UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

Arbeitsgruppe Aktives Sehen

Bildbasiertes, aktives 3D-Laserscannen

Diplomarbeit

zur Erlangung des Grades eines

Diplom-Informatikers
im Studiengang Computervisualistik

vorgelegt von

Stefan Bröhl

Betreuer:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Erstgutachter:	Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik
Zweitgutachter:	Dipl.-Inf. Johannes Pellenz, Institut für Computervisualistik, Fachbereich Informatik

Koblenz, im September 2007

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüufungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Arbeitsgruppe für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☐ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ nein ☐

Koblenz, den ..... Unterschrift

Danksagung

Ein Bildungsabschnitt geht an dieser Stelle zu Ende und es wird Zeit sich auf neue Aufgaben und Wege einzulassen. Bis hierhin war es ein aufregender und interessanter, aber nicht immer ganz einfacher Weg. Ohne die Unterstützung und Hilfe von einigen Menschen, wäre es auf diese Wiese nicht möglich gewesen.

An dieser Stelle möchte ich mich ganz besonders bei folgenden Menschen bedanken:

Meinen Eltern, die stets für mich da sind und mich in meinen Zielen bestärken und unterstützen. Meinen Freunden, die inzwischen auf der ganzen Welt verstreut sind und einen wichtigen Teil meines Lebens, auch außerhalb der Arbeit, einnehmen. Außerdem Prof. Dr. Dietrich Paulus und Johannes Pellenz für die kompetente Betreuung bei offenen Fragen und Entwicklung von neuen Ideen, sowie Dr. Martin Fislake und Ullrich Düning bei der handwerklichen Unterstützung und Benjamin Durth für das Korrekturlesen. Abschließend noch einen Dank an “Robbie-8”, der immer wieder auf einige seiner Komponenten, zumindest zeitweise, verzichten musste und hin und wieder umgebaut worden ist.

Zusammenfassung

Die Selbstlokalisation von Robotern ist schon seit Jahren ein aktuelles Forschungsthema, das insbesondere durch immer weiterentwickelte Techniken und Verfahren verbessert werden kann. Insbesondere finden Laserscanner in der Robotik immer häufiger Anwendung. In dieser Arbeit wird untersucht, ob durch die Fusionierung von Kamerabildern und 3D-Laserscannerdaten eine robuste und schnelle Selbstlokalisation theoretisch sowie praktisch realisierbar ist. Bei der Entwicklung dieses Systems wird darauf geachtet, einen möglichst kompakten und kostengünstigen Aufbau zu realisieren, der stets individuell anpassbar bleibt. Aus diesem Grund wird kein kommerzieller 3D-Laserscanner verwendet sondern eine Eigenentwicklung, bestehend aus einem herkömmlichen 2D-Laserscanner und einer Rotationsplattform; diese wird mit Hilfe eines Servomotors, der über einen Mikrocontroller angesteuert wird, rotiert. Um die Positionsänderung im Raum berechnen zu können, sind trackbare Merkmale aus Bildern zu extrahieren, die daraufhin aktