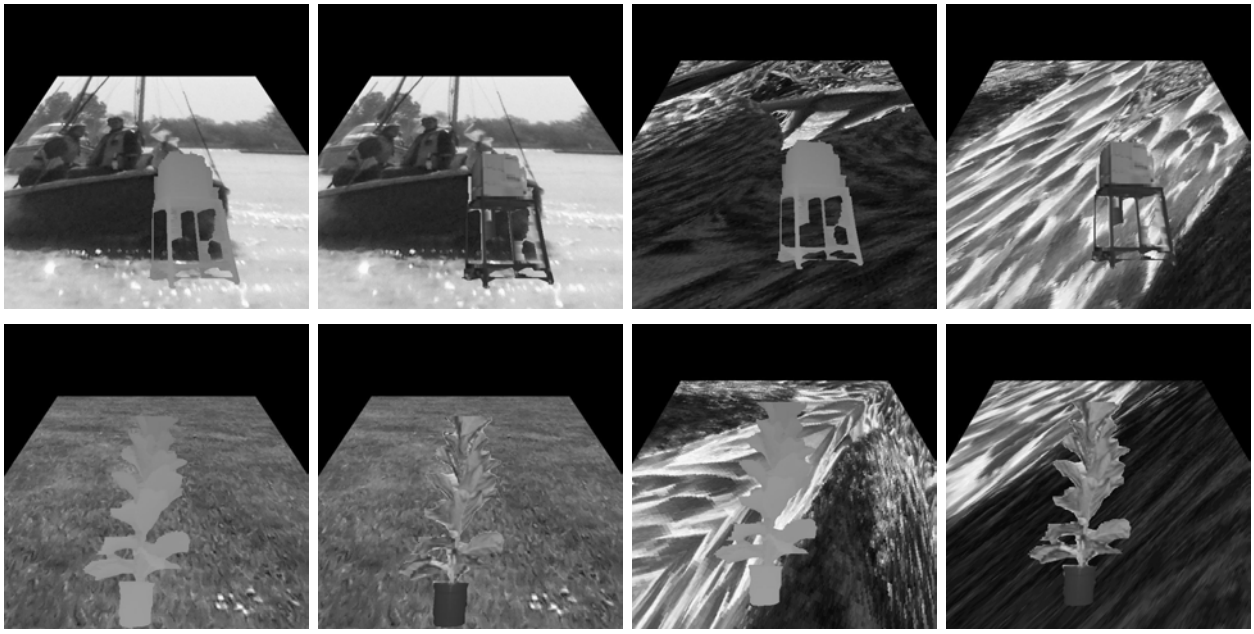


Abbildung 4.24: Oben: Generierte positive Beispiele für das Objekt „printer“. Unten: Positive Beispiele für das Objekt „rubbertree“. Das Verfahren zum automatischen Lernen verwendet beliebige Bilder als Hintergründe und erstellt positive Lernbeispiele sowohl für Tiefen- als auch für Reflektionsbilder.

Das Erzeugen positiver Beispiele ist sehr aufwändig. Rechnet man 2 bis 4 Minuten für die Aufnahme und Verarbeitung eines der 200 3D-Scans, ist man einen Tag mit einem Objekt beschäftigt. Das eigentliche Lernen hingegen dauert zirka 0.5 – 3 Stunden pro Objekt. Die 3D-Scans bergen genügend Informationen, um die Trainingsdaten aus einem Scan heraus zu erzeugen. Das folgende Verfahren generiert die positiven Beispiele und verringert dabei gleichzeitig den manuellen Aufwand:

1. Nimm zunächst *einen* 3D-Scan auf.
2. Extrahiere die zu einem Objekt gehörenden 3D-Punkte aus dem Scan und entferne alle weiteren Punkte.
3. Generiere die positiven Beispiele durch das Platzieren verschiedener Hintergründe hinter dem Objekt und speichere das entstandene Bild. Weitere Beispiele lassen sich durch Rotieren der Szene und Verändern des Öffnungswinkels erzeugen.

Im obigen Algorithmus muss nur ein 3D-Scan aufgenommen und das Objekt auch nur einmal manuell markiert werden. Abbildung 4.24 zeigt exemplarisch vier positive Beispiele für Tiefen- und Reflektionsbilder der Objekte „printer“ und „rubbertree“.

Der Algorithmus zum automatischen Lernen von 3D-Objekten bietet die Möglichkeit, semantische 3D-Karten zu erstellen (vgl. Abschnitt 5.3, Seite 117). Die Abbildung 4.25 zeigt einige Ergebnisse der Objekterkennung.

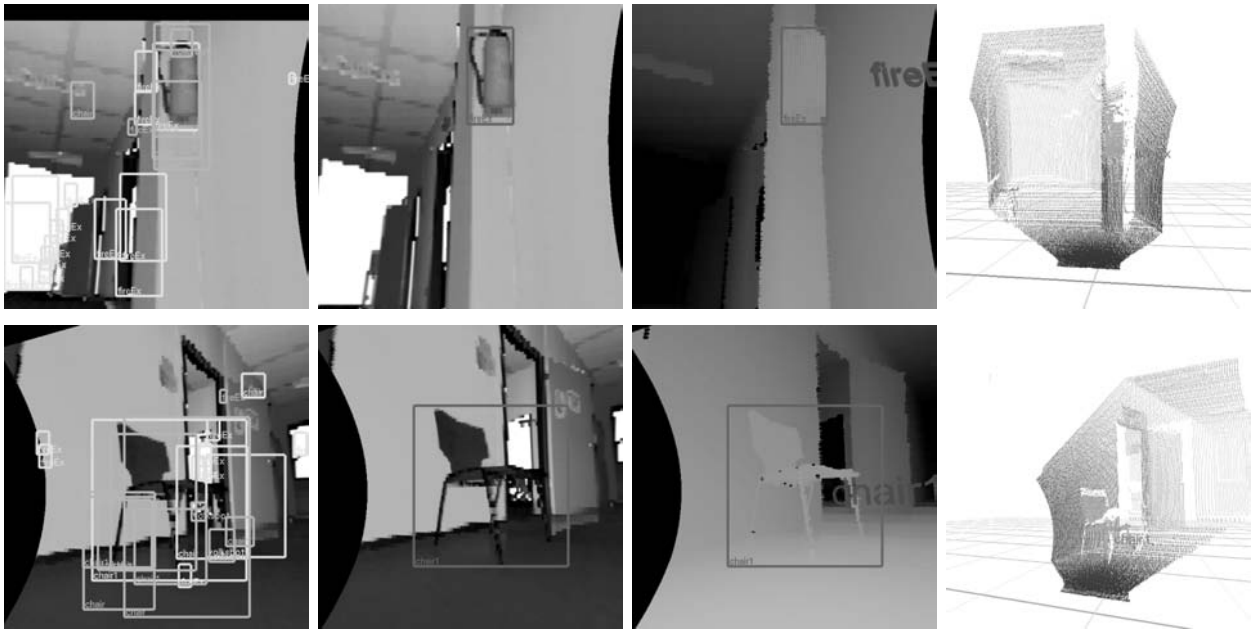


Abbildung 4.25: Objektdetektion und -lokalisierung mit automatisch gelernten Objekten. In den Bildern wird nach 8 verschiedenen Objekten gesucht. Von links nach rechts pro Zeile: (1) Detektion mit Hilfe einer einzelnen Kaskade von Klassifikatoren. Grün: Detektion im Reflektionsbild. Gelb: Detektion im Tiefenbild. (2) Detektion mit der kombinierten Reflektions- und Tiefenbildkaskade. (3) Das 3D-Modell wurde (rot) in das Tiefenbild eingezeichnet. (4) Darstellung der lokalisierten Objekte in den 3D-Scandaten.

4.5 Diskussion

Dieser Abschnitt stellt verschiedene Systeme zur Objekterkennung vor. Die den Systemen zu Grunde liegenden Methoden werden mit dem in der vorliegenden Arbeit entwickelten Verfahren verglichen.

4.5.1 Szeneninterpretation

Bereits Mitte der 70er Jahre beschäftigte man sich mit der Interpretation von Szenen und dem Lösen von *Consistent Labeling* Problemen [209]. Es standen einfache Strichzeichnungen im Vordergrund, bei denen versucht wurde, den Linien eindeutige Bezeichner zuzuordnen. Waltz' Algorithmus hat zum Beispiel zunächst eine Liste aller Linienschnitte erstellt (vgl. Abbildung 4.26, links). Auch wurden Regeln aufgestellt, die die im Bild vorhandenen Linienschnitte einteilen oder die Parallelität feststellen. Als Ergebnis liegen die möglichen Bezeichnungen für die Elemente der Strichzeichungnen vor. Damit wurde lokale Konsistenz geschaffen. Anschließend benutzte Waltz ein Constraint Satisfaction Programm, um alle möglichen global konsistenten Belegungen zu berechnen. Optional selektierte schließlich eine Heuristik eine einzige „plausible“ Interpretation, die weiterverwendet werden konnte. Diese Selektion bevorzugte unter den konsistenten Belegungen die wahrscheinlichsten, z.B. konvexe Objekte.

Mit Hilfe des oben skizzierten Algorithmus konnte zum Beispiel die Szene aus Abbildung 4.26

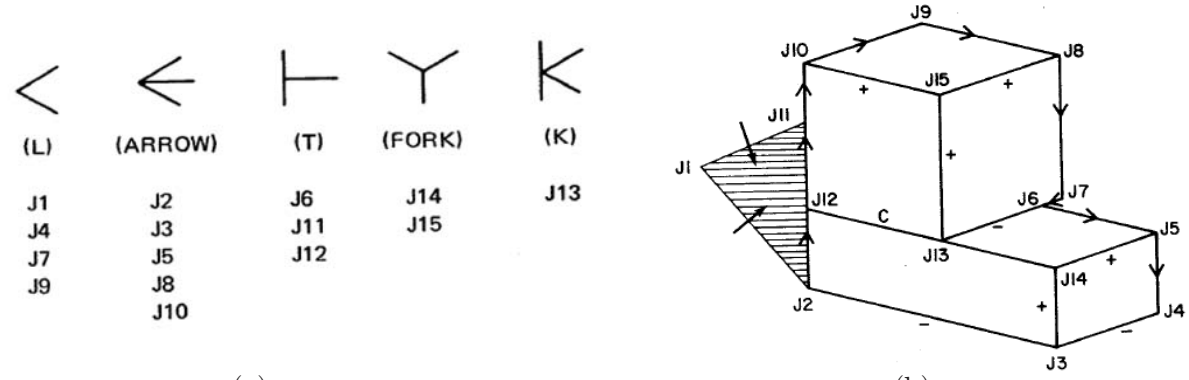


Abbildung 4.26: Interpretation einer Strichzeichnung. Links: Mögliche Schnittmuster von Linien. Quelle: [208]. Rechts: Beispielszene einer Strichzeichnung. Quelle: [208].

rechts in wenigen Sekunden richtig interpretiert werden. Dabei arbeitete das System auf der Basis von MicroPlanner und Lisp auf einer PDP-10. Es initiierte die *Forward Checking* Methode für das *Contraint Logic Programming (CLP)* [173].

Ähnlich zu dem in dieser Arbeit vorgestellten Szeneninterpretationsalgorithmus für 3D-Flächen arbeitet das semiautomatische System zur 3D-Gebäudeerfassung aus Luftbildern [39,41,69] der Universität Bonn. Das System extrahiert Kanten und schließt aus konsistenten Beschriftungen (vgl. Abbildung 4.27) auf den Typ, die Lage und die Größe von Gebäuden. Kolbe et al. [119] benutzen Prolog, um Wissen zu spezifizieren und zu externalisieren. Ihr CLP verwendet ganz ähnliche Prolog-Klauseln und zusätzlich einen expliziten *Forward Check*, z.B. für das Testen zweier Linien auf Parallelität [119]:

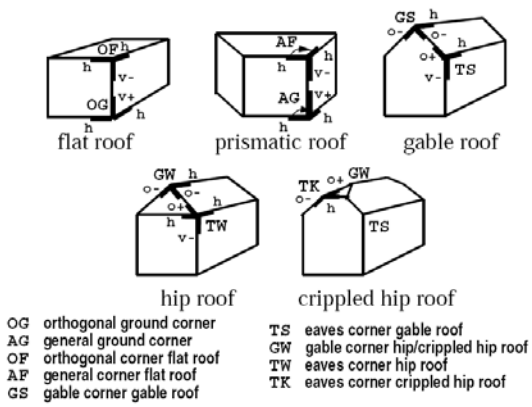


Abbildung 4.27: Kantenbeschriftung für die Interpretation von 3D-Gebäudemodellen. Quelle: [74].

```
line_parallel_test(L1,L2) :-
        line(L1,_,_,_,_,Alpha,_,_,_),
        line(L2,_,_,_,_,Beta,_,_,_),
        Diff is abs(Alpha-Beta),
        Diff < 5.
line_parallel(L1,L2) :- forward_check(L1:L2:line_parallel_test(L1,L2)).
```

Das Gebäudeerfassungsystem wird abgerundet durch Lernalgorithmen, die auf den Graphenrepräsentationen der Gebäude arbeitet [41, 69, 74].

Kolbes Interpretationssystem ist ausgefeilter und erlaubt mehr Einstellungen als das in dieser Arbeit vorgestellte. Eines in diesem Maße ausdifferenzierten Systems bedarf es hier nicht, da das Ziel der Szeneninterpretation für 3D-Scans die Interpretation der Grobstruktur ist, d.h. die der extrahierten, großen Flächen und der daraus folgenden Verbesserung des 3D-Modells. Somit hat aber eine Interpretation Auswirkungen auf die Messdaten und direkten Einfluss auf die Qualität der semantischen 3D-Karte. Im Folgenden wird die Objekterkennung für Freiform-Objekte in den wissenschaftlichen Zusammenhang eingeordnet.

4.5.2 Objekterkennung

Im Allgemeinen erfolgt die computerbasierte Objekterkennung in zwei Schritten [79], eine aufgestellte Hypothese wird durch einen Test überprüft:

1. Die Algorithmen stellen zunächst Korrespondenzen zwischen einer Kollektion von Bild- und Objektmerkmalen her. Anschließend erzeugen sie daraus eine Hypothese über die Projektion des Objektkoordinatensystems in das Bildkoordinatensystem [79].

2. Mit Hilfe der Hypothese rendern die Algorithmen das Objekt (Rückprojektion). Anschließend wird das gerenderte Bild mit dem Eingabebild verglichen. Falls ein Maß an Ähnlichkeit erreicht ist, akzeptiert das Erkennungsprogramm die Hypothese.

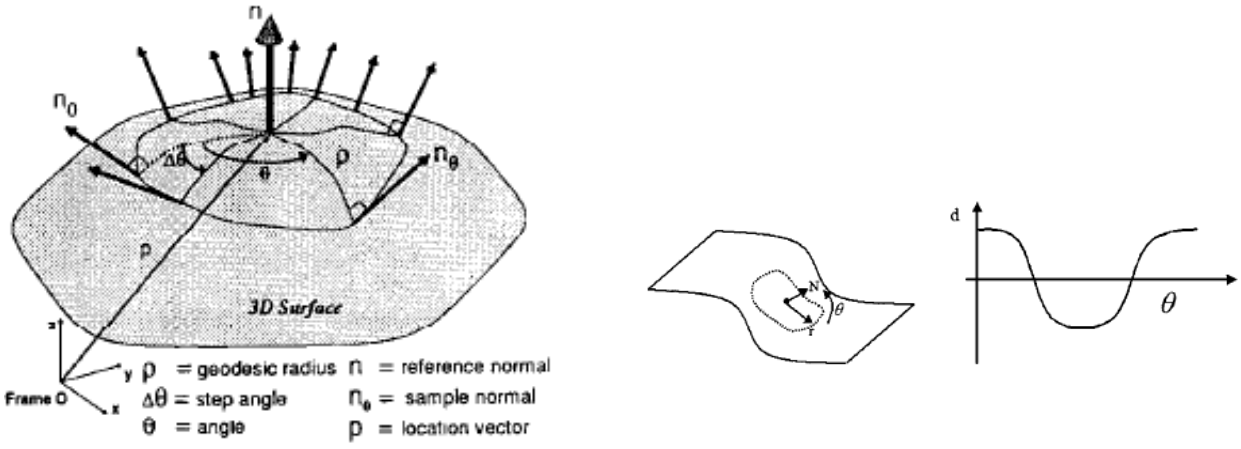
Das in Abbildung 4.10 (Seite 79) vorgestellte System ist dieser Lehrbuchmethode sehr ähnlich. Nach dem 3D-Scannen und der Datenvorverarbeitung (Schritte 1 und 2) wird eine Hypothese aufgestellt (Schritte 3 – 5). Hierbei schätzt der Algorithmus nicht die Projektionsparameter, sondern äquivalent dazu die Objektpose. Anschließend evaluiert Schritt 6 die Objekteinpassung. Der letzte Schritt im System besteht in der Visualisierung der Ergebnisse.

Objekterkennung in Tiefenbildern

Besl bewertet die Objekteinpassung von beliebigen Objekten in Tiefenbilder unter Verwendung von Punkt-, Kurven- und Oberflächenmerkmalen [32]. Die Komplexität von brute-force Verfahren ist exponentiell [32], deshalb sind diese Verfahren nicht einsetzbar. Daher haben unter anderem Besl und McKay [31], Johnson und Hebert [115] sowie Chua und Jarvis [52] Methoden entwickelt, die das Einpassungsverfahren beschleunigen. Tiefenbilder enthalten im Vergleich zu Reflektions- und Intensitätsbildern vollständigere Informationen über die Objektgeometrie. Objekterkennungsalgorithmen nutzen die 3D-Informationen und vergleichen geometrische Relationen zwischen Tiefenbildern und den Trainingsdaten bzw. 3D-Modellen. Während die Rekonstruktion von 3D-Modellen ein neues und hochaktuelles Forschungsthema ist, hat die Objekterkennung in Tiefenbildern eine längere Tradition [44].

Dorai und Jain entwickelten das System COSMOS (engl.: *curvedness-orientation-shape map on sphere*) [64]. Zur Objekterkennung verwenden sie lokale Oberflächenformen, die sie in eine relativ kleine Menge repräsentativer Formen klassifizieren. Hierzu benutzen sie die Gauss-Krümmung, die mittlere Krümmung, einen Formindex und die so genannte *curvedness*. Diese vier Maße lassen sich aus der prinzipiellen Krümmung berechnen. Die prinzipielle Krümmung an einem 3D-Punkt ist charakterisiert durch die beiden Richtungen, an denen sich die Oberflächennormale am meisten und am wenigsten verändert [33]. Dorai und Jain [64] benutzen die vier Maße, um eine betrachtungsabhängige Repräsentation der Objekte in Form von Histogrammen zu erstellen. Diese Histogramme lassen sich schnell vergleichen. Somit wird das Objekt eingepasst.

Zusätzlich zu der Verwendung krümmungsbasierter Merkmale von Dorai und Jain bzw. von Besl und Jain wurden deformierbare Gitter benutzt, um lokale Oberflächengestalten darzustellen. Pipitone und Adams setzen eine verbundene Menge von gleichseitigen Dreiecken (vgl. Abbildung 4.28) ein, um die Objektform zu charakterisieren [161]. Die Dreiecke werden so deformiert, dass sie auf der Oberfläche liegen. Basierend auf den Winkeln zwischen Dreieckspaaren errechnen Pipitone und



(a) (b)

Abbildung 4.29: Links: Die Splash-Repräsentation. Die Winkeldifferenz zwischen der Normalen **n** und der Normalen $\mathbf{n}_\theta$ um **n** ist kodiert. Quelle: [185]. Rechts: Punktsignatur. Sie entsteht durch einen Schnitt einer Kugel mit der Objektoberfläche [52].

Adams einen posen-invarianten Merkmalsvektor, der die lokalen Orientierungsänderungen der Oberfläche repräsentiert. Um ein Objekt vollständig zu codieren, werden von zufällig gewählten Positionen Merkmalsvektoren errechnet. Anschließend wird die eigentliche Objekterkennung durch das Maximieren der posteriori-Wahrscheinlichkeit zwischen im Bild beobachteten und in der Modellbasis vorhandenen Operatoren durchgeführt.

Stein und Medioni nutzen ebenfalls Veränderungen der Oberflächenorientierung für ihre Objekterkennung [185]. Sie verwenden ein lokales Matching, das es ihnen erlaubt, Objekte in ungeordneten Szenen zu erkennen. Um ein gutes Matching zu erstellen, haben Stein und Medioni einen Weg gefunden, die Differenz zwischen Verteilungen von Oberflächennormalen zu messen. Abbildung 4.29 (a) zeigt diese Splashrepräsentation, die an Tropfenbildung durch Wasser erinnert. Zur Objekterkennung werden diese Splashmerkmale an Stellen mit großer Oberflächenkrümmung berechnet. Anschließend verwenden Stein und Medioni einen so genannten Strukturindex. Dieses Indexverfahren ist ein Hashverfahren, wobei die Indizes in der Hashtabelle Bezug zu Objektstrukturen haben. Nachdem so Matchinghypothesen aufgestellt worden sind, werden diese Posen mit den Objektpunkten verifiziert.

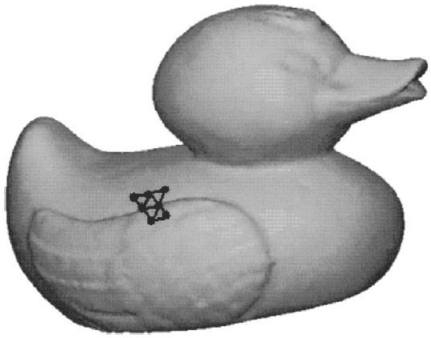


Abbildung 4.28: Beispiel eines Operators, der auf Dreiecken beruht. Quelle: [44].

Chua und Jarvis entwickelten aus der Splash-Repräsentation die Punktsignaturen [52]. Statt Informationen über Normalen zu kodieren, benutzen Chua und Jarvis minimale Abstände zwischen Punkten auf einer 3D-Kontur zu einer Referenzfläche. Die 3D-Kontur entsteht durch den Schnitt der Objektoberfläche mit einer Kugel (Abbildung 4.29 (b)). Eine Analyse der prinzipiellen Komponenten (PCA) definiert die Ebene. In ihr sind die Abstände der 3D-Konturpunkte zur Ebene am geringsten. Diese Ebene wird nun solange verschoben, bis sie den Punkt der Oberfläche enthält.

Die vorzeichenbehaftete Distanz von der 3D-Kontur zum Oberflächenpunkt ist eine eindimensionale Kurve (vgl. Abbildung 4.29 (b), rechts), die im diskretisierten Fall eine sehr kompakte, poseninvariante Darstellung der Oberflächenstruktur um einen Punkt ist. In der Erkennungsphase vergleicht man nun Oberflächenstrukturen mit Hilfe der Signaturen zwischen Modell und Szenenpunkten, um Hypothesen aufzustellen, die anschließend verifiziert werden.

Die aktuellste Arbeit zur Objekterkennung in Tiefenbildern stammt von Johnson und Hebert [114, 115]. Correa et al. entwickeln diese Arbeiten weiter [171]. Dabei verwenden sie ebenfalls Merkmale von Punkten. Ihre „Spin Images“ (Spinbilder) sind 2D-Histogramme der Oberflächenpunkte um einen Punkt. Die Spinbilder werden erzeugt, indem der Algorithmus eine Ebene um das Normal eines Punktes rotieren lässt (vgl. Abbildung 4.30 und 4.31). Während die Ebene rotiert, werden die Punkte, die in der rotierenden Ebene liegen, in das Histogramm aufgenommen. Spinbilder setzen eine polygonale Oberflächenapproximation der Modelle bzw. Tiefenbilder voraus. Dies bedeutet, dass zuvor entweder äquidistante Gitter erzeugt werden müssen oder anderweitig sichergestellt werden muss, dass Punkte uniform verteilt sind. Die Spinbilder werden wiederum dazu benutzt, Korrespondenzen zwischen Szenen und Modellpunkten zu errechnen. Nachdem diese erstellt sind, benutzen Johnson und Hebert einen modifizierten ICP-Algorithmus für die Verifikation und die genauen Einpassung des Modells. Des Weiteren wurde in [115] die erscheinungsbasierte Objekterkennung von Murase und Nayar [139] für die Spinbilder adaptiert. Aus den Spinbildern wird ein niedrig-dimensionaler Unterraum mittels PCA berechnet, um das Matching der Objektbilder mit jenen der Szene zu beschleunigen. Weitere Vorteile bestehen in geringerem Speicherbedarf und in größerer Robustheit bei ungeordneten Szenen.

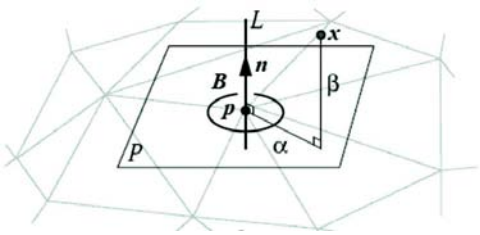


Abbildung 4.30: Definition eines Spinbildes [114]. Im Histogramm werden die Schnittpunkte als α und β gespeichert.

Die vorgestellten Verfahren ähneln dem in dieser Arbeit entwickelten. Zunächst wird eine Hypothese erzeugt und diese evaluiert, wobei die Hypothese allerdings aus den Tiefendaten errechnet wird. Dies entspricht der Anwendung des Klassifikators im Tiefenbild. Für die genaue Einpassung hat sich der ICP-Algorithmus bewährt [115].

Erkennen von 2D-Silhouetten

Neben der direkten Erkennung von Objekten in Tiefenbildern gibt es Ansätze, Silhouetten zur Charakterisierung von Objekten einzusetzen. In einer kontrollierten Umgebung können Silhouetten sehr hilfreich sein, um die Identität und die Pose von Objekten zu bestimmen. Zur Zeit sind keine silhouettebasierten Systeme bekannt, die auf Grundlage von Tiefendaten arbeiten [44], daher stellt dieser Abschnitt nur kurz Methoden vor, die mit Hilfe von Intensitätsbildern arbeiten.

Mokhtarian entwickelte ein System zur Objekterkennung, das auf geschlossenen Silhouetten basiert. Das Sytem ist in der Lage, Objekte mit wenigen Ansichten in einer Testumgebung zu erkennen [137]. Externes Licht strahlt das Objekt an, so dass sich dieses leichter vom Hintergrund separieren lässt. Die Objektgrenzen werden als geschlossener Linienzug extrahiert, der im CSS (engl.: *curvature scale space*) repräsentiert wird. Das Matching der CSS-Kurven basiert auf dem Zuordnen der Maxima und der Null-Durchgänge der Kurve. Dabei wird für eine konvexe Objektgrenze diese solange geglättet,

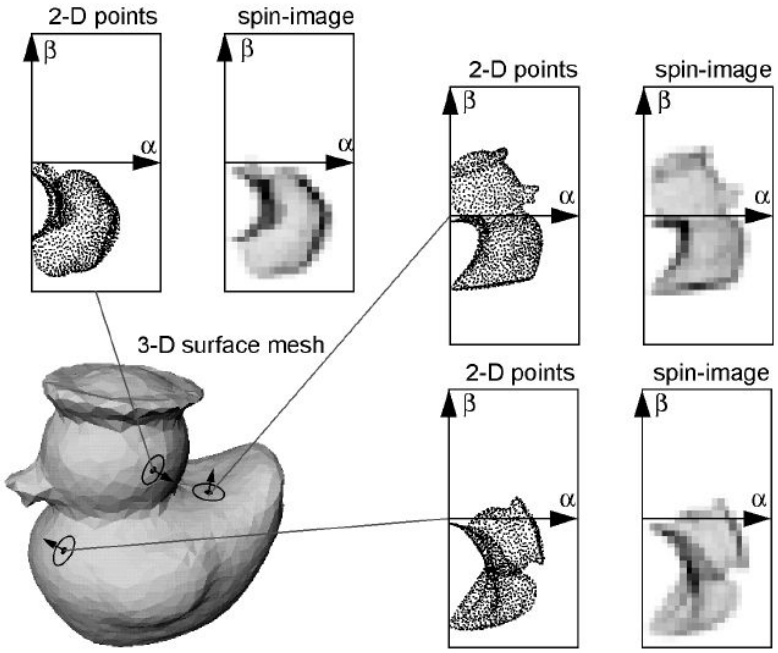


Abbildung 4.31: Anwendung von Spinimages an verschiedenen Modellpunkten. Quelle: [115].

bis die Kurve vier Maxima aufweist. Hingegen werden alle Maxima bei einer konkaven Kurve verwendet. Das Gesamtsystem ist sehr schnell und weist hohe Erkennungsraten auf [44].

Ponce und Kriegmann [162] wenden ein einfaches Kamera-Projektionsmodell [79] an, um eine 3D-Kontur mit einer 2D-Silhouette zu korrelieren. Aus den 3D-Modellen von Objekten werden zunächst Kandidaten für die 2D-Silhouetten errechnet. Diese werden mit denjenigen, die im Bild gefunden wurden, verglichen. Das System ist ebenfalls relativ schnell und erkennt die Lage der Objekte sehr genau [44].

Unter die silhouettebasierten Ansätze fallen auch kantenbasierte Ansätze. Zunächst werden Kanten, darunter auch der Umriss des Objekts, extrahiert. Anschließend werden diese für die Lagebestimmung der Objekte im Bild zugeordnet [48, 49, 123].

Erscheinungsbasierte Objekterkennung

Erscheinungsbasierte (engl.: *appearance-based*) Ansätze sind in der computerbasierten Objekterkennung sehr weit verbreitet. Diese Popularität begründet sich darin, dass diese Methode negative Effekte von Form, Pose, Reflektion und Illumination auffangen und handhaben kann [44]. Erscheinungsbasierte Ansätze zur Objekterkennung kodieren einzelne Ansichten als einen Punkt in einem hochdimensionalen Raum, dem Eigenraum. Die Basis für diesen Raum erhält man durch statistische Analyse von Trainingsbildern. Die Erkennung einer Objektansicht geschieht durch Projektion dieser Ansicht in den Eigenraum und anschließender Suche nach dem nächsten Objekt.

Erste Arbeiten in diesem Gebiet wurden von Kirby und Sirovich bei ihrer Forschung zur Gesichtserkennung durchgeführt [117]. Sie benutzen PCA, um das beste Koordinatensystem bezüglich der Kompression für ihre Trainingsdaten zu erstellen. Die Basisvektoren des neuen Koordinatensystems

heißen Eigenbilder. Turk und Pentlant untersuchten auf PCA beruhende Gesichtserkennungen und fanden heraus, dass das Verfahren robust gegen Illuminationsänderungen ist, aber Probleme mit Größenänderungen der Objekte im Bild hat [203]. Weiterhin zeigten sie mit ihrer Implementation, dass das Verfahren extrem schnell arbeitet. Des Weiteren untersuchten sie den Einsatz von neuronalen Netzen für die Klassifikation des koordinatentransformierten Bildes [203].

Murase und Nayar benutzen erscheinungsbasierte Ansätze zur Objekterkennung und zur Rückgewinnung der Objektpose [139]. Zwei Eigenräume kommen dabei zum Einsatz. Der universale Eigenraum wird von allen Trainingsbildern erstellt und dazu verwendet, die Identität des Objekts in der Bildvorlage zu klären, während je ein Objekteigenraum pro Objekt der Modell-Datenbasis erstellt wird. Der Objekteigenraum dient dazu, die Pose des Objekts zu bestimmen. Die Objekte der Modell-Datenbasis wurden mit Hilfe eines Drehtisches systematisch aufgenommen. Für jedes Objekt erstellt der Algorithmus weiterhin eine bivariate Mannigfaltigkeit, die es erlaubt, diskrete Punkte im Eigenraum zu interpolieren und die Poseerkennung zu verbessern [139].

Erscheinugngsbasierte Objeterkennung in Tiefenbildern wurde von Campbell und Flynn erstmalig untersucht [43]. Hierbei liefert die PCA eine Menge von charakteristischen Formen, so genannte Eigenshapes. Abbildung 4.32 zeigt ein Beispiel. Das entwickelte Erkennungssystem arbeitet jedoch auf synthetischen Daten, Untersuchungen an realen 3D-Scandaten stehen jedoch noch aus.

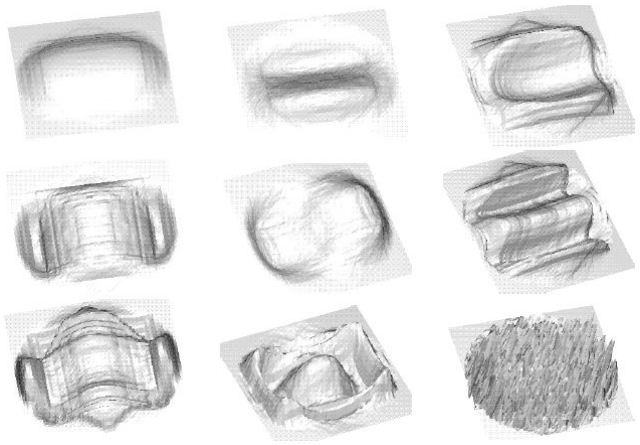


Abbildung 4.32: Beispiele für Eigenshapes. Quelle: [43].

Ein prinzipielles Problem der erscheinungsbasierten Objekterkennung – sowie der silhouettebasierten und jener mit direkter Verwendung von Tiefendaten – ist der Mangel, bei teilweisen Verdeckungen zu versagen. Huang et al. schlagen vor, die zu erkennenden Objekte in Teile zu zerlegen, die Teile danach zum Beispiel mittels PCA zu erkennen, um anschließend über die Lage der erkannten Teile Schlussfolgerungen zu ziehen [110]. Dies führt schließlich zur Verwendung von hierarchischen Datenbasen und von Bayes-Matching [45].

Kapitel 5

Ergebnisse – Kurt3D

Dieses Kapitel führt die Verfahren aus den vorangegangen Teilen zusammen und stellt das Robotersystem Kurt3D vor. Die Zusammenhänge zwischen den bisherigen Methoden werden herausgearbeitet. Des Weiteren werden Algorithmen, die für den Betrieb des Roboters notwendig sind, die Systemintegration sowie weitere Anwendungen werden präsentiert. Ein funktionierender Roboter ist die Basis der in den bisherigen Abschnitten präsentierten Algorithmen, da die Daten mit Kurt3D akquiriert worden sind.

5.1 Der mobile Roboter Kurt3D

Der mobile Roboter Kurt3D (vgl. Abbildung 5.1, links) erweitert die KURT2 Plattform [13], die als Indoor- und Outdoor-Version zur Verfügung steht. Die Maße der Plattform belaufen sich auf 45 cm (Länge) × 33 cm (Breite) × 26 cm (Höhe). Der Roboter wiegt zirka 15.6 kg. Durch den 3D-Laserscanner erhöht sich das Gewicht auf 22.6 kg. Die Gesamthöhe beträgt 47 cm. Zwei 90 W (Kurzzeitleistung: 210 W) Motoren treiben die 6 Räder an. Da sich die drei Räder einer Seite immer gleich schnell drehen, rutscht der Roboter über die Vorder- und Hinterräder, wenn er sich auf der Stelle dreht. Um die dabei entstehende Reibung zu verringern, haben die Vorder- und Hinterräder kein Profil. Die Outdoor-Version besitzt etwas größere Räder, um die Bodenfreiheit zu vergrößern.

Mit einer Batterieladung kann Kurt3D etwa 4 Stunden lang fahren (20 NiMH Akkus, Kapazität 4500 mAh). Das mitgeführte Notebook ist ein Panasonic CF-73, das auf einem Intel-Centrino-1400 MHz mit 768 MB RAM basiert. Die Akkulaufzeit des Toughbooks beträgt maximal 4 Stunden. Ein eingebetteter 16-Bit CMOS Mikrocontroller (Infinion C167) steuert die Motoren und ist über einen CAN Bus mit dem Notebook verbunden.

5.1.1 Der 3D-Laserscanner

Der 3D-Laserscanner (vgl. Abbildung 5.1, rechts) besteht aus einem 2D-Laserscanner und einer Halterung, die das Nicken des Scanners erlaubt [191]. An dieser Nick-Erweiterung ist ein Servomotor (Typ: Volz Alu-Star digital-220 mit Rutschkupplung) zur Ausführung einer kontrollierten Bewegung angebracht. Der 3D-Scanner kann 5 Stunden lang betrieben werden (Scanner: 17 W, 20 NiMH Zellen, Kapazität 4500 mAh).

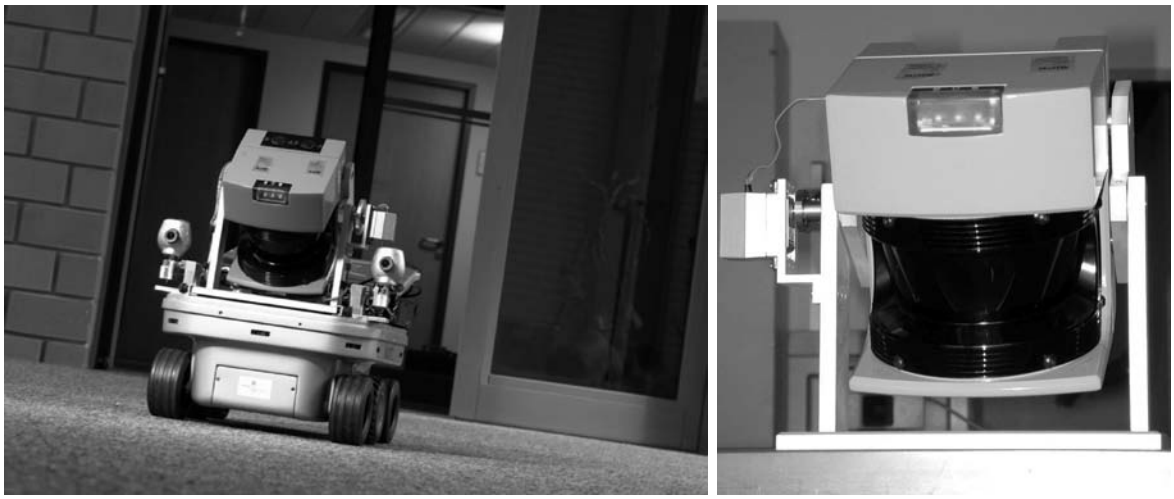


Abbildung 5.1: Links: Der Roboter Kurt3D. Rechts: Der 3D-Laserscanner.

Ein Bereich von 180°(horizontal) × 90°(max.120°)(vertikal) kann mit verschiedenen Abtastraten (horizontal: 181, 361, 721; vertikal: 128, 176, 256, 400, 500) berührungslos erfasst werden. Das Scannen einer Ebene mit 181 Abstandspunkten geschieht in 13 ms durch den SICK LMS 200. Das Scannen einer Ebene mit mehr Punkten, z.B. 361 oder 721, erfordert die doppelte bzw. vierfache Zeit. Folglich benötigt ein 3D-Scan mit 361 × 176 Punkten eine Zeit von 4.5 Sekunden. Die Auflösung in Tiefenrichtung beträgt beim LMS 200 14 Bits, die maximale Reichweite ist 32 m und die Genauigkeit wird mit einem Zentimeter angegeben. Um in sich konsistente 3D-Scans zu erhalten, werden die Scans in einem Stop-Scan-Go Prozess aufgenommen.

Neben der Abstandsmessung ist der Scanner in der Lage, die vom Objekt zum Scanner reflektierte Lichtmenge zu quantifizieren. Der so für jeden Abstandswert zugeordnete Remissionswert dient zur Erzeugung von Reflektionsbildern (vgl. Abbildung 4.11).

Neben dem 3D-Scanner besitzt Kurt3D noch zwei Kamerasysteme. Anhang D beschreibt das Kamarasystem und stellt die Algorithmen vor, die notwendig sind, um die aufgenommenen 3D-Daten mit Texturinformation zu versehen. Erstaunlicherweise weisen diese Algorithmen große Ähnlichkeit zu den Registrierungsmethoden des Kapitels 3 auf.

5.1.2 Die Roboterkontrolle

Das Fahren eines mobilen Roboters erfordert die Implementation zuverlässiger Algorithmen, die garantieren, dass die Sicherheit für den Roboter und seine Umgebung gewährleistet ist. Hierfür ist die Fähigkeit grundlegend, auf plötzliche Änderungen in der Umgebung, z.B. sich öffnende Türen, zu reagieren. Je höher die Geschwindigkeit des Roboters ist, desto striktere Anforderungen müssen an die Echtzeitfähigkeit gestellt werden. Die Kontrolle bzw. Regelung von Kurt3D besteht aus zwei Teilen: Einem schnellen Motorregler und einer Menge von Verhalten für die Stellgrößenberechnung.

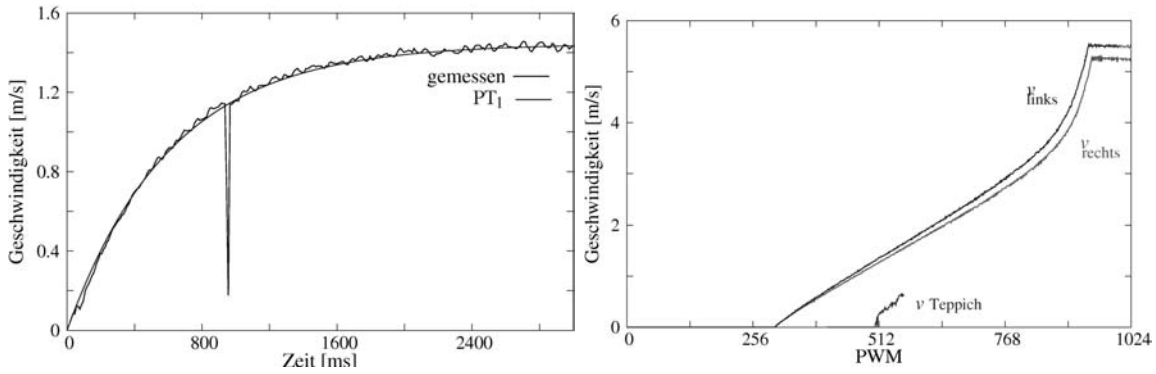


Abbildung 5.2: Links: Sprungantwort zur Maximalgeschwindigkeit (gemessen auf Teppichboden), die ein PT_1-Element approximiert (PT_1: $b\big(a(1-e^{-ax})\big)^{-1}$). Der Messfehler hat seine Ursache im Aufbau der Odometrie, die die Ticks am Motor und nicht am Rad misst. Die Sprünge treten beim Rutschen der Gummi-Kette auf. Rechts: Geschwindigkeit des linken und rechten Rades als eine Funktion der PWM-Signale gemessen im Leerlauf. Die Geschwindigkeit auf Teppichboden weist eine kostante Verschiebung auf.

Die Motorenregelung

Das Design einer Robotersteuerung für mobile Roboter bedeutet die Entwicklung von Verhalten, d.h. die Stellgrößenberechnung für die Regler. Die Regler justieren die Größen so, dass externe Störungen, beispielsweise Schwankungen der Batteriespannung oder unterschiedliche Reibungswerte, ausgeglichen werden. Auf Kurt3D läuft die Regelschleife für die Motoren mit 100 Hz ab.

Um die physikalischen Eigenschaften des Motors zu modellieren, wurden verschiedene Sprungantworten und Geschwindigkeitsmessungen im Leerlauf und auf Teppichboden mit Kurt3D aufgenommen. Abbildung 5.2 (links) zeigt die Sprungantwort auf die Maximalgeschwindigkeit, die der Charakteristik eines PT_1-Elements folgt. Der rechte Teil der Abbildung präsentiert PWM/Geschwindigkeits-Messungen im Leerlauf und auf Teppichboden. Jedes diskrete PWM-Signal (pulsweitenmodelliertes Signal) von 0 bis 1023 wird den Robotermotoren eine Sekunde lang gegeben und die resultierende Geschwindigkeit gemessen. Dabei zeigt das System Nichtlinearitäten im oberen Geschwindigkeitsbereich. Die Leerlaufmessungen und diejenigen auf Teppichboden unterscheiden sich nur um einen Verschiebungsterm. Daher betrachtet der implementierte Controller als Verallgemeinerung dieser Messungen den Verschiebungsterm im gesamten Geschwindigkeitsintervall als konstant. Es ist durch Größenbegrenzungen der Testumgebung (der Korridor ist mit 32 m nicht lang genug) nicht möglich, die komplette PWM/Geschwindigkeitstabelle auf Teppichboden aufzunehmen.

Eine schneller, zuverlässige Regelungsschleife ergibt sich durch den Einsatz eines PI-Reglers[1] mit zwei Feed-Forward-Termen (F_v, F_ω, vgl. Abbildung 5.3) [156]. PI-Algorithmen generieren einen Korrekturwert genau dann, wenn eine Abweichung vom Sollwert exsistiert. Dies impliziert jedoch eine permanente Abweichung. Feed-Forward-Terme zielen darauf ab, diese Abweichung zu minimieren, indem das zukünftige Systemverhalten abgeschätzt wird. Des Weiteren verbessert sich das zeitliche Verhalten, so dass schneller auf sich ändernde Geschwindigkeitskommandos reagiert werden kann. F_v wurde auf 1 gesetzt, so dass der Regler nur die verbleibende Abweichung minimieren muss. Ein

[1] Ein PI-Regler wurde dem PID-Regler vorgezogen, da PID-Regler oftmals Schwierigkeiten mit Messfehlern haben (vgl. Abbildung 5.2, links).

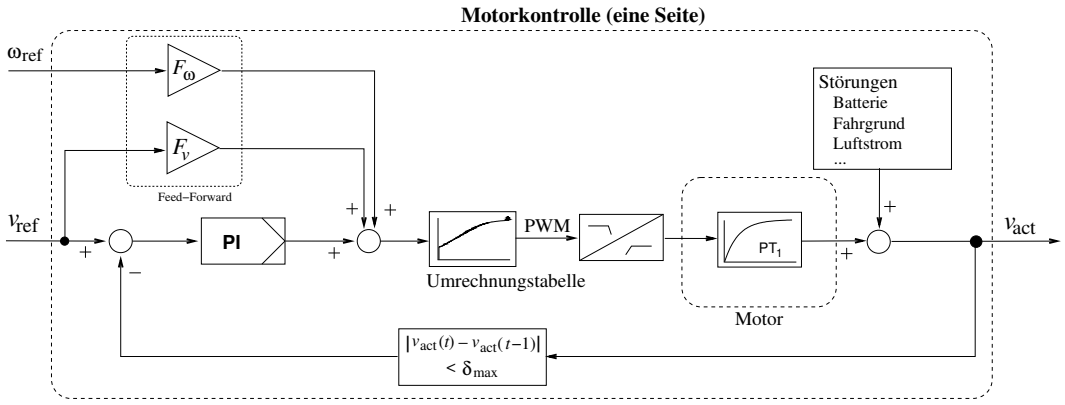


Abbildung 5.3: Der Regler für einen Motor besteht auf einem PI-Regler mit Feed-Forward-Term und Linearisierung.

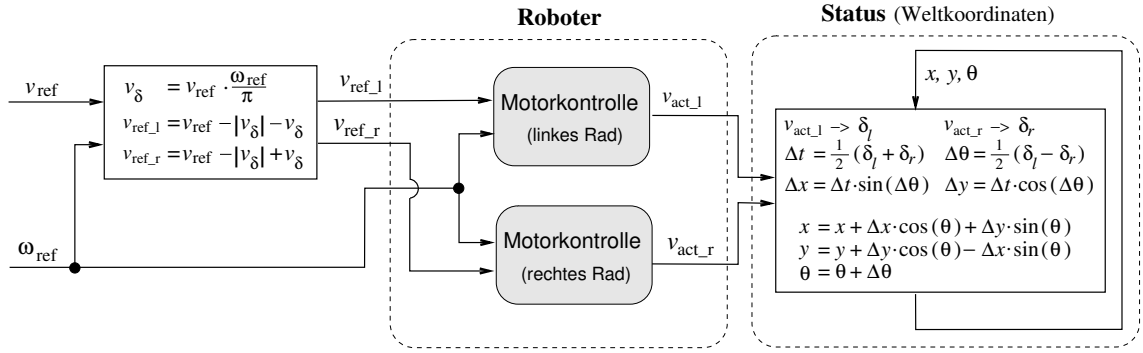


Abbildung 5.4: Schematischer Überblick über die Robotersteuerung mit den beiden Motorreglern (vgl. Abbildung 5.3 für die Regler des rechten bzw. linken Motors).

zusätzlicher Feed-Forward-Term wird für die Winkelgeschwindigkeit verwendet ($F_\omega = \pi^{-1}$ cm, vgl. Abbildung 5.3). Eine Tabelle, die die inverse Funktion aus Abbildung 5.2 (rechts) enthält, generiert die PWM-Signale von den Geschwindigkeiten. Schließlich erhalten die Motoren das tiefpassgefilterte Signal. Abbildung 5.4 zeigt das komplette System für die linken und rechten Räder mit den entsprechenden Zustandsänderungen.

Die Stellgrößenberechnung

Drei verschiedene Alternativen stehen auf dem Kurt3D Roboter zur Berechnung der Stellgrößen zur Verfügung. Je nach Anwendung kann der Roboter durch einen Operator mit Hilfe eines Joysticks teleoperiert, durch eine Fuzzy-Regelung schnell und sicher autonom gesteuert oder durch eine global stabile Regelung positioniert werden.

Teleoperation. Für die Teleoperation wird ein Joystick benutzt. Die Joysticksignale werden direkt auf die Geschwindigkeiten v_{ref} und ω_{ref} abgebildet. Allerdings ist es sehr schwer, Kurt3D manuell schneller als 1 m/s zu steuern.

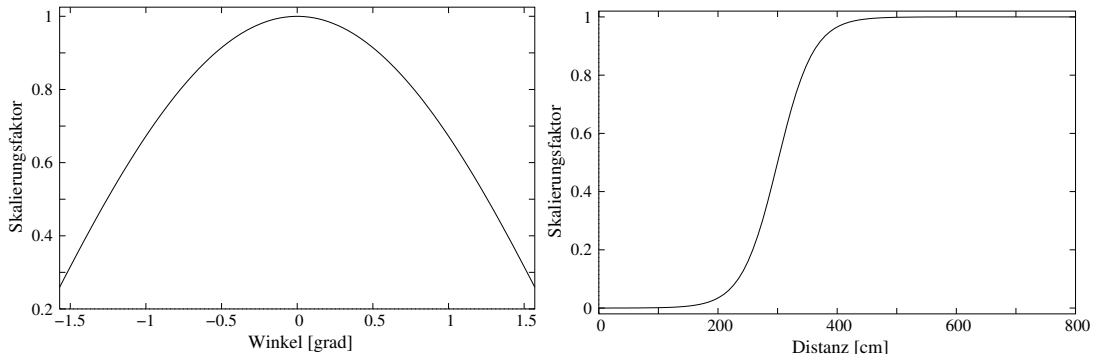


Abbildung 5.5: Stellgrößenberechnung mit Fuzzy-Regeln (vgl. Gleichungen (5.1), (5.2)). Links: Die Funktion f_1 gewichtet die Orientierung ("`ist in Fahrtrichtung`"). Rechts: f_2 implementiert die Gewichtung der Abstandswerte ("`ist groß`").

Fuzzy-Regelung. Um die Probleme eines menschlichen Operators zu überwinden, gibt es eine Fuzzy-Regelung, die den Roboter bei voller Geschwindigkeit steuern kann. Es wird immer freier Raum angesteuert, wobei zusätzlich die zu steuernde Richtung relativ stabil bleibt, damit die Trajektorie keine Oszillation aufweist. Eine Fuzzy-Regel in 181 Ausprägungen (vgl. Formel (5.1)) bestimmt die zu fahrende Richtung. Die i-te Regel wendet der Regler auf den i-ten Abstandswert an.

```
IF (Winkel_i ist in Fahrtrichtung) AND
   (Entfernung_i ist groß)                    (5.1)
THEN fahre in diese Richtung.
```

Das Fuzzy-UND ist als Multiplikation implementiert. Die zu fahrende Richtung α ist das Ergebnis einer Addition aller i Richtungsvektoren; d.h. aus den Messwerten $\{(\phi_i, r_i)\}_{i=1,\ldots,181}$ wird α durch

$$\alpha = \text{ATAN2}\left(\sum_{i=1}^{181} \sin(\phi_i) \cdot f_1(\phi_i) \cdot f_2(r_i), \sum_{i=1}^{181} \cos(\phi_i) \cdot f_1(\phi_i) \cdot f_2(r_i)\right) \tag{5.2}$$

berechnet. Abbildung 5.5 zeigt die dabei verwendeten Funktionen f_1 und f_2, die die Gewichtung der Orientierung bzw. Distanz durch die Fuzzy-Regeln (5.1) bestimmen. Die Drehgeschwindigkeit ω_{ref} für die Motorenregelung ist direkt proportional zu α.

Um bei hohen Geschwindigkeiten sicher fahren zu können, wendet Kurt3D folgenden Algorithmus für die Bestimmung der zu fahrenden Geschwindigkeit an: Eine virtuelle Straße definiert sich in Anlehnung an Kurts Breite (vgl. Abbildung 5.6). Falls sich auf dieser Straße kein Hindernis vor dem Roboter befindet, wird die Stellgröße v_{ref} auf $v_{\max}$ gesetzt. Im Fall, dass ein Hindernis näher als $dto_{\max}$ zu Kurt detektiert wird, skaliert sich die Geschwindigkeit durch $v_{\text{ref}} = dto/dto_{\max}\, xv_{\max}$, wobei dto die gemessene Distanz zum Hindernis ist. Nun steuert die Fuzzy-Regelung Kurt3D um Hindernisse. Dennoch kann es passieren, dass dto unter eine definierte Schwelle fällt. In diesem Fall wird v_{ref} auf 0 gesetzt und ω_{ref} erhält einen konstanten Wert. Dadurch dreht sich Kurt3D vom Hindernis weg, bis die virtuelle Straße wieder frei ist. Für diese Berechnung gelten die Konstanten: $dto_{\min} = 50$ cm, $dto_{\max} = 600$ cm, mit einer maximalen Geschwindigkeit $v_{\max}$ von 4 m/s.

Positionsregelung. Die koordinatenbasierte Regelung findet Anwendung, um eine diskrete Pose anzufahren, wie sie beispielsweise von einem Planungsalgorithmus generiert wird [194]. Bei dieser

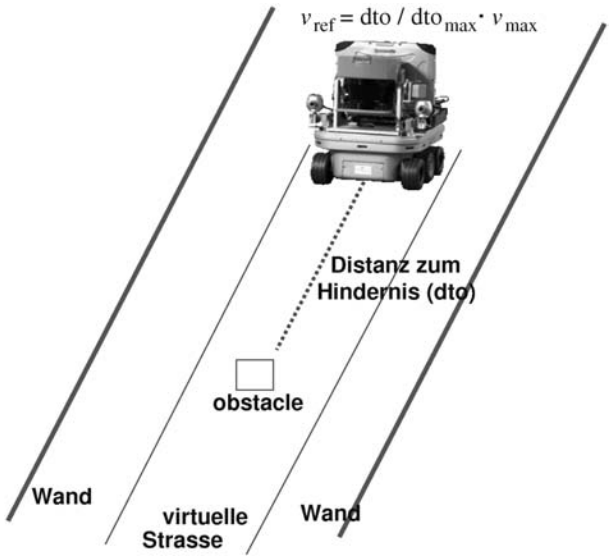


Abbildung 5.6: Um sicher zu navigieren, wird die Stellgröße Geschwindigkeit v_{ref} als Funktion des Abstandes eines Hindernisses auf einer virtuellen Straße berechnet.

Steuerungsmethode wird Kurt3D durch eine geschlossene, zeitinvariante und global-stabile Regelung betrieben, die von G. Indiveri [112] entwickelt worden ist. Der Roboter erreicht die Zielpose immer auf einer geraden Linie, d.h. die Krümmung c konvergiert im Ziel gegen 0. Die Bewegung des Roboters ist immer eine Vorwärtsbewegung und die Trajektorie ist differenzierbar. Der Notation aus [194] folgend, bezeichne (x_G, y_G, ϕ) die Roboterpose in einem zielzentrierten Koordinatensystem. Der Regler basiert auf einem kartesischen kinematischen Modell, das durch

$$\dot{x}_G = v_{\text{ref}} \cdot \cos\phi \qquad \dot{y}_G = v_{\text{ref}} \cdot \sin\phi \qquad \dot{\phi} = \omega_{\text{ref}} = v_{\text{ref}} \cdot c$$

beschrieben wird. Hierbei ist v_{ref} die Bahngeschwindigkeit des Roboters, ω_{ref} die Drehgeschwindigkeit, c die (begrenzte) Krümmung und $(0, 0, 0)$ die Zielposition. Die Transformation der kartesischen Koordinaten in eine Polardarstellung führt zu:

$$\begin{aligned} e &= \sqrt{{x_G}^2 + {y_G}^2} & \dot{e} &= -v_{\text{ref}} \cdot \cos\alpha \\ \theta &= \text{ATAN2}(-y_G, -x_G) \quad \rightarrow & \dot{\alpha} &= v_{\text{ref}} \left(c - \frac{\sin\alpha}{e} \right) \\ \alpha &= \theta - \phi & \dot{\phi} &= v_{\text{ref}} \cdot \frac{\sin\alpha}{e}. \end{aligned}$$

G. Indiveri benutzt für die Berechnung der Robotergeschwindigkeit die Formel

$$v_{\text{ref}} = \gamma e \qquad \text{mit} \qquad \gamma > 0$$

und einen Lyapunov-Ansatz zur Herleitung der Regelgleichung für die Krümmung:

$$c = \frac{\sin\alpha}{e} + h\frac{\theta}{e} \cdot \frac{\sin\alpha}{\alpha} + \beta\frac{\alpha}{e},$$

mit $h > 1$ und $2 < \beta < h + 1$ [112]. Diese beiden Formeln für die Geschwindigkeit und Krümmung (bzw. Winkelgeschwindigkeit) stellen die geschlossene Positionsregelung von Kurt3D dar. Die Auswertung des Regelgesetzes geschieht mit der Zeitkomplexität $\mathcal{O}(1)$.

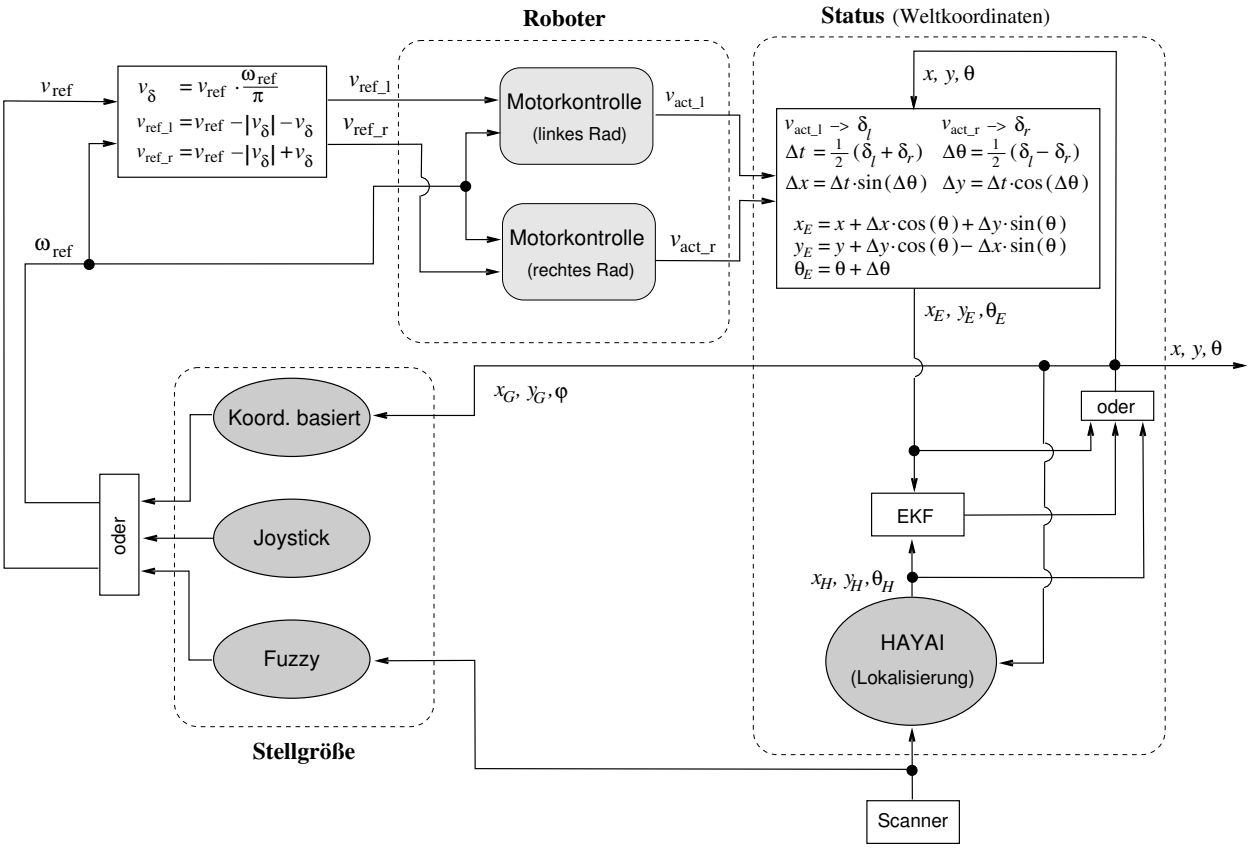


Abbildung 5.7: Überblick über das Kontrollsystem von Kurt3D. Die Erweiterung der Abbildung 5.4 durch verschiedene Stellgrößenberechnungen und durch HAYAI als Lokalisierungsmethode, um die Pose des Roboters zu verfolgen.

Abbildung 5.7 gibt einen Überblick über das komplette Steuerungssystem. Der darin aufgeführte Lokalisierungsalgorithmus wird kurz im folgenden Abschnitt beschrieben: Details finden sich in [129, 131].

5.1.3 Positionstracking mit HAYAI

Der HAYAI-Algorithmus (engl.: *Highspeed And Yet Accurate Indoor/outdoor-tracking*) wurde von K. Lingemann entwickelt. Das Verfahren ist in der Lage, die Position eines mobilen Roboters mit Hilfe eines 2D-Laserscanners bei einer Geschwindigkeit von 4 m/s zu verfolgen [127,129,131]. HAYAI ist ein Matching Algorithmus, der nach folgendem Muster arbeitet:

1. Detektiere die Merkmalmenge M innerhalb des aktuellen 2D-Scans $\mathcal{R}$. Ebenso bestimme die Merkmalmenge D in einem vorausgegangenen Referenzscan $\mathcal{S}$.
2. Suche nach einer paarweisen Korrespondenz zwischen Merkmalen der beiden Mengen und entferne nicht zugeordnete Merkmale aus M und D.
3. Berechne die Verschiebung der Pose $\Delta\mathbf{p} = (\Delta x, \Delta y, \Delta\theta)^T$, so dass die Menge M optimal auf D abgebildet wird.
4. Aktualisiere die Schätzung Roboterpose $\mathbf{p}_n \xrightarrow[\Delta\mathbf{p}]{} \mathbf{p}_{n+1}$.
5. Setze den aktuellen 2D-Scan als neuen Referenzscan $\mathcal{R} \leftarrow \mathcal{S}$.

Bei gegebener Roboterpose $\mathbf{p}_n = (x_n, y_n, \theta_n)$ und einer Verschiebung, d.h. Transformation $\Delta\mathbf{p} = (\Delta x, \Delta y, \Delta\theta)$, lautet die Berechnung der neuen Position $\mathbf{p}_n \xrightarrow[\Delta\mathbf{p}]{} \mathbf{p}_{n+1}$ wie folgt:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ \theta_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \\ \theta_n \end{pmatrix} + \begin{pmatrix} \cos\theta_n & \sin\theta_n & 0 \\ -\sin\theta_n & \cos\theta_n & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{pmatrix}. \tag{5.3}$$

Die Bestimmung der Merkmale und des Matchings

Der Scanmatching Algorithmus berechnet die Transformation $\Delta\mathbf{p}$, so dass zwei Mengen von Merkmalen, die von dem aktuellen und dem vorangegangenen 2D-Scan extrahiert wurden, optimal aufeinander abgebildet werden. Um Merkmale für das Nachführen der Pose zu verwenden, müssen sie erstens *invariant* bezüglich von Rotation und Translation sein und zweitens effizient *berechnen* lassen, damit die Echtzeitbedingung erfüllt wird.

Die Laserscandaten werden durch den Drehspiegel in einer sortierten Reihenfolge erzeugt. Dies erlaubt die Anwendung linearer Filter. HAYAI wählt Extrema in der Polardarstellung der 2D-Scans als Merkmale aus. Diese entsprechen Stellen, an denen sich die Umgebungsstruktur ändert, also natürlichen Landmarken. Beispielsweise korrelieren Ecken mit Extrema. Das Verwenden der Polardarstellung impliziert eine Reduktion um eine Dimension. Die linearen Filter werden auf eine Sequenz von Abstandswerten $(r_i)_{i\in\mathbb{N}}$ eines 2D-Scans $\mathcal{S} = \big((\phi_i, r_i)\big)_{i=1,\ldots,N}$ angewandt.

Das Ergebnis r_i^Ψ eines eindimensionalen Filters $\Psi = \begin{bmatrix} \psi_{-1}, & \psi_0, & \psi_{+1} \end{bmatrix}$ des Scanpunktes r_i ($i = 2, \ldots, N-1$) ist definiert als $r_i^\Psi = \sum_{k=-1}^{1} \psi_k r_{i+k}$. Zur Detektion eines Merkmals wird das Eingabesignal wie folgt gefiltert:

1. Schärfe das Signal, um die signifikanten Bereiche des 2D-Scans, d.h. Extrema, hervorzuheben. Verwende dazu den Filter $\Psi_1 = \big[-1,\ 4,\ -1\big]$.
2. Berechne die Ableitung des Signals durch die Anwendung des Gradientenfilters $\Psi_2 = \big[-\frac{1}{2},\ 0,\ \frac{1}{2}\big]$.
3. Glätte das Signal, um die Erkennung der Nulldurchgänge zu erleichtern. Verwende dabei den Weichzeichnungsfilter $\Psi_3 = \big[1,\ 1,\ 1\big]$.

Abbildung 5.8 (oben links) zeigt die Effekte der verwendeten Filter.

Nachdem HAYAI die Mengen M und D von beiden 2D-Scans erzeugt hat, muss eine Zuordnung der Merkmale berechnet werden. Statt der Lösung des harten Optimierungsproblems der optimalen Zuordnung wird ein heuristisches Verfahren verwendet. Dieses benutzt implizites Wissen über das Problem, z.B. dass sich die grundlegende Topologie der Merkmale zwischen zwei Scans nicht ändert. Das Ziel ist, eine Matrix möglicher Paarungen zu erstellen, die auf den Distanzen zwischen zwei Punkten $\mathbf{m}_i, \mathbf{d}_j$ basiert, mit $\mathbf{m}_i = (m_{i,x}, m_{i,y})^T$ in kartesischen und $\mathbf{m}_i = (m_{i,\phi}, m_{i,r})^T$ in Polarkoordinaten ($\mathbf{d}_i$ analog):

$$\begin{aligned}\operatorname{dist}\big(\mathbf{m}_i, \mathbf{d}_j\big) = &\sqrt{\big(\omega_1 \cdot (m_{i,\phi} - d_{j,\phi})\big)^2 + \omega_2 (m_{i,r} - d_{j,r})^2} \\ &+ \omega_3 \cdot \sqrt{(m_{i,x} - d_{j,x})^2 + (m_{i,y} - d_{j,y})^2} \\ &+ \Theta\big(\mathbf{m}_i, \mathbf{d}_j\big). \end{aligned} \tag{5.4}$$

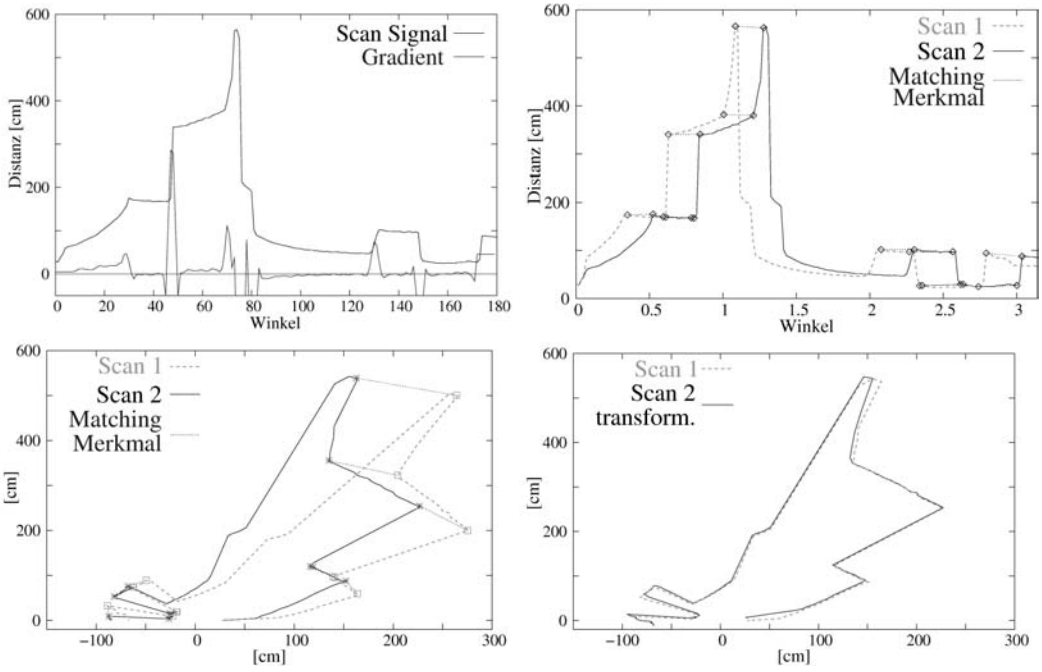


Abbildung 5.8: Merkmalsextraktion und Zuordnung bei HAYAI. Von links oben nach rechts: (1) Anwendung der Filter zur Merkmalsextraktion. (2) und (3) Zuordnung korrespondierender Merkmale, (ϕ, r) Repräsentation (2) vs. euklidische Repräsentation (3). (4) Transformierter 2D-Scan.

Die Konstanten $(\omega_k)_{k\in\{1,2,3\}}$ implementieren die Gewichtung zwischen polaren und kartesischen Koordinaten. Die Funktion Θ verhindert das Matching von Merkmalen unterschiedlichen Typs:

$$\Theta\big(\mathbf{m}_i, \mathbf{d}_j\big) = \begin{cases} 0 & \Gamma(\mathbf{m}_i) = \Gamma(\mathbf{d}_j) \\ \infty & \text{else} \end{cases}$$

mit der Klassifikationsfunktion $\Gamma\colon (M \cup D) \mapsto \{\text{max.}, \text{min.}, \text{Wendepunkt}\}$.

Die resultierende Matrix $w_{i,j}$ bezeichnet die Merkmalskorrespondenzen und wird solange vereinfacht, bis Eindeutigkeit entsteht [127, 129]. Abbildung 5.8 zeigt das HAYAI-Matching zweier 2D-Scans.

Aktualisierung der Poseschätzung

Gegeben seien nun zwei Mengen von Merkmalen: $M = \left\{ \mathbf{m}_i \;\middle|\; \mathbf{m}_i \in \mathbb{R}^2,\ i = 1, \ldots, N_m \right\}$ und $D = \left\{ \mathbf{d}_i \;\middle|\; \mathbf{d}_i \in \mathbb{R}^2,\ i = 1, \ldots, N_d \right\}$. Die Berechnung der optimalen Transformation ΔP für die Abbildung der Menge D nach M führt dazu, dass folgende Fehlerfunktion minimiert werden muss:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t})\right|\right|^2 .$$

Diese Gleichung entspricht genau der Fehlerfunktion (3.1) in Kapitel 3.1 (Seite 12). Der einzige Unterschied besteht darin, dass die Vektoren zweidimensional sind (im Gegensatz zu Kapitel 3.1, wo es sich um 3D-Vektoren handelt). Aus den gleichen Argumenten wie in Kapitel 3.1 folgt, dass die Fehlerfunktion umgeschrieben werden darf:

$$E(\mathbf{R}, \mathbf{t}) \propto \frac{1}{N} \sum_{i=1}^{N} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t})\right|\right|^2 .$$

Es entsteht das Äquivalent zu Gleichung (3.2). Nun wird analog zu der Argumentation und den Gleichungen (3.3), (3.4), (3.5) und (3.6) obiges Ergebnis zu Gleichung (3.7) umgeformt:

$$E(\mathbf{R}) \propto \sum_{i=1}^{N} \left|\left|\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i\right|\right|^2 .$$

Jetzt müssen die Lösungsmethoden aus Kapitel 3.1 für den 2D-Fall angepasst werden. Durch das Lösen der Gleichung $\frac{\partial}{\partial \Delta\theta} E(\mathbf{R}_{\Delta\theta}) = 0$ für eine 2D-Rotation $\mathbf{R}_{\Delta\theta} = \mathbf{R}$ ergibt sich die optimale Rotation:

$$\Delta\theta = \arctan \left(\frac{\sum_{i=1}^{N} \left(m'^x_i d'^x_i + m'^y_i d'^y_i \right)}{\sum_{i=1}^{N} \left(m'^y_i d'^x_i - m'^x_i d'^y_i \right)} \right) .$$

Mit Hilfe der Rotation lässt sich die Translation wiederum analog zu der Gleichung (3.5) bestimmen, d.h.

$$\underbrace{\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}}_{=\Delta\mathbf{t}} = \mathbf{c}_m - \underbrace{\begin{pmatrix} \cos\Delta\theta & \sin\Delta\theta \\ -\sin\Delta\theta & \cos\Delta\theta \end{pmatrix}}_{=\mathbf{R}_{\Delta\theta}} \cdot \mathbf{c}_d .$$

5.2 Anwendung: Kurt3D im RoboCup Rescue Wettbewerb

RoboCup ist eine internationale Initiative zur Förderung der Forschung in den interdisziplinären Bereichen „Künstliche Intelligenz“, „Mechatronik“ und „Autonome Mobile Roboter“. In einem Wettbewerb treten verschiedene Teams mit ihren Robotern gegeneinander an. Neben unterschiedlichen Fußballliegen gibt es den Rescue-Wettbewerb. Darin suchen ferngelenkte Roboter in einer zerstörten Umgebung nach Opfern. Eine weitere Aufgabe ist hierbei das Erstellen einer Karte, nach der die Opfer geborgen werden können. Der Wettkampf findet in drei verschiedenen Schwierigkeitsstufen, die als gelbe, orangene und rote Arena benannt sind (vgl. Abbildung 5.9), statt. Die Arenen werden nach jeder Runde in der so genannten Earthquake-Phase vollkommen neu errichtet. Anschließend installieren Mitglieder der Jury an beliebigen Stellen Opferpuppen, die menschliche Merkmale aufweisen. Der Operator selbst sitzt während der Mission in einiger Entfernung zur eigentlichen Arena an seinem Arbeitsplatz in einem vollständig abgeschotteten Raum.

Abbildung 5.9: RoboCup 2004 Arenen. Links: Gelbe Arena. Mitte: Orange Arena. Rechts: Rote Arena. Quelle: [155].

Der RoboCup Rescue Wettbewerb ist als eine Evaluationsform für die in dieser Arbeit vorgestellten Kartierungs-Algorithmen geeignet. Fernziel ist es, einen mobilen Roboter zu entwickeln, der in der Lage ist, in realen Katastrophengebieten zu fahren und die Rettungshelfer bei der Arbeit zu unterstützen. Existierende Systeme sind zur Zeit für diesen Zweck mangels zuverlässiger Lokomotion sowie wegen Sensor- und Kartierungsproblemen noch ungeeignet, wie nicht zuletzt die Erfahrungen am World-Trade-Center in New York zeigten [140, 141].

Abbildung 5.10: Kurt3D als Rettungsroboter, wie er beim Wettbewerb in Lissabon 2004 präsentiert worden ist.

Der Roboter Kurt3D mit der 3D-Kartierung und seinen Steueralgorithmen wurde seit Sommer 2004 für den Wettbewerb weiterentwickelt. Abbildung 5.10 zeigt die Rettungsroboterversion. Im Vergleich zu Abbildung 5.1 links besitzt er größere Räder, die die Bodenfreiheit erhöhen, damit der Roboter leichter kleinere Hindernisse überwinden kann. Die Übersetzung der Motoren wurde geändert, so dass statt Geschwindigkeit mehr Kraft zur Verfügung steht. Zusätzlich besitzt er eine Beleuchtungseinrichtung, bestehend aus Neonröhren und 2×4 Leuchtdioden, die an den schwenkbaren Kameras angebracht sind, und einen zweiten Laptop

als WLAN-Gateway. Gesteuert wird Kurt3D durch einen Operator, der auch in der 3D-Karte die Opferpuppen markiert hat.

Während des Wettbewerbs 2004 erzeugte Kurt3D verschiedene 3D-Karten. Abbildung 5.11 zeigt eine 3D-Karte der orangenen Arena von oben als Parallelprojektion, Abbildung 5.12 3D-Ansichten. Die Abbildungen 5.13 und 5.14 zeigen 3D-Karten einer gelben Arena.

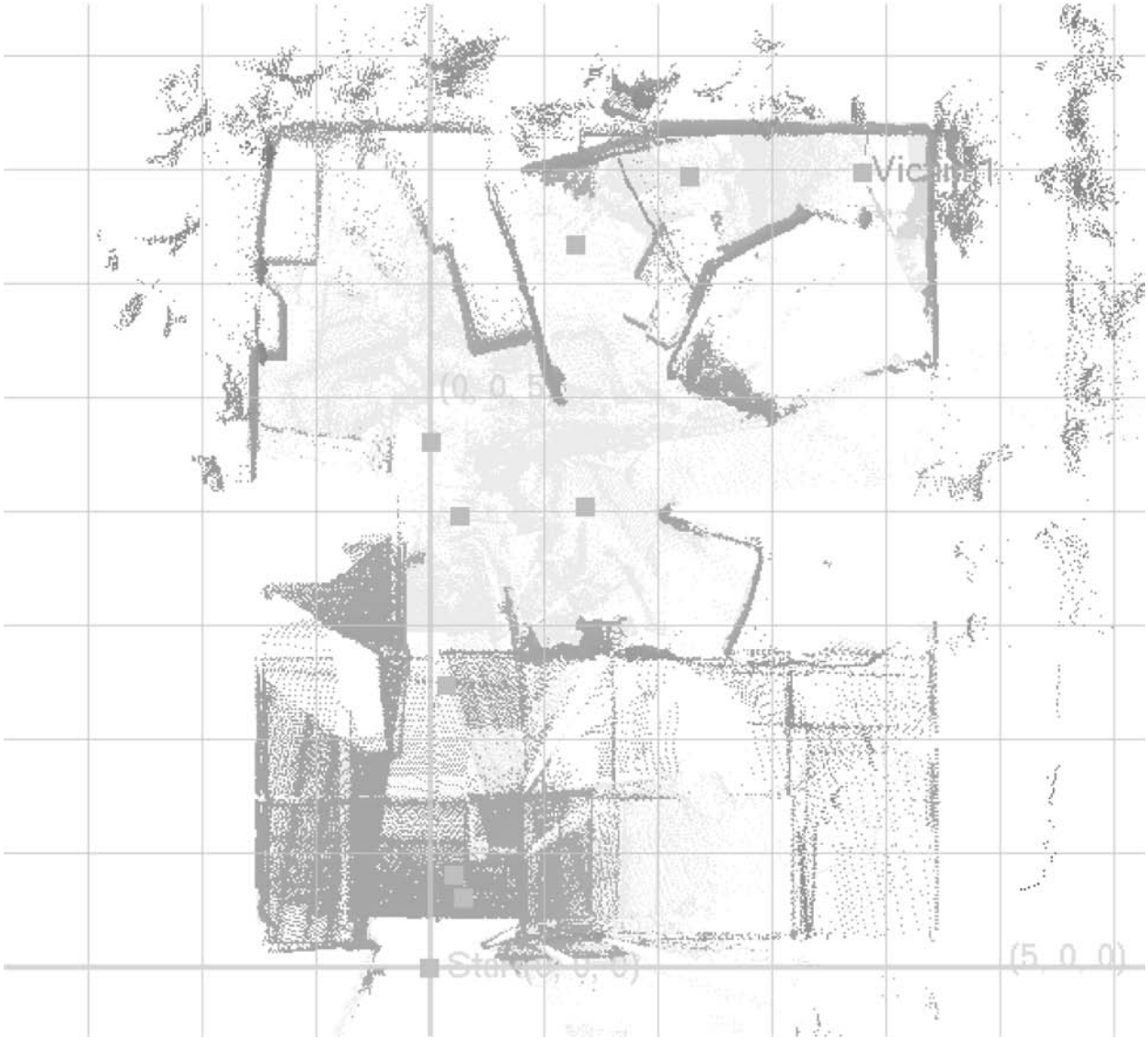


Abbildung 5.11: Eine 3D-Karte der orangenen Arena, dargestellt von oben in Parallelprojektion. Punkte auf dem Boden sind hellgrau dargestellt. Die Positionen der 3D-Scans (graue Quadrate) und der gefundenen Opfer (hellgrau) sind eingezeichnet. Die Rasterquadrate haben eine Größe von 1 m^2.

Für die 3D-Kartierung wurden ausschließlich die Scanmatching-Algorithmen mit ihren Beschleunigungen angewandt. Somit wurde auf eine globale Optimierung, wie auf Seite 41 beschrieben, verzichtet.

Abbildung 5.12: 3D-Ansichten der orangenen Arena, vgl. Abbildung 5.11. Die Projektionskamera ist leicht nach oben verschoben. In die linke 3D-Karte wurde ein Koordinatensystem mit einem 1 m^2 großen Gitter eingezeichnet.

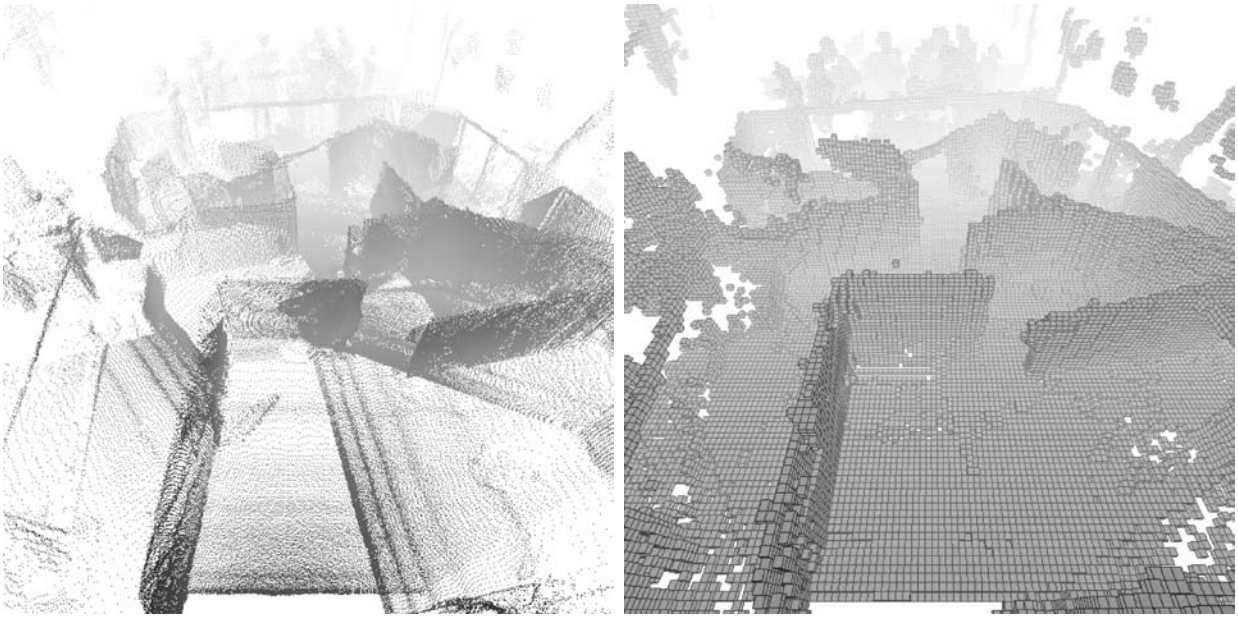


Abbildung 5.13: 3D-Ansichten der gelben Arena, vgl. Abbildung 5.14. Die Projektionskamera ist leicht nach oben verschoben. Links: Punktwolke. Rechts: Darstellung als Octreemodell (vgl. [144]).

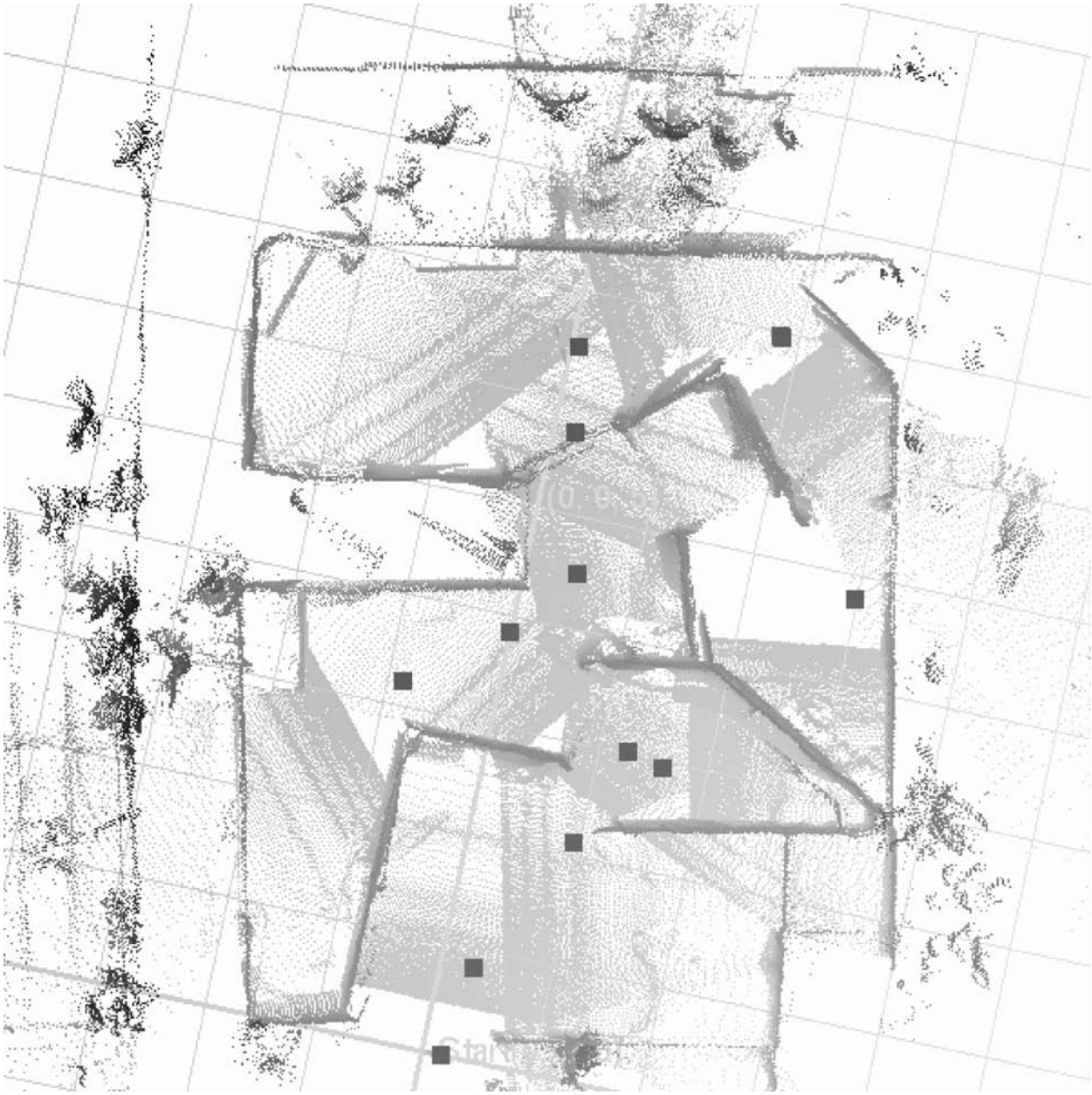


Abbildung 5.14: Eine 3D-Karte der gelben Arena, dargestellt als Draufsicht in Parallelprojektion.

5.3 Semantische 3D-Karten

5.3.1 Das Erstellen semantischer 3D-Karten

Ziel der vorliegenden Arbeit ist es, semantische Karten mit Hilfe des Roboters Kurt3D, der mit einem 3D-Laserscanner ausgestattet ist, zu erzeugen. Dreidimensionale semantische Karten enthalten neben Geometrieinformationen auch Informationen über Objekte. Die Karten können mittels einem `OpenGL`-Renderingprogramms aus jeder Perspektive betrachtet werden. Abbildung 5.15 zeigt das Benutzerinterface zum Betrachten der semantischen 3D-Karten. Damit ist es möglich, die 3D-Szene von virtuellen Kameraposen aus zu rendern, verschiedene Visualisierungstechniken zu wählen sowie Informationen über Objekte abzurufen. Die dargestellten Informationen über die Objekte bestehen aus:

- Objektinformationen, d.h., dass der semantische Bezeichner des ausgewählten Objekts angezeigt wird. Zusätzlich gibt das Programm die Lage der Objekte im Raum an, also die 6D-Pose $(x, y, z, \theta_x, \theta_y, \theta_z) \in \mathbb{R}^3$.
- 3D-Koordinaten, d.h. $(x, y, z) \in \mathbb{R}^3$ sowie Indexinformationen der einzelnen Objektpunkte. Der Index gibt an, welcher Messpunkt aus welchem 3D-Scan ausgewählt wurde.
- Entfernungen zur aktuellen Betrachtungspose,
- Angaben zur 6D-Pose, an denen ein 3D-Scan aufgenommen wurde,
- Daten über die Trajektorie, die der Roboter zurückgelegt hat, sowie
- Texturinformationen, die optional mit den Kameras aufgenommen wurden.

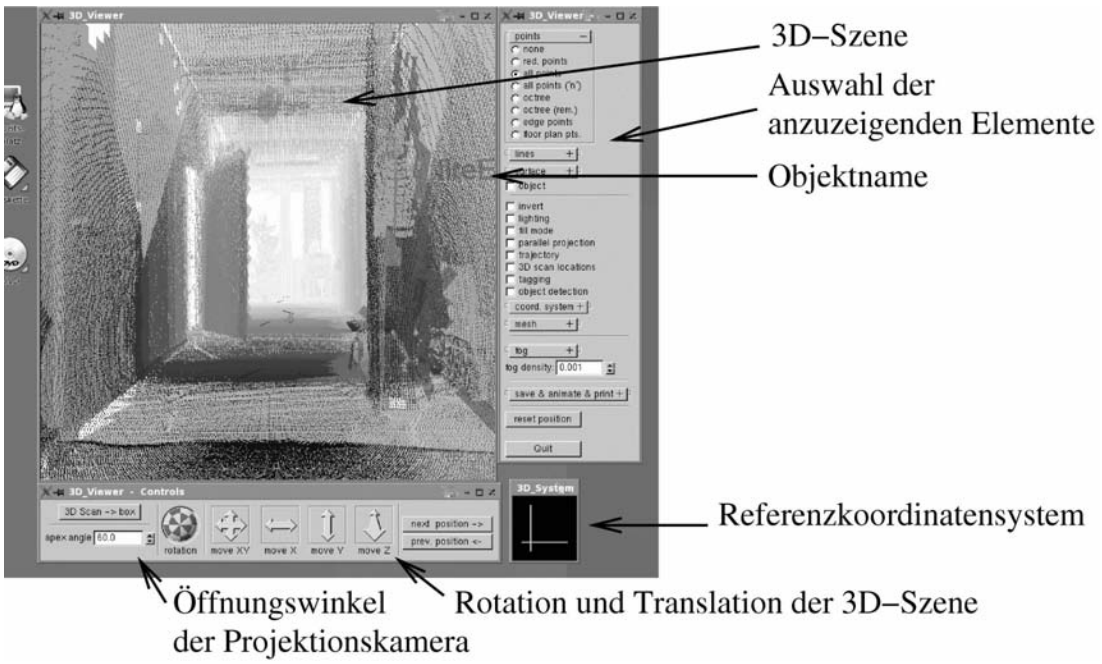


Abbildung 5.15: Das Interface zum 3D-Viewer.

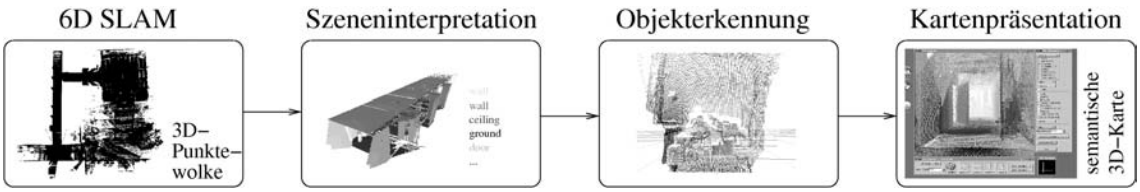


Abbildung 5.16: Das System zum Erstellen semantischer 3D-Karten.

Das Ergebnis des in dieser Arbeit entwickelten 6D SLAM ist eine 3D-Punktwolke, die aus registrierten 3D-Scans besteht (vgl. Kapitel 3). Im Anschluss an die Aufnahme der 3D-Karte erfolgt eine Interpretation der Grobstruktur, d.h. eine Klassifikation des Bodens, der Wände, der Decke und der offenen Türen (vgl. Abschnitt 4.2). Hierbei können in der Regel bereits sehr viele 3D-Punkte diesen Grobstrukturen zugeordnet werden. Die verbleibenden Punkte gehören zu Objekten, die die Objekterkennung im Anschluss detektiert. Detektierbare Objekte sind hier beispielsweise Stühle, Menschen, Pflanzen oder Feuerlöscher. Der Algorithmus zum Erstellen einer semantischen Karte lautet demnach wie folgt (vgl. Abbildung 5.16):

1. Erstellen einer 3D-Karte. Zwei Teilschritte sind dazu notwendig:
   (a) Aufnahme der 3D-Scans durch den mobilen Roboter Kurt3D (teleoperiert).
   (b) Zusammenfügen der 3D-Scans zu einer globalen Karte (6D SLAM).
2. Interpretation der 3D-Kartengrobstruktur. 3D-Flächen werden aus der 3D-Punktmenge extrahiert und bezüglich ihrer Lage benannt.
3. Extraktion von Objekten aus den 3D-Scans mit Hilfe von Klassifikatoren und anschließender Einpassung der Objekte. Hierbei wird die 6D-Pose der Objekte bestimmt.
4. Visualisierung der semantischen 3D-Karte.

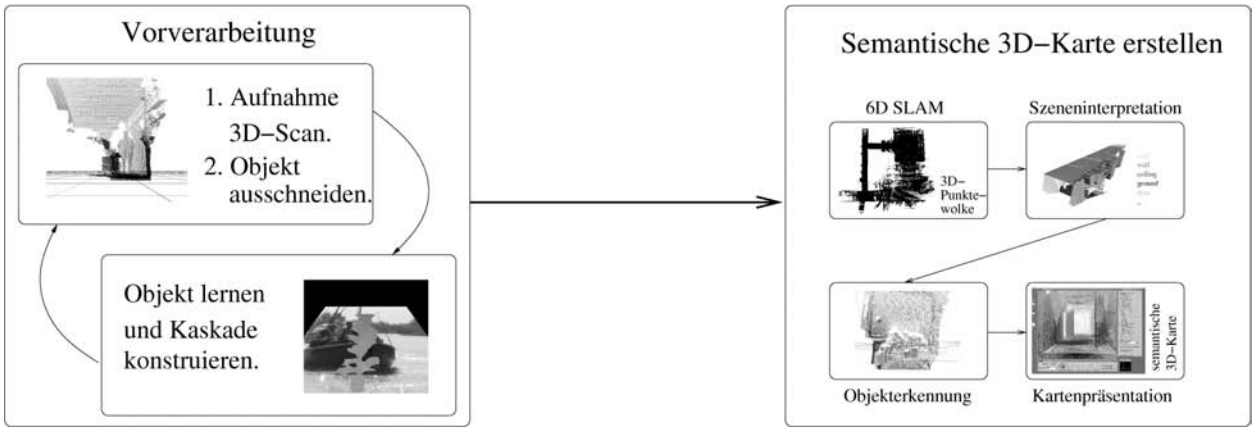


Abbildung 5.17: Zweischrittige semantische 3D-Kartierung: Zunächst werden 3D-Objekte gelernt. Anschließend wird die 3D-Karte erstellt, die Szene interpretiert und die Objekte erkannt (vgl. Abbildung 5.16).

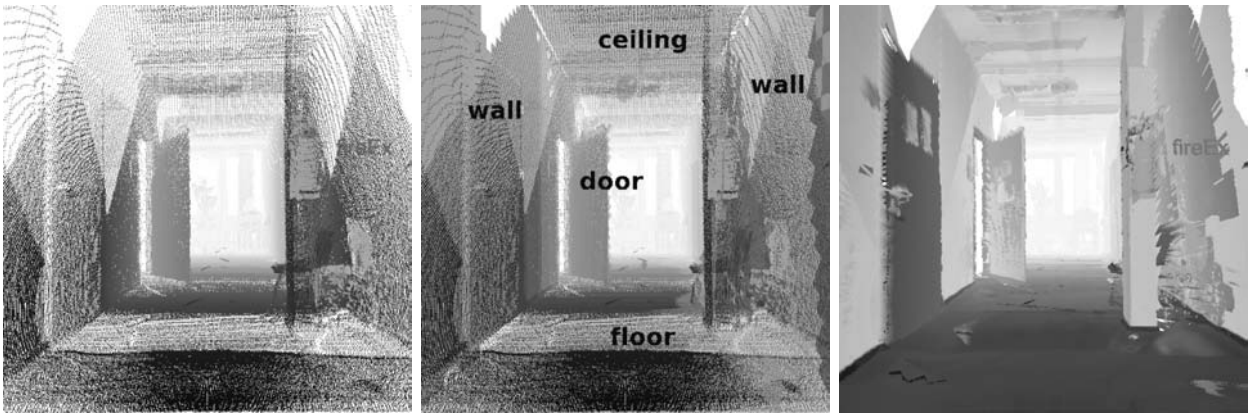


Abbildung 5.18: Semantische 3D-Karte des AVZ-Gebäudes an der Universität Osnabrück. Der Ausschnitt zeigt einen Büroflur mit einer offenen Tür und einem erkannten Objekt (Feuerlöscher, Label: „fireEx“). Links: 3D-Punktwolke. Mitte: Szeneninterpretation mit Hilfe extrahierter Ebenen. Rechts: Gitterdarstellung mit Reflektionswerten.

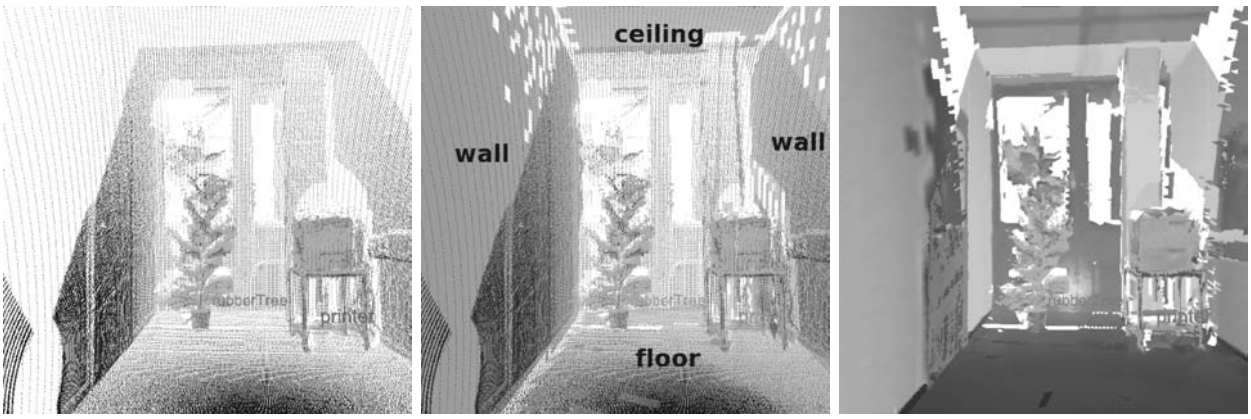


Abbildung 5.19: Semantische 3D-Karte (Forts.) mit den Objekten „printer“ und „rubbertree“.

Das Verfahren zum Erstellen einer semantischen Karte benötigt außerdem eine Initialisierungsphase. Hier nimmt der Operator 3D-Scans von Objekten auf, die später wieder erkannt werden sollen. Nach der Aufnahme der Scans markiert der Operator im Anzeigemodul diejenigen Punkte, die zum Objekt gehören, und startet den automatischen Lernprozess. Das Ergebnis sind zwei Kaskaden zur Objektdetektion sowie eine erweiterte Datenbasis mit Objekt. Abbildung 5.17 zeigt das Gesamtsystem zur Erstellung von semantischen 3D-Karten.

Die Abbildungen 5.18, 5.19, 5.20, 5.21, 5.22 und 5.23 zeigen Ausschnitte einer semantischen 3D-Karte. Als Testumgebung diente dabei das 5. Stockwerk des AVZ-Gebäudes der Universität Osnabrück. Zu sehen sind jeweils die 3D-Punktwolke, die extrahierten 3D-Flächen und deren Beschriftung sowie eine Darstellung als Gittermodell, wobei die einzelnen Dreiecke mit den Reflektionswerten eingefärbt wurden. Abbildung 5.18 gibt den Ausschnitt der semantischen 3D-Karte mit erkanntem Feuerlöscher (Objekt „fireEx“) wieder, Abbildung 5.19 zeigt einen Teil mit den Objekten „printer“ und „rubbertree“ zeigt. In Bild 5.20 ist zu sehen, dass der Stuhl zwar erkannt wurde, jedoch die Objekteinpassung fehlschlug. Die folgenden Darstellungen präsentieren im Wesentlichen die erkannten Objekte „bottles“, „plant“, „projector“ und „human“. Die vollständige semantische 3D-Karte befindet sich als Animation unter `http://www.informatik.uni-osnabrueck.de/nuechter/videos.html`

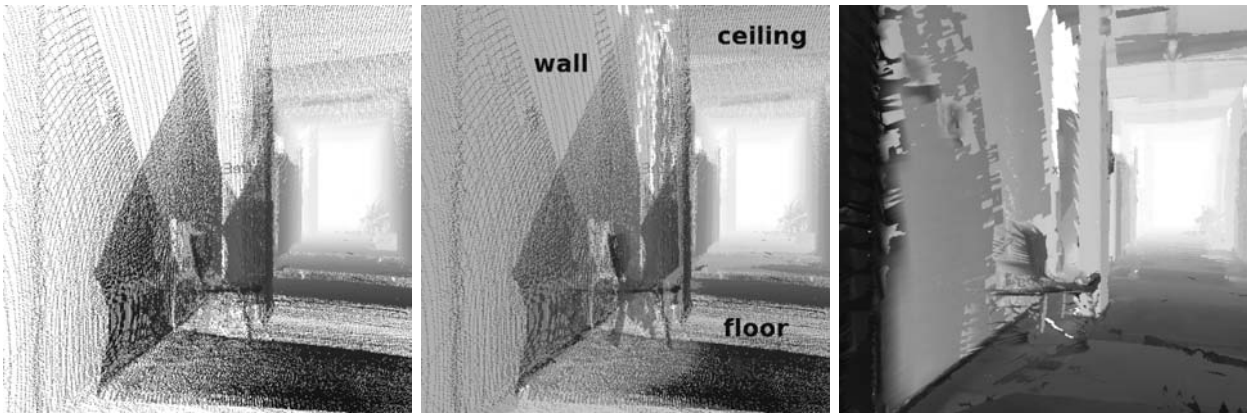


Abbildung 5.20: Semantische 3D-Karte (Forts.) mit dem Objekt „chair“. Das Objekt wurde korrekt erkannt, aber die Objekteinpassung schlug fehl.

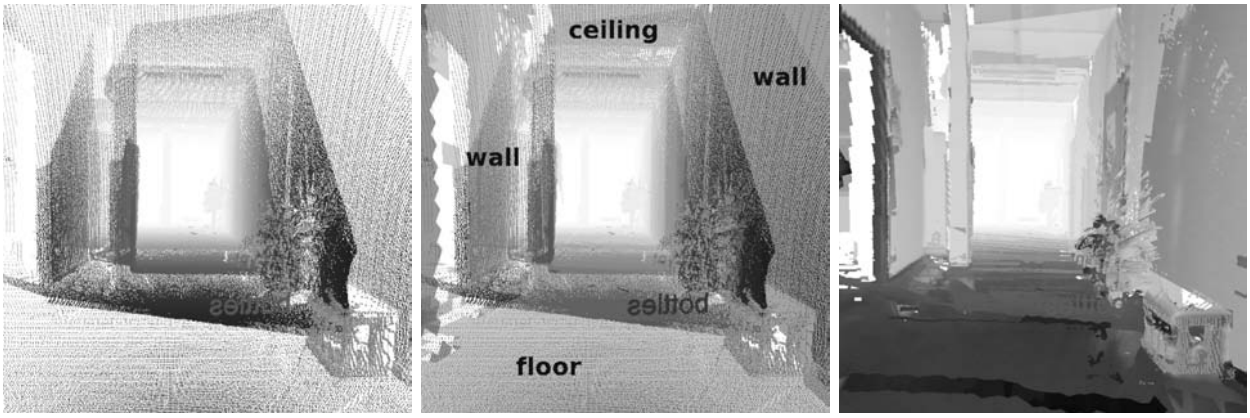


Abbildung 5.21: Semantische 3D-Karte (Forts.) mit den Objekten „bottles“ und „plant“. Bemerkung: Die Objektbeschriftung ist spiegelverkehrt, da die Bezeichner global gesetzt werden und nicht von der Projektionskamera abhängen.

und enthält zusätzlich das Objekt „box“, sowie das Treppenhaus des AVZ-Gebäudes als weiteren Raum. In diesem Raum gibt es Wände im Winkel von 45° zum Flur. Hier schlägt der semantische Beschriftungsalgorithmus fehl, so dass die entsprechende Wand die Bezeichnung „unknown“ erhält. Insgesamt wurden 82% aller 3D-Punkte eine Bezeichnung zugeordnet.

Im Experiment wurde 3D-Objekterkennung mit einer Datenbasis von 11 3D-Objekten durchgeführt, die zuvor gelernt worden waren. Die für die 3D-Objekterkennung benötigte Zeit steigt mit der Anzahl der Objekte. Um diesen Nachteil auszugleichen, hat Frintrop eine Aufmerksamkeitssteuerung entwickelt [84]. Der folgende Abschnitt beschreibt kurz ihr System und demonstriert die erzielten Verbesserungen.

5.3.2 Ein Aufmerksamkeitsalgorithmus für die Kartierung von Objekten

Die Aufmerksamkeitssteuerung ermittelt hervorstechende Regionen in den Laserdaten. Ziel ist es, die Detektion von Objekten zu beschleunigen, indem die Objekterkennung auf interessante Bereiche beschränkt wird. Zur Berechnung der hervorstechenden Regionen verwendet der Aufmerksamkeits-

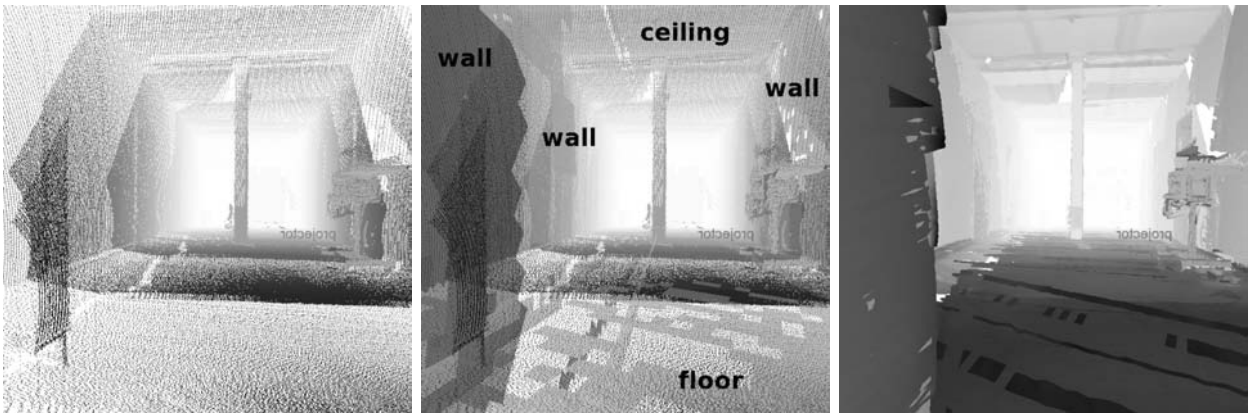


Abbildung 5.22: Semantische 3D-Karte (Forts.) mit dem Objekt „projector“.

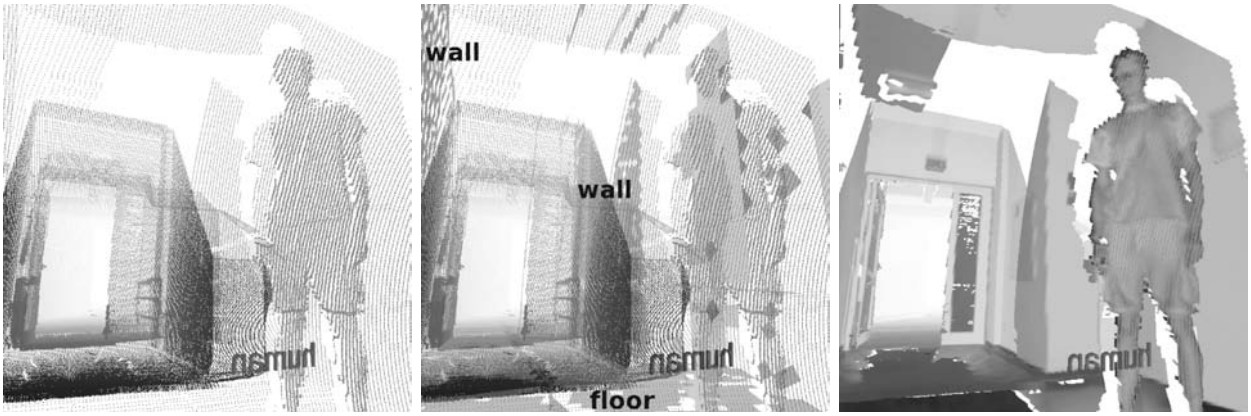


Abbildung 5.23: Semantische 3D-Karte (Forts.) mit dem Objekt „human“.

algorithmus die Tiefen- und Reflektionsbilder aus Abschnitt 4.3. Auffälligkeiten errechnen sich durch Intensitäts- und Orientierungsmerkmale in einer Bottom-up-Analyse der zwei Bildertypen und werden anschließend in einer so genannten Auffälligkeitskarte (engl.: *saliency map*) zusammengeführt. Der Aufmerksamkeitsfokus wandert mit Hilfe eines Inhibitionsprozesses danach sequentiell durch die Auffälligkeitskarte. Der Algorithmus beruht auf einem Standard-Aufmerksamkeitsmodell, wie es von Koch und Ullman eingeführt wurde [118]. Abbildung 5.24 zeigt die Struktur des Aufmerksamkeitsalgorithmus.

Merkmalsberechnung

Aus den Eingabebildern erzeugt man zunächst mit Hilfe von Gaußpyramiden fünf Skalierungen (0 bis 4), wobei ein Tiefpassfilter und eine Datenreduzierung angewandt werden [79]. Die skalierten Bilder unterscheiden sich jeweils um den Faktor zwei. Die so angefertigten Bildpyramiden erlauben die Berechnung von auffälligen Merkmalen unterschiedlicher Größe. Als Merkmale werden Intensitäten und Orientierungen bestimmt. Center-surround-Mechanismen erzeugen die Intensitätskarten, in-

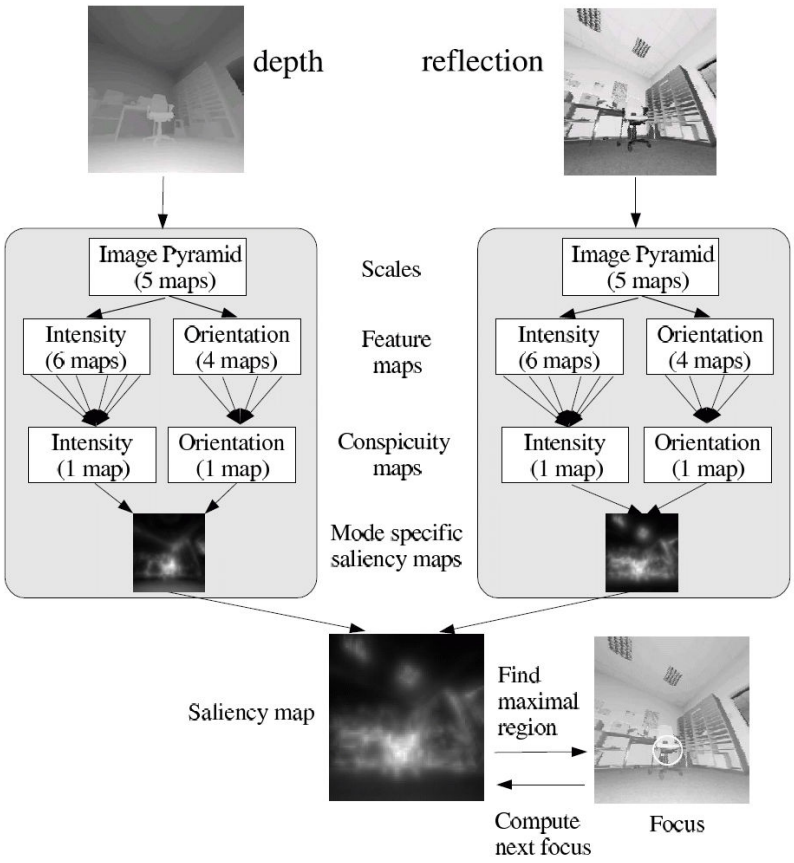


Abbildung 5.24: Das laserbasierte Aufmerksamkeitssystem. Merkmale in Tiefen- und Reflektionsbildern werden separat berechnet und anschließend in einer Auffälligkeitskarte zusammengeführt. Quelle: [85].

dem sie die Differenz zwischen einer Region und ihrer Umgebung bestimmen. Das Zentrum c ist durch einen Pixel in den Skalierungen 2 bis 4 gegeben; die Umgebung s bestimmt sich als Mittelwert der Umgebungspixel für zwei verschiedene Umgebungsgrößen, so dass sechs Intensitätskarten entstehen. Die Center-surround-Differenz $d = |c - s|$ ist ein Maß für den Intensitätskontrast in der Region.

Um die Orientierungskarten zu erhalten, werden mit vier ausgerichteten Gaborpyramiden rechteckähnliche Merkmale der Orientierungen 0°, 45°, 90° und 135° ermittelt. Der Algorithmus summiert die Karten 2 bis 4 jeder Pyramide mit einer so genannten Inter-Scale-Addition, d.h., alle Karten müssen zuvor auf Größe 2 gebracht werden, um eine pixelbasierte Addition zu ermöglichen. Es entstehen vier Orientierungskarten der Skalierung 2, also eine für jede Orientierung.

Die Fusion der Auffälligkeiten

Für jedes Merkmal kombiniert der Algorithmus die Merkmalskarten in einer Karte (vgl. Abbildung 5.24). Diese werden anschließend zu einer Auffälligkeitskarte zusammengeführt. Zunächst getrennt nach Tiefen- und Reflektionsbild, dann in einer einzigen Auffälligkeitskarte S. Die Summation der Karten läuft mittels Gewichtung, Größenanpassung und pixelbasierter Addition ab. Einige

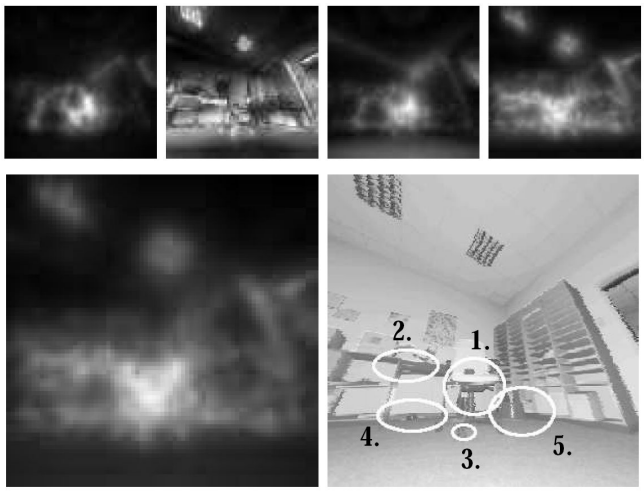


Abbildung 5.25: Auffälligkeitskarte zu einem 3D-Scan. Oben: Auffälligkeitskarte für Orientierungen (Tiefenbild), Auffälligkeitskarte für Intensitäten (Reflektionsbild), Auffälligkeitskarte "Tiefenbild" und Auffälligkeitskarte "Reflektionsbild". Unten links: Kombinierte Auffälligkeitskarte. Unten rechts: Die fünf auffälligsten Regionen (nummeriert). Quelle: [85].

Auffälligkeitskarten zu einem 3D-Scan sind in Abbildung 5.25 wiedergegeben.

Der Fokus der Aufmerksamkeit

Zur Bestimmung des auffälligsten Punktes in S wird der hellste Punkt gewählt. Ausgehend von diesem Punkt startet eine Regionsvergrößerung, um rekursiv alle Nachbarpixel mit ähnlichen Helligkeitswerten zu finden. Die Breite und die Höhe dieser Region bestimmen die Größe eines elliptischen Aufmerksamkeitsfokus. Schließlich werden die Helligkeitswerte an der Fokusregion zurückgesetzt, so dass sich mit dem gleichen Verfahren der nächste Aufmerksamkeitsfokus berechnen lässt. Abbildung 5.25 (unten rechts) zeigt die fünf auffälligsten Regionen eines 3D-Scans.

Geschwindigkeitsvorteil durch Aufmerksamkeit

Nach der Berechnung der auffälligsten Regionen findet die in Abschnitt 4.3 beschriebene Objekterkennung ausschließlich an diesen Punkten statt, folglich muss nicht das ganze Bild untersucht werden. Dies bringt einen erheblichen Geschwindigkeitsvorteil, zumal während der Erstellung semantischer 3D-Karten nach vielen Objekten gesucht wird. Der Geschwindigkeitsgewinn korreliert linear mit der Anzahl der Objekte der Datenbasis. Das Verfahren birgt die Gefahr, dass einige Objekte nicht gefunden werden. Empirische Untersuchungen ergaben jedoch, dass dies nur sehr selten auftritt [85].

Der Aufmerksamkeitsalgorithmus profitiert von der Verwendung von Tiefen- und Reflektionsdaten, da diese beiden Modalitäten sich ergänzen: Angenommen zwei Objekte haben die gleichen Reflektionseigenschaften, dann kann dennoch eines der beiden auffälliger sein, da Unterschiede in den Tiefendaten auftreten können. Weitere Einzelheiten zum Aufmerksamkeitsalgorithmus finden sich in [84–87].

5.4 Diskussion

Dieses Kapitel hat das Robotersystem Kurt3D vorgestellt und präsentiert das Verfahren zum Erstellen von semantischen 3D-Karten. Kurt3D ist ein Explorationsroboter und unterscheidet sich daher von ewas von Serviceroboter, wie z.B dem erfolgreichen RHINO der Universität Bonn [42, 58]. Auch befindet sich Kurt3D in permanenter Weiterentwicklung [130,190]. Neuere, hier nicht behandelte Hardwareelemente für den Roboter sind eine vertikal rotierender 3D-Scanner (RTS-Hannover) [154, 211], ein Gyroskop [183], Ultraschall- und Infrarotsensoren, eine Wärmebildkamera [130], sowie ein GPS-Empfänger. Die Roboterplattform KURT2 lässt sich mit den kommerziell erhältlichen, ähnlichen Robotern P3DX und P3AT von Activemedia [14, 15], sowie iRobots ATRV-Jr [10] vergleichen (siehe. Abbildung 5.26):

P3DX ist ein Roboter mit Differentialantrieb, d.h. er besitzt zwei angetriebene Räder. Ein passives in alle Richtungen drehbares Rad am Heck des Roboters verleit dem System Stabilität. Zur Standardausstattung gehören Sonarsensoren und Odometrie, sowie Microcontroller-Software zum Ansprechen der Motoren. Oft sind P3DX Roboter mit einem SICK Laserscanner und einer Kamera ausgestattet. Trotz ähnlicher Größe, ist die KURT2-Plattform mobiler, da das paasive Rad nur sehr klein ist und somit selbst keine Hindernisse nicht überwunden werden können.

P3AT besitzt wie die KURT2-Roboter ein Skid-Steer Antrieb. Er verfügt über 4 Räder; zwei an jeder Seite. Im Gegensatz zu KURT2 lassen sich alle Räder separat ansteuern. Dennoch ist der P3AT nicht mobiler, weil die Bodenfreiheit der Plattformen ungefähr gleich ist. Durch seine größeren Ausdehnungen benötigt der P3AT mehr Platz zum Fahren braucht als KURT2.

ATRV-Jr ist noch etwas größer als der P3AT und ist ebenfalls ein vier rädriger skid-steered Roboter. Seine Ausdehnungen belaufen sich auf 55 cm (Länge) $\times$ 77.5 cm (Breite), sein Gewicht beträgt 60 – 85 Kg. Er besitzt zwei Pentium-III CPUs zur Sensordatenverarbeitung und Steuerung. Als Sensoren sind standardmässig ein SICK LMS Scanner, 17 Sonarsensoren, ein GPS System und eine Kamera auf einer Schwenk-Neigevorrichtung. Anwendung findet der ATRV-Jr in Outdoorszenarien, für Einsätze bei RoboCup Rescue ist er zu groß.

Die von der Firma KTO in Lizenz für die Fraunhofer Gesellschaft gefertigte Roboterplattform KURT2 ähnelt den obigen Plattformen: Ein Microcontroller dient zum Ansprechen der Räder und

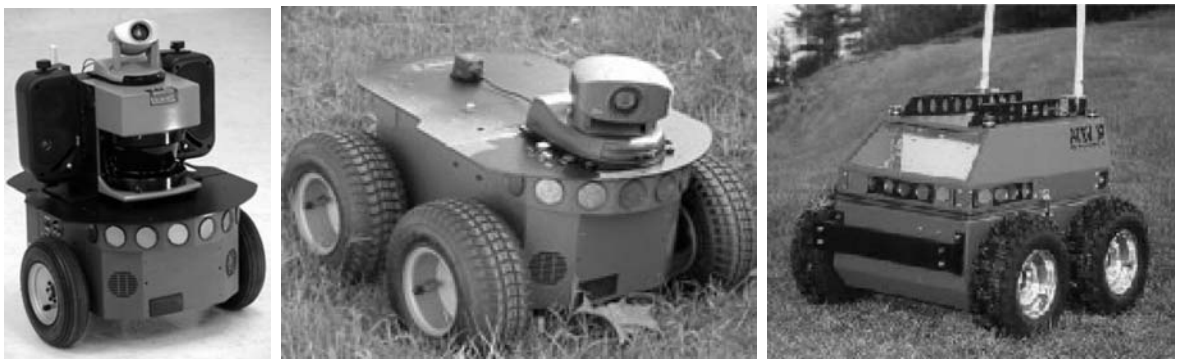


Abbildung 5.26: Roboterplattformen. Links: P3DX von Activemedia [14] . Mitte: P3AT von Activemedia [14]. Rechts: ATRV-Jr von iRobot [10].

dem Auslesen von Sensorwerten, jedoch ist die hier vorgestellte Steuerung des Roboters Kurt3D etwas ungewöhnlich: Die Motorenregelschleife läuft auf dem Notebook und nicht auf dem Microcontroller. Die 100 Hz schnelle Regelschleife ist der einzige Task. Daher sind alle weiteren Algorithmen serialisert, d.h., sie sind so programmiert, dass sie nach einer bestimmten fest definierten Zeit unterbrochen werden können, um die Regelschleife weiterzuführen. Dies ermöglicht dem Entwickler sehr hohe Flexibilität, da die Regelparameter leicht angepasst werden können. Die oben erwähnten vergleichbaren Roboter, besitzen fest eingebaute Motorenregler auf die die Steuerung aufsetzt.

2D-Laserscanner gehören seit Anfang der 90er zur Standardausstattung von Forschungsrobotern. Für die Verarbeitung von 2D-Laserscans zur Lokalisierung wurden eine Vielzahl von Verfahren entwickelt: Lu und Milios führten 1994 zunächst punktbasiertes 2D-Scanmatching als ICP, bzw. IDC (engl.: *iterative dual correspondence*) ein [135] und eintwickelten darauf aufbauend ein SLAM Verfahren [136]. Dazu gleichzeitig wurden Matching mittels Belegtheitsgittern erforscht [164]. Weitere Verfahren benutzen eine Scanvorverarbeitung um die Scan zu matchen: Hierzu zählen die Methoden, die sich auf Histogramme stützen [169,210], die Scanpunkte vorhandenen Linien zuordnen [57], oder Ecken und Kanten nutzen [37,113]. Im Vergleich zu diesen Verarbeitungsverfahren für 2D-Scan ist HAYAI um Größenordnungen schneller und gliedert sich nahtlos in die Softwarearchitektur, d.h. in den Kontrollzyklus, von Kurt3D ein.

Das Herzstück von Kurt3Ds Sensorik ist der 3D-Laserscanner mit dem der Roboter die semantischen 3D-Karten erstellt. Das Kartieren von Umgebungen mit Objekten wurde bisher fast gar nicht erforscht. Kester et al. berichten von einem Roboterarchitektur, die Umgebungskartierung und Objektdarstellung vereinigt [116]. Die räumliche Repräsentation Architektur (vgl. Abbildung 5.27) kombiniert eine Wissensbasis mit einer Regionenkarte und einem Belegtheitsgitter, so dass ein zwischen diesen Komponenten ein aktiver Datenaustausch stattfindet. Die Umbebung wird mit Hilfe eines 2D-Laserscanners und Sonarsensoren zweidimensional kartiert. Das Erkennen von Objekten geschieht im Kamerabild mit Hilfe visueller Aufmerksamkeit, Merkmalsextraktion und Klassifikation von neuronalen Netzen. Für die Repräsentation der Karten schlagen Kestler et al. eine bis zu 10000 mal kompakter Darstellung als Belegtheitsgitter vor. Abbildung 5.28 (links) zeigt diese Grundrißrepräsentation, sowie die Attribute der Regionen für eine Büroumgebung (rechts).

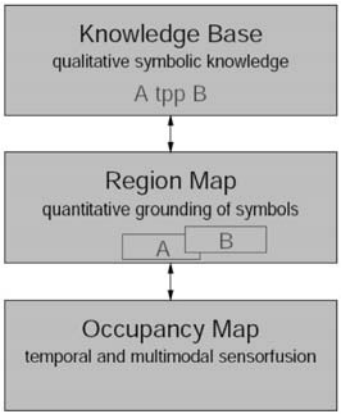
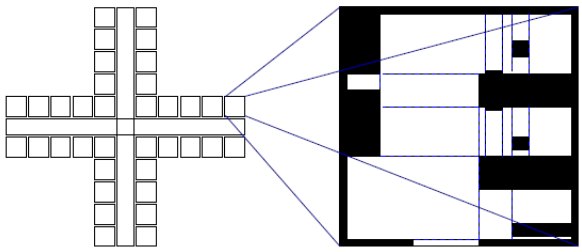


Abbildung 5.27: Räumliche Repräsentation Architektur DYNAMO. Quelle: [116].



$\text{occupied}(R) \mapsto \{\text{true}, \text{false}\}$
$\text{free\_space}(R) \mapsto \{\text{true}, \text{false}\}$
$\text{cylinder}(R) \mapsto \{\text{true}, \text{false}\}$
$\text{ball}(R) \mapsto \{\text{true}, \text{false}\}$
$\text{coffeemachine}(R) \mapsto \{\text{true}, \text{false}\}$
$\text{tetra}(R) \mapsto \{\text{true}, \text{false}\}$
$\text{possible\_location\_of\_chair\_X}(R) \mapsto [0, 1]$
$\text{color}(R) \mapsto \{\text{blue}, \text{green}, \text{red}, \text{yellow}\}$

Abbildung 5.28: Links: Grundrißrepräsentation zur Objektkartierung von Kester et al. Rechts: Typische Attribute und Konsistenzkriterien einer Büroumgebung. Quelle: [116].

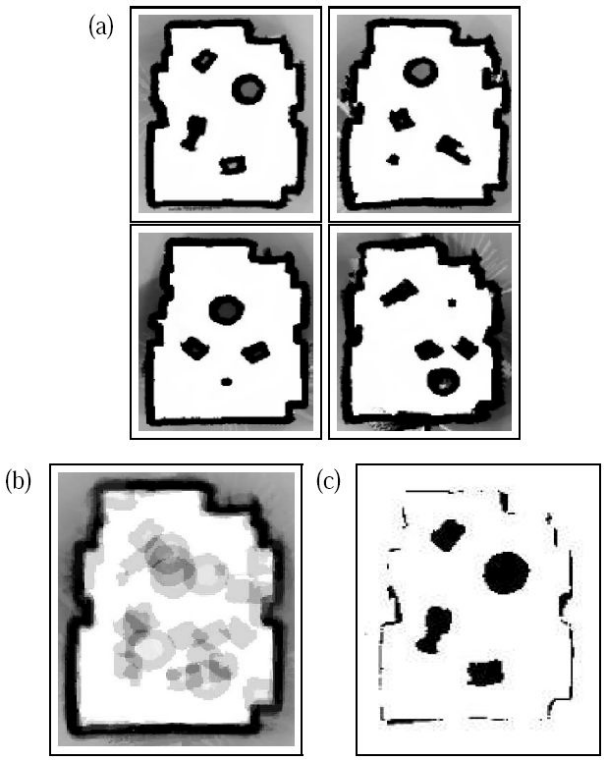


Abbildung 5.29: Lernen von Objekten aus mehreren Karten. Quelle: [35].

Einen weiteren Forschungszweig in Bezug auf die Kartierung von Objekten stellen die Arbeiten von Anguelov [20] und Biwas [35] dar. Hier steht im Vordergrund, kartierte Objektpunkte als Objekte zu klassifizieren. Dies geschieht durch wiederholte Anwendung eines Kartierungsverfahrens, wobei Objekte daruch identifiziert werden, dass sie Ihre Position verändert hat. Abbildung 5.29 zeigt ein Beispiel: Zunächst werden 4 Karten mit einer konstanten Anzahl Objekten erstellt (a) und überlagert (b). Eine so genannte Differenzkarte (c) zeigt schließlich die Objekte. Schwerpunkt dieser Arbeiten ist die Wissensrepräsentation mit Unsicherheiten.

Kapitel 6

Zusammenfassung und Ausblick

Die vorliegende Arbeit beschäftigt sich mit semantischen dreidimensionalen Karten für Roboter. Sie sind definiert als Karten, die neben den geometrischen Informationen von 3D-Messpunkten semantische Beschriftungen der Punkte enthalten. Der erste Teil der Arbeit behandelt das Erstellen dreidimensionaler Karten, die anschließend interpretiert werden. Dabei wird ein 3D-Laserscanner als Sensor verwendet, der das berührungslose Abtasten der Umgebung auf zirka 5 cm erlaubt. Durch die Genauigkeit, die maximale Reichweite und den Öffnungswinkel des Scanners lassen sich mit ihm besonders gut mittelgroße Objekte wie Stühle oder Personen sowie Strukturen wie Wände oder Fußböden, etc. wahrnehmen. Die Interpretation bezieht sich auf diese Objekttypen. Der verwendete 3D-Laserscanner basiert auf einem Standard-2D-Lasercanner [191, 192].

2D-Laserscanner gehören seit längerer Zeit standardmäßig in die mobile Robotik. Die metrische 2D-Kartierung von planaren, wohl-definierten Innenräumen gilt weitestgehend als erforscht. Ein offenes Problem stellt die Kartierung von unstrukturierten Umgebungen dar. Das Erstellen einer Karte unter Berücksichtigung der vollständigen Roboterpose mit 6 Freiheitsgraden kann mit ungeordneten Umgebungen umgehen, ist flexibel und selbst im Freien einsetzbar. Die vollständige Roboterpose lässt sich mit bildgebender 3D-Sensorik erfassen, die in letzter Zeit in zunehmendem Maße in der mobilen Robotik eingesetzt wird [18, 25, 100, 153, 178, 179, 211].

Die Basis des 3D-Kartierungsalgorithmus ist ein schneller Scanmatchingalgorithmus, der auf einer mathematisch geschlossenen Berechnung der Scantransformationen beruht. Auf der Grundlage theoretischer Ergebnisse wird ein schneller Algorithmus zur Bestimmung der approximativen nächsten Nachbarn eingesetzt. Es wird empirisch nachgewiesen, dass sich die Approximation positiv auf das 3D-Kartierungsverfahren auswirkt. Eine Heuristik zur Bestimmung der Startpose für ICP und zur Detektion geschlossener Kreise rundet den vorgestellten SLAM Algorithmus ab.

Als eine wichtige Aufgabe bei der robotergestützten Kartierung mit Hilfe von 3D-Tiefenbildern wird die Kombination der deterministischen Techniken zur Registrierung mit den stochastischen Lokalisationstechniken genannt [105]. Es muss das Ziel verfolgt werden, die Genauigkeit der Registrierungstechniken mit der Flexibilität der stochastischen Methoden zu verbinden. Im Gegensatz zur 3D-Kartierung gibt es bereits Ansätze auf dem Gebiet der Navigation, z.B. Monte Carlo Lokalisation auf der Basis von 3D-Daten [212].

Die Semantik wird in zwei Schritten in die 3D-Karten gebracht. Zunächst dient ein semantisches Netz zur Interpretation der in der 3D-Szene gefundenen Ebenen. Anschließend detektiert und lokalisiert ein Algorithmus zuvor gelernte 3D-Objekte im Scan. Hierbei berücksichtigt der Algorithmus

alle 6 Freiheitsgrade der Objekte und nutzt sowohl den Tiefen- als auch den Reflektionsbildmodus des 3D-Scanners. Als Lernverfahren wurde das im aktuellen Trend liegende Ada-Boost Verfahren verwendet.

In der vorliegenden Arbeit spielen Anwendungen eine wichtige Rolle. Hierbei stehen die außerordentlich wichtigen Anwendungen der robotergestützten 3D-Kartierung im Vordergrund: Die Kartierung von Kohleminen und die Rettungsrobotik. Eine weitere Anwendung des Kartierungsverfahrens stammt aus der Medizin. Die Aufnahme von 3D-Gesichtsprofilen mittels Holographie kann Eingabedaten für das 3D-Scanmatching liefern. In dem demonstrierten Beispiel wird das Gesicht von Patienten dreidimensional kartiert.

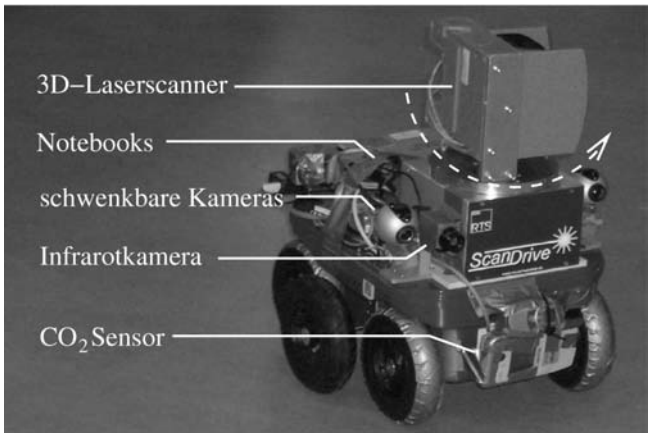


Abbildung 6.1: Der Roboter Kurt3D (Version 2005), ausgestattet mit einem kontinuierlich um die vertikale Achse drehenden 3D-Laserscanner. Der 3D-Scanner wurde am Institut für Echtzeitsysteme (RTS) an der Universität Hannover entwickelt.

3D ist ein aktueller Trend in der Robotik. Die Entwicklung von 3D-Sensorik schreitet schnell voran. So kam beispielsweise im Robocup Rescue Wettbewerb 2005 ein kontinuierlich drehender Laserscanner zum Einsatz [154]. Abbildung 6.1 zeigt den Roboter. Seine Vorteile bestehen in der geringeren mechanischen Belastung der Hardware, dem großen Öffnungswinkel von 360° sowie in der Aufnahme von 3D-Daten ohne Unterbrechung. Die Umlaufzeit ist größer als 2.4 Sekunden. Virtuelle 2D-Scan, die aus den 3D-Daten gewonnen werden, verbessern viele planare Verfahren, z.B. planares SLAM [211] und Monte Carlo Lokalisation [212]. Verfahren wie HAYAI können allerdings nicht eingesetzt werden. Mit der Verbesserung der Robotersensorik müssen auch die Algorithmen schritthalten: Immer mehr Umgebungsdaten können zuverlässig aquiriert werden, die mit den begrenzten Mitteln eines mobilen Roboters verrechnet werden müssen.

Mobile Manipulation wird in Zukunft ein großes Forschungsthema werden. Industrieroboter spielen bereits eine bedeutende wirtschaftliche Rolle und die mobile Robotik liefert zunehmend zuverläsige Geräte. Die Kombination beider Richtungen birgt viel wirtschaftliches Potential. Mit Manipulatoren ausgestattete mobile Roboter benötigen sichere Objekterkennungsmethoden und semantische Interpretationen. Dies ermöglicht, dass das Wissen kommunizierbar und inspizierbar wird.

Zukünftige Arbeiten am Roboter Kurt3D werden zu den Themen Verbesserung der Objekterkennung, Planung, Rettungsrobotik, Robotersimulation und Mensch-Roboter Interaktion stattfinden:

Objekterkennung. Eine silhouettebasierten Objekterkennung wird zurzeit untersucht. Hierbei bestimmt die Objekterkennung die Konturen in den Tiefenbildern und nutzt die Tatsache aus, dass Umrisse im Tiefenbild klar zum Vorschein kommen. S. Stiene zeigt, dass sich Silhouette von auf dem Fußboden stehenden Objekten leicht extrahieren lassen, wenn die Semantik "Fußbodenpunkt" bekannt ist [187].

Planung. Die Kontrollarchitektur des Roboters wird noch weiter ausgebaut werden. So ist es dem Roboter derzeit nicht möglich, geplant zu navigieren, beispielsweise aus Sackgassen heraus-

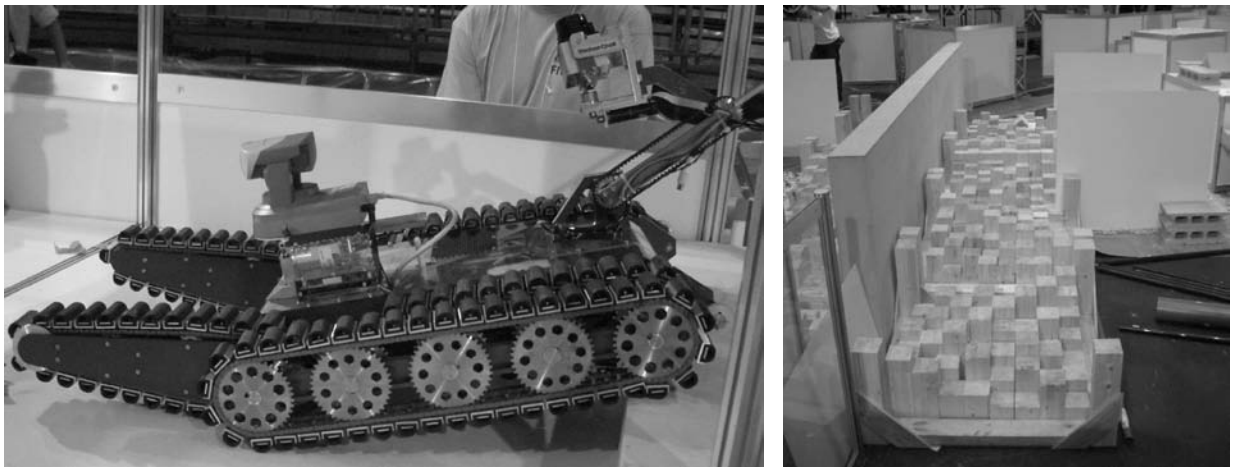


Abbildung 6.2: Der Toin Pelikan Roboter der Toin Universität in Yokohama (links) und Stufenfelder (rechts).

zumanövrieren oder den Weg zu weiteren Scanpositionen mit mehreren Zwischenstationen anzufahren.

Rettungsrobotik. Der mobile Roboter Kurt3D kann nicht in zerstörten Gebieten fahren. Die NIST fördert die Entwicklung solcher Plattformen durch die Einführung von Stufenfeldern (engl.: *step fields*, vgl. Abbildung 6.2 rechts), die von Ziegelhaufen gleichen. Die Holzquader haben einen Querschnitt von zirka 10 cm × 10 cm. Um in solchen Gebieten navigieren zu können, sind mobilere Plattformen als Kurt3D notwendig. Beispielsweise verfügt der Toin Pelikan Roboter über eine enorme Mobilität. In den zukünftigen Arbeiten zur Rettungsrobotik wird mit kettengetriebenen Fahrzeugen experimentiert. Dabei sollen 3D-Karten von mehreren Robotern simultan bzw. kooperativ erstellt werden.

Simulation. Das Entwickeln von Algorithmen für autonome Roboter besitzt einen zeitaufwändigen, hohen Anteil an Experimenten. Simulationen können diesen reduzieren. Daher werden kommende Arbeiten mit Unterstützung des Simulators USARSIM entwickelt, wobei in der Ausprägung UOSSIM neben Activmedias P2AT, P2DX und iRobots ATVR Robotern auch Modelle des 3D-Laserscanners, von KURT2 und Kurt3D zur Verfügung stehen. „USARSIM ist die erste hochentwickelte Robotersimulation auf der Basis einer *game engine*, die hier aus dem Computerspiel *Unreal Tournament* 2003 bzw. 2004 stammt. Damit nutzt der Simulator die fortschrittliche Grafik und physikalische Modellierung eines kommerziellen Programms“ [106].

Mensch-Roboter-Interaktion. Die Mensch-Roboter-Interaktion (engl.: *Human-robot interaction (HRI)*) für mobile Roboter steckt in den Kinderschuhen. Viele Benutzerinteraktionen mit Robotern sind vom Typ Teleoperation, wobei Videorückmeldung von der Roboterplattform auf dem Eingaberechner stattfindet. Der Operator bestimmt den Weg des Roboters. Roboterexperten entwickeln in der Regel diese Mensch-Roboter-Schnittstelle. Dementsprechend kann sie oft nur von den Experten selbst benutzt werden. Da mehr und mehr Nicht-Experten in die Roboter-Mensch-Regelschleife einbezogen werden, muss die Interaktion entsprechend umgestaltet werden [175].

Anhang A

Herleitungen und Beweise

A.1 Sätze der Linearen Algebra

Die folgenden Beweise stammen aus [108] und [75]. Sie vertiefen und verallgemeinern die Beweise in Kapitel 3.1.1.

Lemma 4. Sei $\mathbf{R}$ eine orthonormale Matrix. Dann gilt für jeden Vektor $\mathbf{x}$:

$$(\mathbf{Rx}) \cdot \mathbf{x} \leq \mathbf{x} \cdot \mathbf{x}.$$

Die beiden Ausdrücke sind gleich, falls $\mathbf{Rx} = \mathbf{x}$.

Beweis: Es gilt

$$(\mathbf{Rx}) \cdot (\mathbf{Rx}) = (\mathbf{Rx})^T(\mathbf{Rx}) = \mathbf{x}^T\mathbf{R}^T\mathbf{Rx} = \mathbf{x}^T\mathbf{x} = \mathbf{x} \cdot \mathbf{x},$$

da $\mathbf{R}^T\mathbf{R} = \mathbb{1}$. Weiterhin gilt

$$\begin{aligned}(\mathbf{Rx} - \mathbf{x}) \cdot (\mathbf{Rx} - \mathbf{x}) &= (\mathbf{Rx}) \cdot (\mathbf{Rx}) - 2(\mathbf{Rx}) \cdot \mathbf{x} + \mathbf{x} \cdot \mathbf{x} \\ &= 2(\mathbf{x} \cdot \mathbf{x} - (\mathbf{Rx}) \cdot \mathbf{x}). \end{aligned} \tag{A.1}$$

Da $(\mathbf{Rx} - \mathbf{x}) \cdot (\mathbf{Rx} - \mathbf{x}) \geq 0$ gilt, ist die Behauptung des Satzes bewiesen. Die Gleichheit gilt nur wenn $(\mathbf{Rx} - \mathbf{x}) \cdot (\mathbf{Rx} - \mathbf{x}) = 0$ auftritt. Nach Gleichung (A.1) tritt dieser Fall nur ein, wenn $\mathbf{Rx} = \mathbf{x}$ ist. □

Lemma 5. Jede positiv semidefinite $n \times n$ Matrix $\mathbf{S}$ lässt sich mit Hilfe einer orthonormalen Menge von Vektoren $\{\mathbf{u}_i\}$ schreiben als

$$\mathbf{S} = \sum_{i=1}^{n} \mathbf{u}_i \mathbf{u}_i^T.$$

Beweis: Seien die Eigenwerte von $\mathbf{S}$ $\{\lambda_i\}$ und die zugehörige Menge von Einheitseigenvektoren $\{\hat{\mathbf{u}}_i\}$. Dann lässt sich $\mathbf{S}$ als

$$\mathbf{S} = \sum_{i=1}^{n} \lambda_i \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i^T$$

ausdrücken. Da die Eigenwerte einer positiv semidefiniten Matrix nicht negativ sind, folgt die Behauptung mit $\mathbf{u}_i = \sqrt{\lambda_i}\hat{\mathbf{u}}_i$. □

Lemma 6. Wenn $\mathbf{S}$ eine positiv semidefinite Matrix ist, dann gilt für jede orthonormale Matrix $\mathbf{R}$

$$\text{Trace}\,(\mathbf{RS}) \leq \text{Trace}\,(\mathbf{S})\,,$$

wobei die Gleichheit eintritt, wenn $\mathbf{RS} = \mathbf{S}$ ist.

Beweis: Sei $\mathbf{S}$ wie in Lemma 5 dargestellt. Da $\text{Trace}\left(\mathbf{ab}^T\right) = \mathbf{a} \cdot \mathbf{b}$ gilt, folgt

$$\text{Trace}\,(\mathbf{S}) = \sum_{i=1}^{n} \mathbf{u}_i \cdot \mathbf{u}_i \qquad \text{und} \qquad \text{Trace}\,(\mathbf{RS}) = \sum_{i=1}^{n} (\mathbf{Ru}_i) \cdot \mathbf{u}_i,$$

wobei letztere Gleichung mit Lemma 4 kleiner oder gleich der Spur $\text{Trace}\,(\mathbf{S})$ ist. Gleichheit ergibt sich für $\mathbf{Ru}_i = \mathbf{u}_i$, also wenn $\mathbf{RS} = \mathbf{S}$ ist, weil $\mathbf{Su}_i = \mathbf{u}_i$ gilt. □

Korollar 1. Wenn $\mathbf{S}$ eine positiv definite Matrix ist, dann gilt für jede orthonormale Matrix $\mathbf{R}$

$$\text{Trace}\,(\mathbf{RS}) \leq \text{Trace}\,(\mathbf{S})\,,$$

wobei die Gleichheit nur eintritt, wenn $\mathbf{R} = \mathbb{1}$ ist. □

Lemma 7. Die folgende Matrix $\mathbf{T}$ ist die positiv semidefinite Quadratwurzel der positiv definiten Matrix $\mathbf{S}$:

$$\mathbf{T} = \sum_{i=1}^{n} \sqrt{\lambda_i}\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^T \qquad \text{und} \qquad \mathbf{S} = \sum_{i=1}^{n} \lambda_i\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^T,$$

mit $\{\lambda_i\}$ der Menge der Eigenvektoren und $\{\hat{\mathbf{u}}_i\}$ der Menge der orthogonalen Einheitseigenvektoren von $\mathbf{S}$.

Beweis: Es gilt

$$\begin{aligned} \mathbf{T}^2 &= \left(\sum_{i=1}^{n} \sqrt{\lambda_i}\hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^T\right)\left(\sum_{j=1}^{n} \sqrt{\lambda_j}\hat{\mathbf{u}}_j\hat{\mathbf{u}}_j^T\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \sqrt{\lambda_i\lambda_j}\,(\hat{\mathbf{u}}_i \cdot \hat{\mathbf{u}}_j)\,\hat{\mathbf{u}}_i\hat{\mathbf{u}}_j^T \\ &= \sum_{k=1}^{n} \lambda_k\hat{\mathbf{u}}_k\hat{\mathbf{u}}_k^T = \mathbf{S}, \end{aligned}$$

da $\hat{\mathbf{u}}_i \cdot \hat{\mathbf{u}}_j = 0$ ist für $i \neq j$. Des Weiteren ist

$$\mathbf{xTx} = \sum_{i=1}^{n} \lambda_i(\hat{\mathbf{u}}_i \cdot \mathbf{x})^2 \leq 0,$$

weil die Eigenwerte $\lambda_i \geq 0$ sind. Somit ist auch $\mathbf{T}$ positiv semidefinit. □

Korollar 2. Für alle positiv semidefiniten Matrizen existiert eine positiv semidefinite Quadratwurzelmatrix. □

Korollar 3. Die Matrix

$$\mathbf{T}^{-1} = \sum_{i=1}^{n} \frac{1}{\sqrt{\lambda_i}} \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i^T$$

ist die Inverse der positiv semidefiniten Quadratwurzelmatrix der positiv semidefiniten Matrix

$$\mathbf{S} = \sum_{i=1}^{n} \lambda_i \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i^T.$$

□

Lemma 8. Für jede Matrix $\mathbf{X}$ sind folgende Matrizen positiv semidefinit:

$$\mathbf{X}^T\mathbf{X} \qquad \text{und} \qquad \mathbf{X}\mathbf{X}^T.$$

Die Matrix $\mathbf{X}^T\mathbf{X}$ ist symmetrisch, da $(\mathbf{X}^T\mathbf{X})^T = \mathbf{X}^T(\mathbf{X}^T)^T = \mathbf{X}^T\mathbf{X}$ gilt. Weiterhin folgt für jeden Vektor $\mathbf{x}$

$$\mathbf{x}^T(\mathbf{X}^T\mathbf{X})\mathbf{x} = (\mathbf{x}^T\mathbf{X}^T)(\mathbf{X}\mathbf{x}) = (\mathbf{X}\mathbf{x})^T(\mathbf{X}\mathbf{x}) = (\mathbf{X}\mathbf{x}) \cdot (\mathbf{X}\mathbf{x}) \geq 0.$$

Beweis: Bei $\mathbf{X}\mathbf{X}^T$ wird analog argumentiert. □

Korollar 4. Für jede nicht singuläre quadratische Matrix $\mathbf{X}$ sind folgende Matrizen positiv definit:

$$\mathbf{X}^T\mathbf{X} \qquad \text{und} \qquad \mathbf{X}\mathbf{X}^T.$$

□

Lemma 9. Jede nicht singuläre Matrix $\mathbf{X}$ kann als

$$\mathbf{X} = \mathbf{P}\mathbf{S}$$

ausgedrückt werden. Dabei ist

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1/2}$$

eine orthonormale Matrix und

$$\mathbf{S} = (\mathbf{X}^T\mathbf{X})^{-1/2}$$

eine positiv semidefinite Matrix.

Beweis: Da $\mathbf{X}$ nicht singulär ist, folgt mit Korollar 4, dass $\mathbf{X}^T\mathbf{X}$ positiv definit ist. Auch ergibt sich aus Korollar 2 die Existenz einer positiv definiten Quadratwurzelmatrix $(\mathbf{X}^T\mathbf{X})^{1/2}$, die sich nach Korollar 3 konstruieren lässt. Daher können $\mathbf{S}$ und $\mathbf{P}$ bei gegebenem $\mathbf{X}$ gefunden werden. Offensichtlich ist ihr Produkt

$$\mathbf{P}\mathbf{S} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1/2}(\mathbf{X}^T\mathbf{X})^{-1/2} = \mathbf{X}.$$

Zu überprüfen bleibt, ob $\mathbf{P}$ orthonormal ist. Es gilt:

$$\mathbf{P}^T = (\mathbf{X}^T\mathbf{X})^{-1/2}\mathbf{X}^T$$

und folglich

$$\begin{aligned}\mathbf{P}^T\mathbf{P} &= (\mathbf{X}^T\mathbf{X})^{-1/2}(\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1/2} \\ &= (\mathbf{X}^T\mathbf{X})^{-1/2}(\mathbf{X}^T\mathbf{X})^{1/2}(\mathbf{X}^T\mathbf{X})^{1/2}(\mathbf{X}^T\mathbf{X})^{-1/2} \\ &= \mathbb{1}.\end{aligned}$$

□

Anhang B

Minimierungsalgorithmen

Viele Algorithmen führen Optimierungsprobleme auf die Minimierung oder Maximierung von Funktionen zurück, z.B. indem sie zu minimierende Fehlerfunktionen konstruieren. Dieses Kapitel beschäftigt sich mit den Verfahren, die eine gegebene Funktion f minimieren oder maximieren. Dabei hängt die Funktion f von einer oder mehreren Variablen ab. Die Aufgaben der Maximierung und Minimierung sind äquivalent, da die Multiplikation des Funktionswertes mit -1 sie ineinander überführt. Daher wird im Folgenden nur die Minimierung von Funktionen betrachtet. Ein minimaler Wert der Funktion f an einem Punkt wird als Extremum bezeichnet. Ein Extremum kann global sein, dann liegt der minimale Funktionswert des gesamten Wertebereichs vor. Ein lokales Extremum ist ein minimaler Funktionswert in einer endlichen Umgebung.

Nachfolgend beschriebene Algorithmen finden lokale Extrema (vgl. auch [80] und [165]). Sie sind Bestandteil der in den vorangegangenen Kapiteln beschriebenen Methoden. Weitere Minimierungsverfahren finden sich in [165]. Die Abschnitte B.1, B.2 und B.3 beschäftigen sich mit den Verfahren, die keine Ableitungen der Funktion f benötigen.

B.1 Die Minimierung eindimensionaler Funktionen nach Brent

Für die Minimierung einer eindimensionalen Funktion $f : \mathbb{R} \to \mathbb{R}$ schlägt Brent eine Methode vor, die auf der so genannten „Goldenen-Schnitt-Suche“ aufbaut. Diese Suche wird um eine meist zutreffende Heuristik erweitert [165].

Die Goldene-Schnitt-Suche. Gegeben sei eine nicht-singuläre Funktion f und ein Tripel (a, b, c) mit den Funktionswerten $f(a)$, $f(b)$ und $f(c)$. Es gelte $a < b < c$. Die Funktion f hat im Intervall $[a, c]$ ein Minimum, falls $f(b) < f(a)$ und $f(b) < f(c)$ gilt (vgl. Abbildung B.1 links). Die Suche bestimmt einen Punkt x, der entweder zwischen a und b oder zwischen b und c liegt. Falls der letztere Fall vorliegt und die Auswertung der Funktion an der Stelle x $f(b) < f(x)$ ergibt, wird das Tripel (a, b, x) gewählt. Ergibt die Auswertung $f(b) > f(x)$, ist (b, x, c). Mit diesem neu gebildeten Tripel startet die Suche nochmals. Die Iteration wird beendet, wenn das Intervall klein genug ist.

Die Strategie zur Auswahl des neuen Punktes x bei gegebenen (a, b, c) verwendet den Goldenen Schnitt: Sei w der Teil zwischen a und c, d.h.

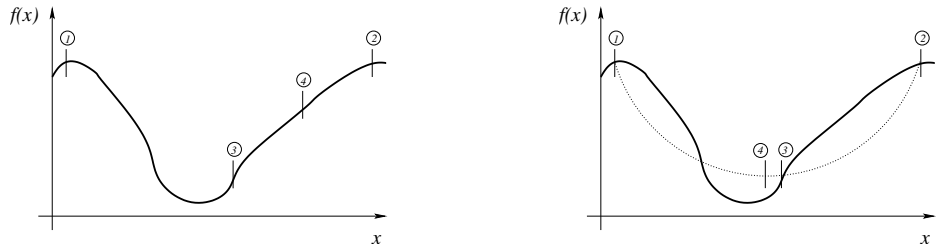


Abbildung B.1: Bestimmung des Minimums einer eindimensionalen Funktion $f(x)$. Links: Das Minimum liegt im Startintervall (1,2,3). Nach der Evaluation der Funktion an 4 verkleinert sich das Intervall zu (1,3,4). Rechts: Brents Heuristik legt eine Parabel durch die Funktionswerte und bestimmt den neuen Intervallpunkt als Minimum der Parabel.

$$\frac{b-a}{c-a} = w \quad , \quad \frac{c-b}{c-a} = 1-w.$$

Sei weiterhin der nächste Punkt x um den Teil z von b entfernt:

$$\frac{x-b}{c-a} = z.$$

Damit hat das nächste Intervall eine Größe von $w+z$ oder $1-w$, relativ zum aktuellen Intervall. Der Wert z bestimmt sich nun so, dass beide Intervalle gleich groß sind, d.h. $|b-a| = |x-c|$. Damit ist das Verfahren für den schlechtesten Fall optimiert. Es gilt:

$$z = 1-2w. \tag{B.1}$$

Der Wert w bestimmt sich aus der vorherigen Iterationsstufe. Die Wahl von z ist optimal, falls zuvor jene von w optimal war. Dies impliziert, dass x um den gleichen Teil zwischen b und c liegt (falls dies das größere Segment ist), wie b zwischen a und c lag, d.h.

$$\frac{z}{1-w} = w. \tag{B.2}$$

Setzt man die Gleichungen (B.1) und (B.2) ineinander ein, so ergibt sich die quadratische Gleichung des Goldenen Schnitts

$$w^2 - 3w + 1 = 0 \qquad \text{also} \qquad w = \frac{3-\sqrt{5}}{2} = 0.38197\ldots.$$

In jedem Intervall (a,b,c) ist b um den Teil 0.38197 von dem einen Intervallende und um den Teil 0.61803 vom anderen Intervallende entfernt. Die Goldene Schnitt Suche garantiert, dass das neu gebildete Intervall um den Faktor 0.61803 kleiner ist als das vorige. Entspricht das Startintervall nicht der Goldenen-Schnitt-Regel, stellt diese sich automatisch nach wenigen Iterationen ein.

Brents Heuristik. In der Goldenen-Schnitt-Suche werden keine Annahmen über die Funktion gemacht. Für glatte Funktionen schlägt Brent folgende Heuristik vor: Durch die Funktionswerte des Intervalls (a,b,c) wird eine Parabel gelegt. Die Heuristik nimmt an, dass das gesuchte Minimum

nahe dem Minimum der Parabel liegt (vgl. Abbildung B.1 rechts). Daher bestimmt sich der neue Intervallpunkt x als Minimum der Parabel, d.h.

$$x = b - \frac{1}{2}\frac{(b-a)^2(f(b)-f(c)) - (b-c)^2(f(b)-f(a))}{(b-a)(f(b)-f(c)) - (b-c)(f(b)-f(a))}.$$

Falls die drei Intervallpunkte colinear sind, kann die Formel nicht angewendet werden. Daher greift die Heuristik in diesem Fall auf die Goldene-Schnitt-Suche zurück.

B.2 Powells Minimierungsmethode für mehrdimensionale Funktionen

Für die Minimierung einer mehrdimensionalen Funktion $f : \mathbb{R}^n \to R$ mit $n \geq 2$ schlägt Powell eine Methode vor, die das Optimierungsproblem auf den eindimensionalen Fall zurückführt [163]. Sein Verfahren berechnet Richtungen, entlang derer ein eindimensionaler Funktionsminimierer zum Einsatz kommt. Dabei wird Brents Methode (vgl. Abschnitt B.1) benutzt [165]. Ausgehend von einem Startpunkt $\mathbf{x}_0$ im n-dimensionalen Suchraum wird die Funktion f entlang von Richtungen $\mathbf{u}_i$ minimiert.

Die Einheitsvektoren $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ könnten als Richtungen dienen. Für einige Funktionen ist dieses Verfahren sehr ineffizient. Beispiel dafür ist eine zweidimensionale Funktion, deren Funktionswerte ein enges Tal bilden. Ist dieses Tal nicht entlang der Achsen des Koordinatensystems ausgerichtet, sind sehr viele Funktionsauswertungen nötig. Im Gegensatz dazu sind konjugierte Richtungen gute Suchrichtungen, da diese sich nicht negativ beeinflussen.

Konjugierte Richtungen. Wird eine Funktion entlang der Richtung $\mathbf{u}_i$ minimiert, ist am Minimum der Gradient senkrecht zu $\mathbf{u}_i$ orientiert. Zusätzlich zu dieser Tatsache existiert eine Approximation der n-dimensionalen Funktion f am Punkt $\mathbf{p}$ durch eine Taylor-Reihe unter Benutzung von $\mathbf{p}_0$ als Ursprung des Koordinatensystems. Nach Taylor gilt

$$f(\mathbf{p}) = f(\mathbf{p}_0) + \sum_l \frac{\partial f}{\partial p_l} p_l + \frac{1}{2}\sum_{k,l} \frac{\partial^2 f}{\partial p_k \partial p_l} p_k p_l + \cdots \tag{B.3}$$

$$\approx c - \mathbf{b}\cdot\mathbf{p} + \frac{1}{2}\mathbf{p}\cdot\mathbf{A}\cdot\mathbf{p} \tag{B.4}$$

mit $c = f(\mathbf{p}_0)$, $\mathbf{b} = -\nabla f|_{\mathbf{p}_0}$ und $\mathbf{A}$ der Hesse-Matrix der Funktion f an dem Punkt $\mathbf{p}_0$. Die Methode der konjugierten Gradienten wählt zu einer gegebenen Richtung $\mathbf{u}_i$ eine neue $\mathbf{u}_j$ aus, so dass die Vektoren $\mathbf{u}_i$ und $\mathbf{u}_j$ senkrecht zueinander stehen. In der Approximation (B.4) ist der Gradient von f gegeben durch $\nabla f = A\cdot\mathbf{p} - \mathbf{b}$. Aus dem Differenzieren ($\delta(\nabla f) = \mathbf{A}(\delta\mathbf{p})$) folgt für die Richtungen $\mathbf{u}_i$ and $\mathbf{u}_j$, dass

$$0 = \mathbf{u}_i \cdot \delta(\nabla f) = \mathbf{u}_i \cdot \mathbf{A} \cdot \mathbf{u}_j. \tag{B.5}$$

Gleichung (B.5) definiert konjugierte Richtungen. Powells Methode bestimmt solche Richtungen, ohne Ableitungen der Funktion f zu verwenden.

Der folgende Algorithmus liefert konjugierte Suchrichtungen. Als Startrichtungen werden die Einheitsvektoren benutzt, d.h. $\mathbf{u}_i = \mathbf{e}_i$ für $i = 1, \ldots, n$.

Solange die Funktionwerte von f kleiner werden, führe vier Schritte aus:

1. Speichere die Startposition als $\mathbf{p}_0$.
2. Für $i = 1, \ldots, n$ bestimme mit $\mathbf{p}_{i-1}$ als Startwert das Minimum von f entlang der Richtung $\mathbf{u}_i$. Speichere das Minimum als $\mathbf{p}_i$.
3. Für $i = 1, \ldots, n-1$ setze $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$. Anschließend setze $\mathbf{u}_n \leftarrow \mathbf{p}_n - \mathbf{p}_0$.
4. Minimiere ausgehend von $\mathbf{p}_n$ die Funktion f entlang der Richtung $\mathbf{u}_n$. Das Minimum wird auf $\mathbf{p}_0$ gespeichert.

Obiger Algorithmus hat das Problem, dass durch das Ersetzen der Richtung $\mathbf{u}_1$ zugunsten der Richtung $\mathbf{p}_n - \mathbf{p}_0$ die vorgeschlagenen Vektoren linear abhängig werden. Dies hat zur Folge, dass nur ein Unterraum des n-dimensionalen Suchraums benutzt und das Minimum nicht gefunden wird. Für das Beheben des Problems gibt es etliche Vorschläge. In [165] wird eine Heuristik zur Lösung des Problems erklärt. Dabei ersetzt man in Schritt 3 die Richtung des größten Abstiegs mit der Differenz $\mathbf{p}_n - \mathbf{p}_0$. Für die in Abschnitt 4.2 gegebene Anwendung des Minimierungsalgorithmus hat sich experimentell herausgestellt, dass die Heuristik funktioniert.

B.3 Die Downhill-Simplex Methode

Ein weiterer Minimierungsalgorithmus für mehrdimensionale Funktionen, der ausschließlich auf Auswertungen der Funktion beruht, ist die Downhill-Simplex Methode. Sie ähnelt dem Verkleinern des Suchintervalls in Abschnitt B.1, ohne jedoch eine Regel oder Heuristik anzuwenden. Ein nicht-degenerierter Simplex in n-Dimensionen ist eine geometrische Struktur, die aus $n+1$ Knoten und ihren Verbindungslinien, Facetten etc. besteht. In zwei Dimensionen ist ein Simplex ein Dreieck, in drei Dimensionen ein Tetrahedron. Ein Simplex in n-Dimensionen spannt einen n-dimensionalen Vektorraum auf [165].

Sei ein Startpunkt $\mathbf{p}_0$ gegeben. Die Downhill-Simplex Methode berechnet zuerst weitere n Punkte, die den Simplex initialisieren. Es gilt

$$\mathbf{p}_i = \mathbf{p}_0 + \lambda \mathbf{e}_i,$$

mit $\mathbf{e}_i$ als Einheitsvektoren. Die Konstante λ sollte entsprechend der zu erwartenden Größenskala des Minimierungsproblems gewählt werden [165]. Das Downhill-Simplex Verfahren besteht aus einer Reihe von Schritten, die als Reflektionen und Kontraktionen bezeichnet werden. In einem Reflektionsschritt bewegt der Algorithmus den Punkt des Simplex mit dem größten Funktionswert durch die gegenüberliegende Facette zu einem Punkt mit kleinerem Funktionswert. Falls der Algorithmus ein Tal erreicht, führt er eine Kontraktion aus, d.h. der Rauminhalt des Simplex wird verkleinert. Dabei verschieben sich ein oder mehrere Punkte. Abbildung B.2 verdeutlicht die Transformationen des Simplex [165].

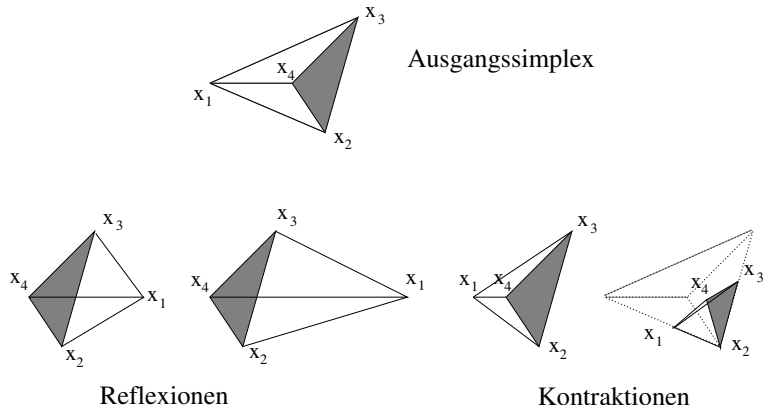


Abbildung B.2: Die möglichen Schritte des Downhill-Simplex Algorithmus sind die Reflektion, die Reflektion mit Expansion, die Kontraktion und die vielfache Kontraktion. Die Funktion $f(\mathbf{x})$ hat bei $\mathbf{x}_1$ den größten und bei $\mathbf{x}_4$ den kleinsten Funktionswert [165].

B.4 Minimierungsmethoden mit ersten Ableitungen

B.4.1 Der Gradientenabstieg

Um eine Funktion $f : \mathbb{R}^n \rightarrow \mathbb{R}$ zu minimieren, verwendet man oft Ableitungsinformationen. Der Vektor der Ableitungen von f in alle n Koordinatenrichtungen wird der Gradient von f genannt und oft als grad$f$ geschrieben. Der Gradient gibt die Steigungen an. Nachdem die Richtung des steilsten Abstieges bestimmt ist, müssen die Minimierungsalgorithmen ermitteln, wie weit in diese Richtung gegangen wird (vgl. Abschnitt B.1).

B.4.2 Die Newton Methode und der Levenberg-Marquardt Algorithmus

Das am häufigsten verwendete Verfahren zur nichtlinearen Optimierung ist das Verfahren nach Levenberg-Marquardt. Dieser Algorithmus liefert bei Vorgabe eines ausreichend genauen Startpunkts der Funktion meist in wenigen Schritten sehr genaue Ergebnisse.

Gegeben sei hierbei nun die Gleichung $\mathbf{y} = f(\mathbf{x})$, wobei die Vektoren $\mathbf{x}$ und $\mathbf{y}$ unterschiedliche Dimensionen haben dürfen. Gesucht wird nun der Vektor $\mathbf{x}$, der die Gleichung bestmöglichst erfüllt. Genauer ausgedrückt, es wird ein Vektor $\hat{\mathbf{x}}$ gesucht, der die Gleichung

$$\mathbf{y} = f(\hat{\mathbf{x}}) + \mathbf{e}$$

erfüllt, wobei $||\mathbf{e}||$ minimal ist.

Das Levenberg-Marquardt Verfahren ist eine Weiterentwicklung des Newton Verfahrens: Ein Startwert $\mathbf{x}_0$ wird iterativ unter der Annahme, dass f lokal linear dargestellt werden kann, verfeinert. Die erste Approximation von $f(\mathbf{x} + \Delta)$ liefert:

$$f(\mathbf{x}_0 + \Delta) = f(\mathbf{x}_0) + \mathbf{J}\Delta, \tag{B.6}$$

wobei $\mathbf{J}$ die Jacobi-Matrix und Δ eine kleine Abweichung ist. Unter dieser Voraussetzung kann die Minimierung der Gleichung

$$\mathbf{e} = \mathbf{e}_0 + \mathbf{J}\Delta$$

linear gelöst werden. Eine Erweiterung der Gleichung ergibt

$$\mathbf{J}^T\mathbf{J}\Delta = \mathbf{J}^T e.$$

Die Lösung des Problems kann iterativ erreicht werden, indem man, bei einem Startwert $\mathbf{x}_0$ beginnend, schrittweise die folgenden Werte

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta_i$$

berechnet, wobei Δ_i die Lösung der Gleichung (B.6) an der Stelle $\mathbf{x}_i$ ist. Dieses Verfahren betrachtet die Gradienten der Funktion, wobei es vorkommen kann, dass der ermittelte Minimalwert ein lokales, nicht aber das globale Minimum darstellt. Auch kann es vorkommen, dass der Algorithmus gar nicht konvergiert, sondern zwischen zwei Funktionswerten hin und her springt. Das Ergebnis ist also stark von der Wahl des Startparameters $\mathbf{x}_0$ und Δ abhängig.

Der Levenberg-Marquardt-Algorithmus ist eine Variation des Newton-Verfahrens. Die Gleichung

$$\mathbf{N}\Delta = \mathbf{J}^T\mathbf{J}\Delta = \mathbf{J}^T e$$

wird nun erweitert zu

$$\mathbf{N}'\Delta = \mathbf{J}^T e \qquad \text{mit} \quad N_{i,j} = (1 + \delta_{i,j})\lambda N_{i,j}.$$

Dabei steht $\delta_{i,j}$ für das Kronecker-Symbol. Mit dem Parameter λ wird also die Newton-Methode mit dem Gradientenabstieg kombiniert. Für große λ gleicht der Algorithmus dann dem Verfahren des steilsten Abstiegs.

B.5 Optimierung unter Nebenbedingungen

Optimieren unter Nebenbedingungen bedeutet, für eine gegebene Funktion $f : \chi \to \mathbb{R}$ und $\mathbf{g} : \chi \to \mathbb{R}^q$ und ein $\mathbf{c} \in \mathbb{R}^q$ das Minimum von f zu finden, so dass die Nebenbedingungen $\mathbf{g} = \mathbf{c}$ oder komponentenweise $\mathbf{g} \leq \mathbf{c}$ gilt. In vielen Fällen ist es einfacher, für einen beliebigen Vektor $\lambda \in R^q$ die Funktion $f + \lambda^T g$ auf ganz χ zu minimieren und anschließend λ zu variieren. Der folgende Satz erklärt, inwiefern dies funktioniert [80].

Satz 7. Für ein $\lambda \in \mathbb{R}^q$ sei $\mathbf{x}_\lambda \in \arg\min_{\mathbf{x}\in\chi_\lambda} \big(f(\mathbf{x}) + \lambda^T\mathbf{g}(\mathbf{x})\big)$. Mit $\mathbf{c}\lambda = \mathbf{g}(\mathbf{x}_\lambda))$ ist automatisch

$$\mathbf{x}_\lambda \in \arg \min_{\mathbf{x}\in\chi : g(\mathbf{x}) = c_\lambda} f(\mathbf{x}).$$

Also sei χ_λ eine beliebige Teilmenge von χ, so dass $\mathbf{x}_\lambda \in \chi_\lambda$ und

$$g_i(\mathbf{x}) \left\{ \begin{array}{ll} \leq c_{\lambda,i} & \text{, falls} \quad \lambda_i > 0 \\ \geq c_{i,\lambda} & \text{, falls} \quad \lambda_i < 0 \end{array} \right\} \quad \text{für alle } \mathbf{x} \in \chi_\lambda \text{ ist.}$$

Dann folgt

$$\mathbf{x}_\lambda \in \arg \min_{\mathbf{x}\in\chi_\lambda} f(\mathbf{x}).$$

Wenn $\mathbf{x}_\lambda$ die eindeutige Minimalstelle von $f + \lambda^T\mathbf{g}$ auf χ ist, dann ist $\mathbf{x}_\lambda$ auch die eindeutige Minimalstelle von f auf χ_λ.

Für das Ausgangsproblem, die Minimierung von f unter der Nebenbedingung $\mathbf{g} = \mathbf{c}$, ergibt sich folgende Vorgehensweise: Man minimiert $f + \lambda^T \mathbf{g}$ auf χ. Sei $\mathbf{x}_\lambda$ eine entsprechende Minimalstelle. λ wird derart bestimmt, dass $\mathbf{g}(x_\lambda) = \mathbf{c}$. In diesem Falle ist $\mathbf{x}_\lambda$ die Lösung des Ausgangsproblems. Ersetzt man die Nebenbedingung $\mathbf{g} = \mathbf{c}$ durch die komponentenweise Ungleichung $\mathbf{g} \leq \mathbf{c}$, dann muss man λ so wählen, dass $\mathbf{g}(\mathbf{x}_\lambda) = \mathbf{c}$ und $\lambda \geq 0$.

Beweis: Sei $\mathbf{y}$ ein beliebiger Punkt aus χ_λ. Angenommen $f(\mathbf{y}) \leq f(\mathbf{x}_\lambda)$, dann ist

$$\begin{aligned} f(\mathbf{y}) + \lambda^T g(\mathbf{y}) &= f(\mathbf{y}) + \sum_{i=1}^{q} \underbrace{\lambda_i g_i(\mathbf{y})}_{\leq \lambda_i c_{\lambda,i}} \\ &\leq f(\mathbf{x}_\lambda) + \sum_{i=1}^{q} \lambda_i c_{\lambda,i} \\ &= f(\mathbf{x}_\lambda) + \lambda^T g(\mathbf{x}_\lambda). \end{aligned}$$

Aus der Optimalität von $\mathbf{x}_\lambda$ folgt somit, dass $f(\mathbf{y}) = f(\mathbf{x}_\lambda)$. Wenn $\mathbf{x}_\lambda$ sogar die eindeutige Minimalstelle von $f + \lambda^T \mathbf{g}$ auf χ ist, dann folgt aus der Gleichung $f(\mathbf{y}) \leq f(\mathbf{x}_\lambda)$, dass $\mathbf{y} = \mathbf{x}_\lambda$. □

Minimierung einer quadratischen Form unter linearen Nebenbedingungen

Sei $\mathbf{A} \in \mathbb{R}^{d \times d}$ symmetrisch und positiv definit. Nun sei

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x}$$

über alle $\mathbf{x} \in \mathbb{R}^d$ zu minimieren, die die Nebenbedingung

$$\mathbf{B}\mathbf{x} = \mathbf{c}$$

erfüllen. Dabei ist $\mathbf{B}$ eine Matrix in $\mathbb{R}^{q \times d}$ mit dem Rang $q \leq d$ und $\mathbf{c}$ ist ein Vektor in $\mathbb{R}^q$.

Zu diesem Zweck minimieren wir für ein $\lambda \in \mathbb{R}^d$ die Funktion $f(\mathbf{x}) = \lambda^T \mathbf{B}\mathbf{x}$. Es gilt:

$$\begin{aligned} f(\mathbf{x}) + \lambda^T \mathbf{B}\mathbf{x} &= 2^{-1} \mathbf{x}^T \mathbf{A}\mathbf{x} + (\mathbf{B}^T \lambda)^T \mathbf{x} \\ &= 2^{-1} \left(\mathbf{x}^T \mathbf{A}\mathbf{x} + 2(\mathbf{A}^{-1}\mathbf{B}^T \lambda)^T \mathbf{A}\mathbf{x} \right) \\ &= 2^{-1} (\mathbf{x} + \mathbf{A}^{-1}\mathbf{B}^T \lambda)^T \mathbf{A} (\mathbf{x} + \mathbf{A}^{-1}\mathbf{B}^T \lambda) - 2^{-1} \lambda^T \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T \lambda. \end{aligned}$$

Dieser Ausdruck wird minimal genau dann, wenn

$$\mathbf{x}_\lambda = -\mathbf{A}^{-1}\mathbf{B}^T \lambda$$

ist. Ferner gilt

$$\mathbf{B}\mathbf{x}_\lambda = -\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T \lambda$$

und dies ist gleich c genau dann, wenn

$$\lambda = -(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T)^{-1}\mathbf{c}.$$

Folglich hat das ursprüngliche Problem die eindeutige Lösung

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{B}^T(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T)^{-1}\mathbf{c}$$

mit

$$f(\mathbf{x}) = 2^{-1}\mathbf{c}^T(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T)^{-1}\mathbf{c}.$$

Die Methode von Lagrange erscheint auf den ersten Blick wie ein Trick, der manchmal funktioniert, doch weiß man nicht recht, wann und warum. Mit Hilfe der konvexen Analysis lässt sich zeigen, dass die Methode in vielen Fällen funktionieren muss.

Satz 8. Sei χ eine offene konvexe Teilmenge, des $\mathbb{R}^d$, sei $f : \chi \to \mathbb{R}$ eine konvexe und $g : \mathbb{R}^d \to \mathbb{R}^q$ eine lineare Funktion. Angenommen für ein $\mathbf{c} \in \mathbb{R}^q$ existiert ein

$$\mathbf{x_c} \in \arg \min_{x \in \chi : \mathbf{g}(x) = \mathbf{c}} f(x).$$

Dann gibt es einen Vektor $\lambda_c \in \mathbb{R}^q$, so dass

$$\mathbf{x_c} \in \arg\max_{\mathbf{x} \in \chi} \left(f(x) + \lambda_c^T g(\mathbf{x}) \right) \textit{ gilt.}$$

Beweis: Sei k die Menge aller $\mathbf{x} \in \chi$, so dass $g(\mathbf{x}) = \mathbf{c}$. Nach der Voraussetzung sind

$$E(f) = \{(x, r) \in \chi \times \mathbb{R} : f(\mathbf{x}) \leq r\} \quad \text{und} \quad D = K \times]-\infty, f(\mathbf{x_c})[$$

nichtleere, konvexe Teilmengen von $\mathbb{R}^d \times \mathbb{R}$. Nun liefert der so genannte Trennungssatz aus der Analysis die Existenz eines Paars $(\mathbf{v}, t) \in \mathbb{R}^d \times \mathbb{R} \setminus \{(0, 0)\}$, so dass für alle $(\mathbf{x}, r) \in E(f)$, $(\mathbf{y}, s) \in D$ gilt:

$$\mathbf{v}^T\mathbf{x} + tr \geq \mathbf{v}^T\mathbf{y} + ts.$$

Der Skalar t kann nicht negativ sein, denn sonst würde die rechte Seite der Ungleichung beliebig groß, wenn $f(\mathbf{x_c}) \geq s \to \infty$. Auch der Fall $t = 0$ kommt nicht in Frage, denn anderenfalls wäre $\mathbf{v} \neq \mathbf{0}$ und $\mathbf{v}^T\mathbf{y} \geq \mathbf{v}^T\mathbf{x_c}$ für alle $\mathbf{x} \in \chi$, was der Offenheit von χ widersprechen würde. Also ist $t > 0$ und ohne Einschränkung kann angenommen werden, dass $t = 1$. Nun gilt also

$$\mathbf{v}^T\mathbf{y} + f(\mathbf{x_c}) \geq \mathbf{v}^T\mathbf{y} + f(\mathbf{x_c}) \quad \text{für alle } \mathbf{x} \in \chi, \mathbf{y} \in K. \tag{B.7}$$

Setzt man $\mathbf{x} = \mathbf{x_c}$ auf der linken Seite von (B.7)ein, dann ist

$$\mathbf{v}^T\mathbf{y} \leq \mathbf{v}^T\mathbf{x}_c \tag{B.8}$$

für alle $\mathbf{y} \in K$. Für ein hinreichend kleines $\delta > 0$ gehören alle Punkte mit $\mathbf{y} = \mathbf{x}_c \pm w$ mit $\mathbf{w} \in \mathbb{R}^d$, $g(\mathbf{w}) = 0$ und $||\mathbf{w}|| < \delta$ zu der Menge K. Setzt man diese Punkte in (B.8) ein, so zeigt sich, dass $\mathbf{v}^T\mathbf{w} = 0$ für alle Punkte $\mathbf{w}$ mit $\mathbf{g}(\mathbf{w}) = 0$. Nun sei $\mathbf{g}(\mathbf{w}) = \mathbf{B}\mathbf{w}$ mit einer Matrix $\mathbf{B} \in \mathbb{R}^{q \times d}$. Schreibt man

$$\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_q)^T$$

mit Vektoren $\mathbf{b}_1, \ldots, \mathbf{b}_q \in \mathbb{R}^d$, dann steht $\mathbf{v}$ senkrecht auf allen Vektoren aus $\{\mathbf{b}_1, \ldots, \mathbf{b}_q\}$. Das bedeutet aber, dass v eine Linearkombination der Vektoren $\mathbf{b}_1, \ldots, \mathbf{b}_q$ ist. Mit anderen Worten, $\mathbf{v} = \sum_{i=1}^q \lambda_i \mathbf{b}_i = \mathbf{B}^T\lambda$ für ein $\lambda \in \mathbb{R}^q$. Somit ist $\mathbf{v}^T\mathbf{x} = \lambda^T g(\mathbf{x})$. Aus (B.7) ergibt sich somit für $\mathbf{y} = \mathbf{x}_c$ die Ungleichung

$$f(\mathbf{x}) + \lambda^T\mathbf{g}(x) \geq f(\mathbf{x_c}) + \lambda^T g(\mathbf{x_c}) \quad \text{für alle } \mathbf{x} \in \chi.$$

□

Anhang C

Prolog-Programme für semantische Szeneninterpretationen

Der folgende Prolog-Code bildet die komplette Implementierung des semantischen Netzes zur Szeneninterpretation in Abschnitt 4.2.2. Der erste Teil beschreibt das semantische Netz, während der zweite Teil die automatisch generierten Klauseln zur Lösung des Zuordnungsproblems wiedergibt.

```
% Codierung sematisches Netz

comb(_,[]).                                    % Berechnung von Kombinationen
comb([X|T],[X|Comb]) :- comb(T,Comb).
comb([_|T],[X|Comb]) :- comb(T,[X|Comb]).

parallel(floor,floor).                         % Kodierung des semantischen Netzes
parallel(ceiling,floor).                       % Relationen für Kanten parallel
parallel(ceiling,ceiling).
parallel(floor,ceiling).
parallel(wall,wall).
parallel(X,_) :- X == nofeature.
parallel(_,X) :- X == nofeature.

orthogonal(ceiling,door).                      % Relationen für Kanten orthogonal
orthogonal(ceiling,wall).
orthogonal(floor,door).
orthogonal(floor,wall).
orthogonal(door,ceiling).
orthogonal(door,floor).
orthogonal(wall,wall).
orthogonal(wall,ceiling).
orthogonal(wall,floor).
orthogonal(X,_) :- X == nofeature.
orthogonal(_,X) :- X == nofeature.

notparallel(door,wall).                        % Relationen für Kanten notparallel
notparallel(wall,door).
notparallel(X,_) :- X == nofeature.
notparallel(_,X) :- X == nofeature.

equalheight(floor,floor).                      % Relationen für Kanten equalheight
equalheight(ceiling,ceiling).
equalheight(door,_).                           % aus Sicht der Ebenen
equalheight(wall,_).                           % aus Sicht der Ebenen
equalheight(X,_) :-  X == nofeature.
equalheight(_,X) :-  X == nofeature.
```

```
under(ceiling,floor).                                % Relationen für Kanten under
under(ceiling,wall).
under(ceiling,door).
under(door,_).                                       % aus Sicht der Ebenen
under(wall,_).                                       % aus Sicht der Ebenen
under(X,_) :-  X == nofeature.
under(_,X) :-  X == nofeature.

above(floor,ceiling).                                % Relationen für Kanten above
above(floor,wall).
above(floor,door).
above(door,_).                                       % aus Sicht der Ebenen
above(wall,_).                                       % aus Sicht der Ebenen
above(X,_) :-  X == nofeature.
above(_,X) :-  X == nofeature.

% Belegung der Pi's konsistent mit den automatisch generierten Attributen.

labelling(P0,P1,P2,P3,P4,P5) :- orthogonal(P0,P1),under(P0,P1),orthogonal(P0,P2),
                                under(P0,P2),notparallel(P0,P3),under(P0,P3),
                                orthogonal(P0,P4),under(P0,P4),parallel(P0,P5),
                                under(P0,P5),orthogonal(P1,P0),above(P1,P0),
                                parallel(P1,P2),above(P1,P2),orthogonal(P1,P3),
                                above(P1,P3),orthogonal(P1,P4),above(P1,P4),
                                orthogonal(P1,P5),above(P1,P5),orthogonal(P2,P0),
                                under(P2,P0),parallel(P2,P1),under(P2,P1),
                                orthogonal(P2,P3),under(P2,P3),orthogonal(P2,P4),
                                under(P2,P4),orthogonal(P2,P5),under(P2,P5),
                                notparallel(P3,P0),above(P3,P0),orthogonal(P3,P1),
                                above(P3,P1),orthogonal(P3,P2),above(P3,P2),
                                notparallel(P3,P4),notparallel(P3,P5),above(P3,P5),
                                orthogonal(P4,P0),orthogonal(P4,P1),
                                orthogonal(P4,P2),notparallel(P4,P3),under(P4,P3),
                                orthogonal(P4,P5),above(P4,P5),parallel(P5,P0),
                                above(P5,P0),orthogonal(P5,P1),above(P5,P1),
                                orthogonal(P5,P2),above(P5,P2),notparallel(P5,P3),
                                under(P5,P3),orthogonal(P5,P4),under(P5,P4).

consistent_labelling(P0,P1,P2,P3,P4,P5) :- labelling(P0,P1,P2,P3,P4,P5).

consistent_labelling(P0,P1,P2,P3,P4,P5) :- comb([P0,P1,P2,P3,P4,P5],[nofeature]),
                                                labelling(P0,P1,P2,P3,P4,P5).

consistent_labelling(P0,P1,P2,P3,P4,P5) :- comb([P0,P1,P2,P3,P4,P5],
                                                [nofeature,nofeature]),
                                           labelling(P0,P1,P2,P3,P4,P5).

consistent_labelling(P0,P1,P2,P3,P4,P5) :- comb([P0,P1,P2,P3,P4,P5],
                                                [nofeature,nofeature,nofeature]),
                                           labelling(P0,P1,P2,P3,P4,P5).

consistent_labelling(P0,P1,P2,P3,P4,P5) :- comb([P0,P1,P2,P3,P4,P5],
                                                [nofeature,nofeature,nofeature,
                                                 nofeature]),
                                           labelling(P0,P1,P2,P3,P4,P5).

consistent_labelling(P0,P1,P2,P3,P4,P5) :- comb([P0,P1,P2,P3,P4,P5],
                                                [nofeature,nofeature,nofeature,
                                                 nofeature,nofeature]),
                                           labelling(P0,P1,P2,P3,P4,P5).
```

Anhang D

Das Kamerasystem zur Erfassung von Texturen

Das Kamerasystem von Kurt3D besteht aus zwei USB Webcams, wahlweise TerraCAM USB Pro oder Logitech QuickCam 4000. Die Kameras sind mit manuellem Fokus ausgestattet und können mit einer maximalen Auflösung von 640 × 480 Pixeln betrieben werden. Um das vom 3D-Laserscanner erfasste Gebiet abzudecken muss jede Kamera 6 Bilder aufnehmen (vgl. Abbildung D.1, rechts). Daher sind die Webcams schwenk- und neigbar montiert. Die Bewegungen von maximal 45° werden von Servomotoren (Typ: Volz Micro-Maxx) ausgeführt (vgl. Abbildung D.1, links). Durch den Einsatz dieser hochwertigen Servomotoren ist eine gute Wiederholgenauigkeit garantiert.

D.1 Kamera-Kalibrierung

Die Kamera wird als eine Lochkamera modelliert. Sie projiziert 3D-Punkte $\mathbf{p} \in \mathbb{R}^3$ auf ein 2D-Bild, d.h. $\mathbf{p}' \in \mathbb{R}^2$. Der Zusammenhang zwischen einem 3D-Punkt $\mathbf{p}$ und seiner Projektion $\mathbf{p}'$ ist durch

$$s \begin{pmatrix} \mathbf{p}' \\ 1 \end{pmatrix} = \mathbf{A} \begin{pmatrix} & \mathbf{R} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} \qquad \text{mit} \quad \mathbf{A} = \begin{pmatrix} \alpha & \gamma & u \\ 0 & \beta & v \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{gegeben.}$$

Abbildung D.1: Das Kamerasystem von Kurt3D. Links: Das schwenk- und neigbare Kamerasystem. Rechts: Eine gescannte Szene als Punktwolke. Die Szene wird von 12 Kamerabildern abgedeckt.

A ist die Kameramatrix, die die inneren Kameraparameter enthält: Den Hauptpunkt mit den Koordinaten (u, v) und die Brennweite. **R** und **t** spezifizieren die externen Kameraparameter, d.h. eine orthonormale Rotationsmatrix und einen Translationsvektor, die das Kamera- in das Weltkoordinatensystem überführen. Zusätzlich zu dieser Abbildungsvorschrift berücksichtigt die Kalibrierung vier zusätzliche Parameter für die Verzerrung [78, 214].

Die Kamerakalibrierung erfolgt nach der Methode von Zhang [214]: Die Schlüsselidee von Zhang ist, die Schätzung der intrinsischen, extrinsischen und der Verzerrungsparameter mit Hilfe einer Schätzung korrespondierender Punkte durchzuführen. Diese 3D-nach-2D-Punktkorrespondenzen werden zunächst dazu benutzt, eine analytische Lösung zu bestimmen. Der Kalibrierungsalgorithmus berechnet eine allgemeine Homographie-Matrix $\mathbf{H} \in \mathbb{R}^{4\times4}$:

$$s \begin{pmatrix} \mathbf{p}' \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix}.$$

Die Berechnung geschieht durch das Lösen eines überspezifizierten linearen Gleichungssystems. Unter der Annahme, dass die 3D-nach-2D Punktkorrespondenzen nur kleine Fehler aufweisen, werden hierbei nur wenige Punktpaare benutzt, um die Gleichungen für **H** näherungsweise zu lösen. Anschließend nutzt der Kalibrierungsalgorithmus eine nichtlineare Optimierungstechnik, um das Ergebnis zu verbessern. Der Algorithmus verwendet ein Maximum Likelihood-Kriterium der Form

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \left|\left|\mathbf{p}'_{i,j} - \hat{\mathbf{p}}'_{i,j}(\mathbf{H}_j)\right|\right|.$$

$\mathbf{p}'_{i,j}$ sind die Projektionen der 3D-Punkte $\mathbf{p}_{i,j}$ im Kamerabild und $\hat{\mathbf{p}}_{i,j}$ sind diejenigen, die mittels **H** abgebildet wurden. Der Index i bezieht sich auf die Punkte, der Index j auf die verschiedenen Bilder. Anschließend bestimmt man aus **H** die Kameramatrix **A**, die Rotationsmatrix **R** und den Translationsvektor **t**. Wiederum wird dafür ein überbestimmtes lineares Gleichungssystem gelöst, gefolgt von einer Optimierung des Terms

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \left|\left|\mathbf{p}'_{i,j} - \hat{\mathbf{p}}'_{i,j}(\mathbf{A}, \mathbf{R}_j, \mathbf{t}_j)\right|\right|.$$

Schließlich ist die zu optimierende Form

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \left|\left|\mathbf{p}'_{i,j} - \hat{\mathbf{p}}'_{i,j}(\mathbf{A}, \mathbf{R}_j, \mathbf{t}_j, k_1, k_2, l_1, l_2)\right|\right|,$$

wobei k_1, k_2 die Parameter für die radiale Verzerrung und l_1, l_2 diejenigen für die tangentiale Verzerrung sind. Das Minimum wird durch einen Levenberg-Marquardt-Algorithmus gefunden, der den Gradientenabstieg mit der Gauss-Newton Methode kombiniert (vgl. Anhang B.4.2).

Für die Kamerakalibrierung sind die 3D-nach-2D-Punktkorrespondenzen essentiell. Sie werden mit Hilfe eines Schachbrettmusters bestimmt. Die Eckpunkte des Musters extrahiert man aus den Bildern des Schachbretts (vgl. Abbildung 4.2 links, Seite 69). Die zugehörigen 3D-Punkte gewinnt der Algorithmus ebenfalls automatisch. Dazu wird in einem 3D-Scan wie in Abschnitt 4.1.1 beschrieben, eine 3D-Fläche gematched. Trotz dieser Automatisierung ist der Kalibierungsprozess sehr aufwändig, da 12 Kamerapositionen eingestellt werden müssen. Auch ist die Gewährleistung der Stabilität des Kamerasystems wichtig. Bereits kleinere Störungen erfordern eine Neujustierung. Verschiedene Autoren schlagen deshalb kalibrierungsfreie Ansätze auf der Grundlage von Merkmalszuordnungen vor [19, 61, 184].

Abbildung D.2: Links: Schematische Darstellung des Texturierens von 3D-Szenen. Mitte und rechts: 3D-Szene mit Texturen aus zwei verschiedenen Posen betrachtet.

D.2 Texturieren von 3D-Scandaten

Um eine Textur auf einen 3D-Scan zu legen, wird dieser zunächst in ein Gitter, das aus Dreiecken besteht, umgewandelt. Dazu werden im 3D-Scan benachbarte Punkte als Dreiecke aufgefasst. Danach werden die 3D-Scanpunkte auf die Fotos projiziert, wobei die internen, externen und Verzerrungsparameter Anwendung finden. Um herauszufinden, welche der 12 Fotos die Textur enthält, projiziert der Algorithmus den 3D-Punkt zunächst auf alle Bilder. Anschließend wählt man das Foto, bei dem die 2D-Koordinaten am nächsten zum Bildmittelpunkt liegen (vgl. Abbildung D.1, rechts).

Ein OpenGL-basiertes Anzeigeprogramm schneidet die Texturen aus und „klebt“ sie auf das Dreiecksgitter. OpenGL ermöglicht das Darstellen der Szene aus verschiedenen Blickwinkeln. Abbildung D.2 zeigt verschiedene Perspektiven einer texturierten 3D-Szene.

Literaturverzeichnis

[1] Cyberware Body Scanner, http://www.cyberware.com/products/wbInfo.html, 2005.

[2] Cyrax Laserscanner, http://hds.leica-geosystems.com/products/products.html, 2005.

[3] Pmdtechnologies gmbh, http://www.pmdtec.com/inhalt/produkte/kamera.htm, 2005.

[4] Riegl Laserscanner, http://www.riegl.co.at/terrestrial_scanners/terrestrial_scanner_overview_/terr_scanner_menu_all.htm, 2005.

[5] Schmersal Laserscanner, http://www.produkte.schmersal.de/585/521/23124/range.html?lang=de, 2005.

[6] Sick Laserscanner, http://ecatalog.sick.com/Products/ProductFinder/product.aspx?finder=Produktfinder&pid=9168&lang=de, 2005.

[7] Siemens/Leutze Laserscanner, http://www2.automation.siemens.com/cd/safety/html_00/produkte/optisch_laser.htm, 2005.

[8] Swissranger 3d camera distance measurement object recognition range finding camera depth measurement http://www.swissranger.ch/, 2005.

[9] Zoller und Fröhlich Laserscanner, http://www.zf-laser.com/d_bildgebende.html, 2005.

[10] ATRV - JR Specifications, http://rescue.isr.ist.utl.pt/atrvjr.php, 2006.

[11] Galileo (Satellitennavigation), http://de.wikipedia.org/wiki/Galileo_(Satellitennavigation), 2006.

[12] Global positioning system, http://de.wikipedia.org/wiki/Global_Positioning_System, 2006.

[13] Kurt2 home site at AIS, A Mobile Platform for Research in Robotics, http://www.ais.fraunhofer.de/KURT2/, 2006.

[14] P3-AT, http://www.activrobots.com/ROBOTS/p2at.html, 2006.

[15] P3-DX, http://www.activrobots.com/ROBOTS/p2dx.html, 2006.

[16] Fraunhofer Gesellschaft AIS. The volksbot robot, http://www.ais.fraunhofer.de/BE/volksbot/, 2004.

[17] The RANSAC (Random Sample Consensus) Algorithm. `http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/FISHER/RANSAC/`, 2003.

[18] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. AVENUE: Automated Site Modelling in Urban Environments. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM '01)*, Quebec City, Canada, May 2001.

[19] H. Andreasson, R. Triebel, and W. Burgard. Improving Plane Extraction from 3D Data by Fusing Laser Data and Vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, August 2005.

[20] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proceedings of the 18th Conference on Uncertainty in AI (UAI '02)*, Edmonton, Alberta, Canada, August 2002.

[21] K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698 – 700, September 1987.

[22] S. Arya and D. M. Mount. Approximate nearest neigbor queries in fixed dimensions. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, pages 271 – 280, 1993.

[23] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An Optimal Algorithms for Approximate Nearest Neighbor Searcching in Fixed Dimensions. *Journal of the ACM*, 45:891 – 923, 1998.

[24] T. Bailey. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, University of Sydney, Sydney, NSW, Australia, 2002.

[25] Chr. Baker, A. Chr. Morris, D. Ferguson, S. Thayer, Ch. Whittaker, Z. Omohundro, C. Reverte, W. R. L. Whittaker, D. Hähnel, and S. Thrun. A Campaign in Autonomous Mine Mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 1998 – 2003, New Orleans, USA, April 2004.

[26] J.J. Belwood and R.J. Waugh. Bats and Mines: Abandoned Does Not Always Mean Empty. *Bats*, 9(3), 1991.

[27] R. Benjemaa and F. Schmitt. Fast Global Registration of 3D Sampled Surfaces Using a Multi-Z-Buffer Technique. In *Proceedings IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '97)*, Ottawa, Canada, May 1997.

[28] J. L. Bentley. Multidimensional binary search trees used for associative searchin. *Communications of the ACM*, 18(9):509 – 517, September 1975.

[29] M. Bern and P. Plassmann. Mesh generation. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier Science, 2000.

[30] K. Berns, Th. Christaller, R. Dillmann, J. Hertzberg, W. Ilg, M. Kemmann, E. Rome, and H. Stapelfeldt. LAOKOON – lernfähige autonome kooperierende Kanalroboter. *KI*, 11(2):28 – 32, 1997.

[31] P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.

[32] P. J. Besl. The free-form surface matching problem. In H. Freeman, editor, *Machine Vision for Three-Dimensional Scenes*. Academic Press, San Diego, CA, U.S.A., 1990.

[33] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *Computing Surveys*, 17:75 – 145, 1985.

[34] P. Biber, H. Andreasson, T. Duckett, and A. Schilling. 3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, Sendai, Japan, September 2004.

[35] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in dynamic environments with mobile robots. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS '02)*, Lausanne, Switzerland, September.

[36] G. Blais and D. Levine. Registration Multiview Range Data to Create 3D Computer Objects. *Pattern Analysis and Machine Intelligence*, 17(8):820 – 824, August 1995.

[37] S. Borthwick and H. Durrant-Whyte. Simultaneous Localisation and Map Building for Autonomous Guided Vehicles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, 1994.

[38] P. Boulanger, O. Jokinen, and A. Beraldin. Intrinsic Filtering of Range Images Using a Physically Based Noise Model. In *Proceedings of the 15th International Conference on Vision Interface*, pages 320 – 331, Calgary, Canada, May 2002.

[39] C. Braun, T. H. Kolbe, F. Lang, W. Schicker, V. Steinhage, A. B. Cremers, W. Förstner, and L. Plümer. Models for photogrammetric building reconstruction. *Comupter & Graphics*, 19(1):109 – 118, 1995.

[40] C. Brenneke, O. Wulf, and B. A. Wagner. Using 3D Laser Range Data for SLAM in Outdoor Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '03)*, Las Vegas, USA, October 2003.

[41] A. Brunn, R. Englert, T. Hau, A. B. Cremers, and W. Förstner. Aufnahme und Intelligente Auswertung Großflächiger Szenen. Technical report, Universität Bonn, September 1998.

[42] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI '98)*, Madison, Wisconsin, U.S.A., 1998.

[43] R. J. Campbell and P. J. Flynn. Eigenshapes for 3D Object Recognition in Range Data. In *Proceedings of the IEEE Conference Computer Vision and Pattern Recognition (CVPR '99)*, Fort Collins, CO, U.S.A., 1999.

[44] R. J. Campbell and P. J. Flynn. A Survey Of Free-Form Object Representation and Recognition Techniques. *Computer Vision and Image Understanding (CVIU)*, 81, 2001.

[45] O. I. Camps, C. Y. Huang, and T. Kanungo. Hierarchical Organization of Appearance-Based Parts and Relations. In *Proceedings of the IEEE Conference Computer Vision and pattern Recognition*, pages 685 – 691, Santa Barbara, CA, U.S.A., June 1998.

[46] H. Cantzler, R. B. Fisher, and M. Devy. Improving architectural 3D reconstruction by plane and edge constraining. In *Proceedings of the British Machine Vision Conference (BMVC '02)*, pages 43 – 52, Cardiff, U.K., September 2002.

[47] H. Cantzler, R. B. Fisher, and M. Devy. Quality enhancement of reconstructed 3D models using coplanarity and constraints. In *Proceedings of annual Symposium for Pattern Recognition (DAGM '02)*, pages 34 – 41, Zürich, Switzerland, September 2002.

[48] O. Carmichael and M. Hebert. Object Recognition by a Cascade of Edge Probes. In *Proceedings of the 13th British Machine Vision Conference (BMVC '02)*, volume 1, pages 103 – 112, Cardiff, U.K., September 2002.

[49] J. L. Chen and G. C. Stockman. 3D free-form object recognition using indexing by contour features. *Journal of Computer Vision and Image Understanding (CVIU)*, 1998.

[50] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '91)*, pages 2724 – 2729, Sacramento, CA, USA, April 1991.

[51] H. Choset and J. W. Burdick. Sensor based motion planning: The hierarchical generalized voronoi graph. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, Toulouse, France, 1996.

[52] C. S. Chua and R. Jarvis. Point signatures: A new representation for 3D object recognition. *International Journal Computer Vision*, 25:63 – 85, 1997.

[53] S. Clark. *Autonomous Land Vehicle Navigation*. PhD thesis, The University of Sydney, Sydney, Australia, 1999.

[54] P. Clausen. *Was ist was, Band 37, Computer und Roboter*. Tessloff Verlag, Nünberg, 1999.

[55] P. Corke, J. Cunningham, D. Dekker, , and H. Durrant-Whyte. Autonomous underground vehicles. In *Proceedings of the CMTE Mining Technology Conference*, pages 16–22, Perth, Australia, September 1996.

[56] Th. Cormen, Ch. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw–Hill Book Company, 1997.

[57] I. J. Cox. Blanche: Position Estimation for an Autonomous Robot Vehicle. In *Autonomous Mobile Robots: Control, Planning, and Architecture*, volume 2, pages 285–292. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[58] A. B. Cremers, J. Buhmann, W. Burgard, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S Thrun. The Mobile Robot RHINO. *AI Magazine*, 16(2):31 – 38, 1995.

[59] S. Cunnington and A. Stoddart. N-View Point Set Registration: A Comparison. In *Proceedings of the 10th British Machine Vision Conference (BMVC '99)*, Nottingham, UK., 1999.

[60] E. Dam, M. Koch, and M. Lillholm. Quaternion, Interpolation and Animation. Technical Report DIKU-TR-98/5, Department of Computer Science University of Copenhagen, Copenhagen, Denmark, 1998.

[61] P. Dias, V. Sequeira, F. Vaz, and J. G.M. Goncalves. Registration and Fusion of Intensity and Range Data for 3D Modelling of Real World Scenes. In *Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '03)*, Banff, Canada, October 2003.

[62] J. Diebel, K. Reuterswärd, S. Thrun, J. Davis, and R. Gupta. Simultaneous Localization and Mapping with Active Stereo Vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pages 3436 – 3443, Sendai, Japan, September 2004.

[63] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3):229 – 241, June 2001.

[64] C. Dorai and A. K. Jain. COSMOS — A representation scheme for 3D free form objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:1115 – 1130, 1997.

[65] C. Eberst and J. Sicheneder. Generation of Hypothetical Landmarks Supporting Fast Object Recognition with Autonomous Mobile Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '96)*, Osaka, Japan, November 1996.

[66] D. Eggert, A. Fitzgibbon, and R. Fisher. Simultaneous Registration of Multiple Range Views Satisfying Global Consistency Constraints for Use In Reverse Engineering. *Computer Vision and Image Understanding*, 69:253 – 272, March 1998.

[67] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3:249 – 265, 1987.

[68] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46 – 57, 1989.

[69] R. Englert and A. B. Cremers. Improving reconstruction of man-made objects from sensor images by machine learning. In *Proceedings of the 11th SPIE's International Symposium on AeroSense: Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision III.*, 1997.

[70] H. R. Everett. *Sensors for Mobile Robots: Theory and Application*. AK Peters, Ltd, 1995.

[71] Matrix FAQ. Version 2, `http://skal.planet-d.net/demo/matrixfaq.htm`, 1997.

[72] A. Fasano, M. Callieri, P. Cignoni, and R. Scopigno. Exploiting mirrors for laser stripe 3d scanning. In *Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '03)*, pages 243 – 251, Banff, Canada, October 2003.

[73] D. Ferguson, A. Morris, D. Hähnel, C. Baker, Z. Omohundro, C. Reverte, S. Thayer, W. Whittaker, W. Whittaker, W. Burgard, and S. Thrun. An Autonomous Robotic System for Mapping Abandoned Mines. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proceedings of Conference on Neural Information Processing Systems (NIPS)*. MIT Press, 2003.

[74] A. Fischer, T. H. Kolbe, and F. Lang. On The Use Of Geometric And Semantic Models For Component-Based Building Reconstruction. In *Proceedings of the Workshop Semantic Modeling for the Acquisition of Topographic Information from Images and Maps, (SMATI '99)*, 1999.

[75] G. Fischer. *Lineare Algebra*. Friedrich Vieweg & Sohn Verglagsgesellschaft mbH, Braunschweig / Wiesbaden, 1995.

[76] R. B. Fisher. Applying knowledge to reverse engeniering problems. In *Proceedings of the International Conference. Geometric Modeling and Processing (GMP '02)*, pages 149 – 155, Riken, Japan, July 2002.

[77] A.W. Fitzgibbon. Robust Registration of 2D and 3D Point Sets. In *Proceedings of the 12th British Machine Vision Conference (BMVC '01)*, 2001.

[78] W. Förstner. *Vorlesung Photogrammetrie I und II*. University of Bonn, 2002.

[79] D. A. Forsyth and J. Ponce. *Computer Vision*. Morgan Kaufmann, 2003.

[80] J. Frehse. *Vorlesung Infinitisimalrechnung I – IV (Vorlesungsmitschrift)*. University of Bonn, 1996 – 1997.

[81] U. Frese and G. Hirzinger. Simultaneous Localization and Mapping – A Discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pages 17 – 26, Seattle, USA, August 2001.

[82] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the 13th International Conference*, pages 148 – 156, 1996.

[83] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transaction on Mathematical Software*, 3(3):209 – 226, September 1977.

[84] S. Frintrop. *VOCUS: A Visual Attention System for Object Detection and Goal-directed search*. PhD thesis, University of Bonn, 2005.

[85] S. Frintrop, A. Nüchter, H. Surmann, and J. Hertzberg. Saliency-based Object Recognition in 3D Data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pages 2167 – 2172, Sendai, Japan, September 2004.

[86] S. Frintrop, E. Rome, A. Nüchter, and H. Surmann. An Attentive, Multi-modal Laser Eye. In *Proceedings of the third International Conference on Computer Vision Systems (ICVS '03)*, pages 202 – 211, Graz, Austria, April 2003.

[87] S. Frintrop, E. Rome, A. Nüchter, and H. Surmann. A Bimodal Laser-Based Attention System. *Journal Computer Vision and Image Understanding (CVIU), Special Issue on Attention and Performance in Computer Vision*, 100(1-2):124 – 151, October – November 2005.

[88] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7 - Proceedings of the 7th Advances in Neural Information Processing Systems (NIPS '95)*, pages 625 – 632, Cambridge, MA, USA, 1995.

[89] C. Früh and A. Zakhor. 3D Model Generation for Cities Using Aerial Photographs and Ground Level Laser Scans. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR '01)*, Kauai, Hawaii, USA, December 2001.

[90] C. Früh and A. Zakhor. Fast 3D Model Generation in Urban Environments. In *Proceedings of the 4th International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI '01)*, Baden Baden, Germany, August 2001.

[91] K. Fukuanaga and L. D. Hostetler. Optimization of k-nearest neighbor density estimation. *IEEE Transaction on Information Theory IT-19*, pages 320 – 326, 1973.

[92] N. Gelfand, S. Rusinkiewicz, and M. Levoy. Geometrically Stable Sampling for the ICP Algorithm. In *Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '03)*, page (accepted), Banff, Canada, October 2003.

[93] A. Georgiev and P. K. Allen. Localization Methods for a Mobile Robot in Urban Environments. *IEEE Transaction on Robotics and Automation (TRO)*, 20(5):851 – 864, October 2004.

[94] D. Giel. *Hologram tomography for surface topometry.* PhD thesis, Mathematisch-Naturwissenschaftliche Fakultät der Heinrich-Heine- Universität Düsseldorf, 2003.

[95] D. Giel, S. Frey, A. Thelen, J. Bongartz, P. Hering, A. Nüchter, H. Surmann, K. Lingemann, and J. Hertzberg. Ultra-fast holographic recording and automatic 3d scan matching of living human faces. In *Proceedings of the Scientific Workshop Medical Robotics, Navigation and Visualization (MRNV '04)*, Remagen, Germany, 2004.

[96] M. Golfarelli, D. Maio, and S. Rizzi. Correction of dead-reckoning errors in map building for mobile robots. *IEEE Transaction on Robotics and Automation (TRA)*, 17(1), May 2001.

[97] O. Grau. A Scene Analysis System for the Generation of 3-D Models. In *Proceedings IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '97)*, pages 221 – 228, Ottawa, Canada, May 1997.

[98] M. Greenspan and M. Yurick. Approximate K-D Tree Search for Efficient ICP. In *Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '03)*, pages 442 – 448, Banff, Canada, October 2003.

[99] M. A. Greenspan, G. Godin, and J. Talbot. Acceleration of Binning Nearest Neighbor Methods. In *Proceedings of the Conference Vision Interface*, Montreal, Canada, 2000.

[100] A. Gueorguiev, P. Allen, E. Gold, and P. Blaer. Design, Arcitecture and Control of a Mobile site-Modelling Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, San Francisco, CA, USA, April 2000.

[101] A. Haar. Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, (69):331 – 371, 1910.

[102] D. Hähnel, W. Burgard, and S. Thrun. Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot. In *Proceedings of the fourth European workshop on advanced mobile robots (EUROBOT '01)*, Lund, Sweden, September 2001.

[103] W. Hamilton. On a new Species of Imaginary Quantities connected with a theory of Quaternions. In *Proceedings of the Royal Irish Academy*, Dublin, Ireland, November 1843.

[104] J. Hart, G. Francis, and L. Kauffmann. Visualizing Quaternion Rotation. *ACM Transactions on Graphics*, 13:256 – 276, July 1994.

[105] M. Hebert, M. Deans, D. Huber, B. Nabbe, and N. Vandapel. Progress in 3–D Mapping and Localization. In *Proceedings of the 9th International Symposium on Intelligent Robotic Systems, (SIRS '01)*, Toulouse, France, July 2001.

[106] J. Hertzberg, K. Lingemann, and A. Nüchter. USARSIM – Game-Engines in der Robotik-Lehre. In *Informatik 2005 – Informatik LIVE, vol.1 (Beiträge der 35. Jahrestagung der Gesellschaft für Informatik, Bonn)*, pages 158 – 162, Bonn, Germany, September 2005.

[107] B. K. P. Horn. Closed–form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629 – 642, April 1987.

[108] B. K. P. Horn, H. M. Hilden, and Sh. Negahdaripour. Closed–form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127 – 1135, July 1988.

[109] A. Howard, D. F. Wolf, and G. S. Sukhatme. Towards 3D Mapping in Urban Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, Sendai, Japan, September 2004.

[110] C. Y. Huang, O. I. Camps, and T. Kanungo. Object Recognition using appearance-based parts and relations. In *Proceedings of the IEEE Conference Computer Vision and pattern Recognition*, pages 877 – 882, San Juan, Puerto Rico, June 1997.

[111] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters 5*, pages 15 – 17, May 1976.

[112] G. Indiveri. Kinematic Time-invariant Control of a 2D Nonholonomic Vehicle. In *Proceedings of the 38th Conference on Decision and Control, (CDC '99)*, Phoenix, USA, December 1999.

[113] P. Jensfelt and H. I. Christensen. Pose Tracking Using Laser Scanning and Minimalistic Environmental Models. *Journal of the IEEE Transactions on Robotics and Automation (TRA '01)*, 17(2):138–147, April 2001.

[114] A. E. Johnson and M. Hebert. Efficient multiple model recognition in cluttered 3-Dscenes. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR' 98)*, pages 671 – 677, Santa Barbara, CA, U.S.A., June 1998.

[115] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433 – 449, May 1999.

[116] H. A. Kestler, S. Sablatnög, S. Simon, S. Enderle, A. Baune, G. K. Kraetzschmar, F. Schwenker, and G. Palm. Concurrent Object Identification and Localization for a Mobile Robot. *KI – Künstliche Intelligenz*, (4):23–29, April 2000.

[117] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterisation of human faces. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 12:103 – 108, 1990.

[118] C. Koch and S. Ullman. Shifts in selective visual attantion: towards the underlaying visual circuitry. *Human Neurobiology*, pages 219 –227, 1985.

[119] T. H. Kolbe, L. Plümer, and A. B. Cremers. Using Constraints for the Identification of Buildings in Aerial Images. In *Proceedings of the 2. International Conference on Practical Applications of Constraint Technology (PACT '96)*, pages 143 – 154, 1996.

[120] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the twelfth national Conference on Artificial Intelligence (AAAI '94)*, July 1994.

[121] B. Kuipers and Y. T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8, 1991.

[122] I. S. Kweon and T. Kanade. High-Resolution Terrain Map from Multiple Sensor Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):278–292, 1992.

[123] S. Lazebnik, A. Sethi, C. Schmid, D. J. Kriegman, J. Ponce, and M. Hebert. On Pencils of Tangent Planes and the Recognition of Smooth 3D Shapes from Silhouettes. In *Proceedings of the 7th European Conference on Computer Vision (ECCV '02)*, pages 651 – 665, May 2002.

[124] J.J. Leonard and Hugh Durrant-Whyte. Localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(6), 1991.

[125] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anerson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proceedings of the ACM SIGGRAPH*, New Orleans, USA, July 2000.

[126] R. Lienhart and J. Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. In *Proceedings of the IEEE Conference on Image Processing (ICIP '02)*, pages 155 – 162, New York, USA, Septmber 2002.

[127] K. Lingemann. Schnelles Pose-Tracking auf Laserscan-Daten für autonome mobile Roboter. Master's thesis, University of Bonn, 2004.

[128] K. Lingemann, A. Nüchter, J. Hertzberg, and H. Surmann. About the Control of High Speed Mobile Indoor Robots. In *Proceedings of the Second European Conference on Mobile Robotics (ECMR '05)*, pages 218 – 223, Ancona, Italy, September 2005.

[129] K. Lingemann, A. Nüchter, J. Hertzberg, and H. Surmann. High-Speed Laser Localization for Mobile Robots. *Journal Robotics and Autonomous Systems*, (accepted), 2005.

[130] K. Lingemann, A. Nüchter, O. Wulf, H. Surmann, K. Pervölz, J. Hertzberg, B. Wagner, and T. Christaller. Robocuprescue - Robot League Team, Team Deutschland1. Technical report, Team Description Paper, Rescue Robot League Competition, Bremen, Germany, June 2006.

[131] K. Lingemann, H. Surmann, A. Nüchter, and J. Hertzberg. Indoor and Outdoor Localization for Fast Mobile Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pages 2185 – 2190, Sendai, Japan, September 2004.

[132] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to Learn 3D Models of Indoor Environments with Mobile Robots. In *Proceedings of the 18th Conference on Machine Learning*, Williams College, July 2001.

[133] A. Lorusso, D. Eggert, and R. Fisher. A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations. In *Proceedings of the 4th British Machine Vision Conference (BMVC '95)*, pages 237 – 246, Birmingham, England, September 1995.

[134] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV '99)*, pages 1150 – 1157, Corfu, Greece, September 1999.

[135] F. Lu and E. Milios. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR '94)*, pages 935–938, 1994.

[136] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333 – 349, October 1997.

[137] F. Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 17:539 – 544, 1995.

[138] A. Morris, D. Kurth, W. Whittaker, and S. Thayer. Case Studies of a Borehole Deployable Robot for Limestone Mine Profiling and Mapping. In *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 2003.

[139] H. Murase and S. K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal Comupter Vision*, 14:5 – 24, 1995.

[140] R. R. Murphy. Activities of the Rescue Robots at the World Trade Center from 11-21 September 2001. *IEEE Robotics & Automation Magazine*, 11(3):851 – 864, September 2004.

[141] R. R. Murphy. Rescue Robotics for Homeland Security. *Communications of the ACM, Special Issue on Homeland Security*, 27(3):66 – 69, March 2004.

[142] D. R. Musser. Introspective sorting and selection algorithms. *Software Practice & Experience*, 27(8):983–993, 1997.

[143] M. M. Nevado, J. G. Garcia-Bermejo, and E. Z. Casanova. Obtaining 3D models of indoor environments with a mobile robot by estimating local surface directions. *Robotics and Autonomous Systems*, 48:131 – 143, August 2004.

[144] A. Nüchter. *Autonome Exploration und Modellierung von 3D-Umgebungen, GMD Report 157*. GMD, Sankt Augustin, 2002.

[145] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM with Approximate Data Associoation. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR '05)*, pages 242 – 249, July 2005.

[146] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. Accurate Object Localization in 3D Laser Range Scans. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR '05)*, pages 665 – 672, July 2005.

[147] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. Heuristic-Based Laser Scan Matching for Outdoor 6D SLAM. In *KI 2005: Advances in Artificial Intelligence. 28th Annual German Conference on AI, Proceedings Springer LNAI vol. 3698*, pages 304 – 319, Koblenz, Germany, September 2005.

[148] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, K. Pervölz, M. Hennig, K. R. Tiruchinapalli, R. Worst, and T. Christaller. Mapping of Rescue Environments with Kurt3D. In *Proceedings of the IEEE International Workshop on Rescue Robotics (SSRR '05)*, pages 158 – 163, Kobe, Japan, June 2005.

[149] A. Nüchter, H. Surmann, , and J. Hertzberg. Automatic Classification of Objects in 3D Laser Range Scans. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS '04)*, pages 963 – 970, Amsterdam, The Netherlands, March 2004.

[150] A. Nüchter, H. Surmann, and J. Hertzberg. Automatic Model Refinement for 3D Reconstruction with Mobile Robots. In *Proceedings of the 4th IEEE International Conference on Recent Advances in 3D Digital Imaging and Modeling (3DIM '03)*, pages 394 – 401, Banff, Canada, October 2003.

[151] A. Nüchter, H. Surmann, K. Lingemann, and J. Hertzberg. Consistent 3D Model Construction with Autonomous Mobile Robots. In *KI 2003: Advances in Artificial Intelligence. 26th Annual German Conference on AI, Proceedings Springer LNAI vol. 2821*, pages 550 – 564, Hamburg, Germany, September 2003.

[152] A. Nüchter, H. Surmann, K. Lingemann, and J. Hertzberg. Semantic Scene Analysis of Scanned 3D Indoor Environments. In *Proceedings of the 8th International Fall Workshop Vision, Modeling, and Visualization (VMV '03)*, pages 215 – 222, Munich, Germany, November 2003.

[153] A. Nüchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6D SLAM with an Application in Autonomous Mine Mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1998 – 2003, New Orleans, USA, April 2004.

[154] A. Nüchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, , and H. Surmann. 3D Mapping with Semantic Knowledge. In *Proceedings of the RoboCup International Symposium*, Osaka, Japan, June 2005.

[155] NIST National Institute of Standards and Technology. Intelligent systems division, `http://robotarenas.nist.gov/competitions.htm`, 2005.

[156] K. Ogata. *Modern Control Engineering (4th Edition)*. Prentice Hall, November 2001.

[157] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the 6th International Conference on Computer Vision (ICCV '98)*, Bombay, India, January 1998.

[158] E. Pauley, T. Shumaker, and B. Cole. Preliminary report of investigation: Underground Bituminous Coal mine, non-injury mine inundation accident (entrapment), July 24, 2002, Quecreek, Pennsylvania, 2002. Black Wolf Coal Company, Inc. for the PA Bureau of Deep Mine Safety.

[159] E. Pervin and J. Webb. Quaternions for computer vision and robotics. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR '83)*, Washington, D.C., USA, 1983.

[160] D. Pierce and B. Kuipers. Learning to explore and build maps. In *Proceedings of the twelfth national Conference on Artificial Intelligence (AAAI '94)*, pages 1264 – 1271, Seattle, W.A., U.S.A., July 1994. American Association for Artificial Intelligence.

[161] F. Pipitone and W. Adams. Rapid recognition of freeform objects in noisy range images using tripod operators. In *Proceedings of the IEEE International Conference On Computer Vision*, pages 715 – 716, Berlin, Germany, May 1993.

[162] J. Ponce and D. J. Kriegmann. On recognizing and positioning curved 3D objects from image contours. In *Proceedings of the IEEE Workshop on Interpretation of 3D Scenes*, pages 61 – 67, Austin, TX, U.S.A., November 1989.

[163] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 7:155 – 162, 1964.

[164] E. A. Praßler and E. Milios. Position Estimation Using Equidistance Lines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '95)*, pages 85–92, Nagoya, Aichi-ken, Japan, 1995.

[165] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, January 1993.

[166] SWI Prolog. `http://www.swi-prolog.org/`, 2003.

[167] K. Pulli. Multiview Registration for Large Data Sets. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling (3DIM '99)*, pages 160 – 168, Ottawa, Canada, October 1999.

[168] R. L. Rivest. On the optimality of Elias's algorithm for performing best match searches. *Information Processing*, 74:678 – 681, 1974.

[169] T. Röfer. Building consistent laser scan maps. In *Proceedings of the 4th European Workshop on Advanced Mobile Robots (EUROBOT '01)*, volume 86, pages 83–90, 2001.

[170] E. Rome, J. Hertzberg, F. Kirchner, U. Licht, H. Streich, and Th. Christaller. Towards Autonomous Sewer Robots: the MAKRO Project. *Urban Water*, 1:57 – 70, 1999.

[171] S. Ruiz-Correa, L. G. Shapiro, and M. Meila. A New Paradigm for Recognizing 3-D Object Shapes from Range Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, Madison, U.S.A., June 2003.

[172] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modellling (3DIM '01)*, pages 145 – 152, Quebec City, Canada, May 2001.

[173] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, Inc., Upper Sanddle River, NJ, USA, 1995.

[174] A. Sappa, A. Restrepo-Specht, and M. Devy. Range Image Registration by using an Edge–based Representation. In *Proceedings of th 9th International Symposium on Intelligent Robotic Systems, (SIRS '01)*, Toulouse, France, July 2001.

[175] Jean Scholtz. Theory and Evaluation of Human Robot Interactions. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, Washington, DC, U.S.A., 2003.

[176] U. Schöning. *Logik für Informatiker*. Spektrum Akademischer Verlag, 2000.

[177] S. Se, D. Lowe, and J. Little. Local and Global Localization for Mobile Robots using Visual Landmarks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '01)*, Hawaii, USA, October 2001.

[178] V. Sequeira, J. Goncalves, and M. Ribeiro. 3D environment modelling using laser range sensing. *Robotics and Automation*, 16:81 – 91, 1995.

[179] V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, and D. Hogg. Automated 3D reconstruction of interiors with multiple scan–views. In *Proceedings of SPIE, Electronic Imaging '99, The Society for Imaging Science and Technology /SPIE's 11th Annual Symposium*, San Jose, CA, USA, January 1999.

[180] K. Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19(3):245 – 254, July 1985.

[181] D. Simon, M. Hebert, and T. Kanade. Real–time 3–D pose estimation using a high–speed range sensor. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '94)*, volume 3, pages 2235 – 2241, San Diego, CA, USA, May 1994.

[182] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167 – 193. Springer-Verlag New York, Inc., 1990.

[183] Erik Solda, Rainer Worst, and Joachim Hertzberg. Poor Man's Gyro-based Location. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '04)*, Lisabon, Portugal, July 2004.

[184] I. Stamos and P. Allen. 3-D Model Construction Using Range and Image Data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR '00)*, USA, June 2000.

[185] F. Stein and G. Medioni. Structural indexing: Efficient 3d object recognition. *IEEE Transaction on Pattern Analysis and machine Vision (PAMI)*, 14:125 – 145, February 1992.

[186] K. A. Stetson. Holographic surface contouring by limited depth of focus. *Applied Optics*, 7(5):987 – 989, 1968.

[187] S. Stiene. Konturbasierte Objekterkennung aus Tiefenbildern eines 3D-Laserscanners. Master's thesis, University of Osnabrück, January 2006.

[188] A. Stoddart and A. Hilton. Registration of multiple point sets. In *Proceedings of the 13th IAPR International Conference on Pattern Recognition*, pages 40–44, Vienna, Austria, August 1996.

[189] J. Suomela, J. Kuusela, and A. Halme. A Milimeter wave radar for close terrain mapping of an intelligent autonomous vehicle. In *Proceedings of the 2nd IFAC Conference on Intelligent Autonomous Vehicles*, pages 349 – 354, Helsinki, Finnland, 1995.

[190] H. Surmann, K. Lingemann, A. Nüchter, M. Hennig, K. Pervölz, O. Wulf, J. Hertzberg, B. Wagner, and T. Christaller. Robocuprescue - Robot League Team, Team Deutschland1. Technical report, Team Description Paper, Rescue Robot League Competition, Osaka, Japan, July 2005.

[191] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. A 3D laser range finder for autonomous mobile robots. In *Proceedings of the of the 32nd International Symposium on Robotics (ISR '01)*, pages 153 – 158, Seoul, Korea, April 2001.

[192] H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. Fast acquiring and analysis of three dimensional laser range data. In *Proceedings of the of the 6th International Fall Workshop Vision, Modeling, and Visualization (VMV '01)*, pages 59 – 66, Stuttgart, Germany, November 2001.

[193] H. Surmann and A. Morales. A five layer sensor architecture for autonomous robots in indoor environments. In *Proceedings of the International symposium on robotics and automation (ISRA '00)*, pages 533 – 538, Monterrey, N.L., Mexico, November 2000.

[194] H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor en vironments. *Journal Robotics and Autonomous Systems*, 45(3 – 4):181 – 198, December 2003.

[195] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg. 6D SLAM A Preliminary Report on Closing the Loop in Six Dimensions. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '04)*, Lisabon, Portugal, July 2004.

[196] S. Thrun. Learning Metric-Topological Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[197] S. Thrun. Robotic Mapping: A Survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[198] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots*, 31(5):1 – 25, October 1997.

[199] S. Thrun, D. Fox, and W. Burgard. A real-time algorithm for mobile robot mapping with application to multi robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, San Francisco, CA, USA, April 2000.

[200] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A System for Volumetric Robotic Mapping of Abandoned Mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '03)*, 2003.

[201] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. F. Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *Machine Learning and Autonomous Robots*, 23(7 – 8):693 – 716, July/August 2004.

[202] O. Trullier and J.-A. Meyer. Biomimetic Navigation Models and Strategies in Animats. *AI Communications*, 10(2):79 – 92, July 1997.

[203] M. Turk and P. Pentland. Eigenfaces for Recognition. *Journal Cognitive Neuroscience*, 3:71 – 86, 1991.

[204] P. Viola and M. Jones. Robust Real-time Object Detection. In *Proceedings of the second International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing and Sampling*, Vancouver, Canada, July 2001.

[205] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137 – 154, May 2004.

[206] K. Völker, editor. *Künstliche Menschen*. Carl Hanser Verlag, München, 1971.

[207] M. W. Walker, L. Shao, and R. A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Understanding*, 54:358 – 367, November 1991.

[208] D. Waltz. Understanding line drawings of scenes with shadows. In Patrick Henry Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[209] D. L. Waltz. *Understanding line drawings of scenes with shadows*. McGraw-Hill, New York, U.S.A., 1975.

[210] G. Weiß, C. Wetzler, and E. von Puttkamer. Keeping Track of Position and Orientation of Moving Indoor Systems by Correlation of Range-Finder Scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '94)*, pages 595–601, Munich, Germany, 1994.

[211] O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 4204 – 4209, New Orleans, USA, April 2004.

[212] O. Wulf, B. A. Wagner, and M. Khalaf-Allah. Using 3D data for Monte Carlo localization in complex indoor environments. In *Proceedings of the second European Conference on Mobile Robotics*, pages 170 – 175, Ancona, Italy, September 2005.

[213] Z. Zhang. Iterative point matching for registration of free–form curves. Technical Report RR-1658, INRIA–Sophia Antipolis, Valbonne Cedex, France, 1992.

[214] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(22):1330 – 1334, 2000.

[215] H. Zhao and R. Shibasaki. Reconstructing Textured CAD Model of Urban Environment Using Vehicle-Borne Laser Range Scanners and Line Cameras. In *Second International Workshop on Computer Vision System (ICVS '01)*, pages 284 – 295, Vancouver, Canada, July 2001.

[216] U. R. Zimmer. Robust world-modelling and navigation in a real world. *NeuroComputing*, 13(2 - 4):247 – 260, 1996.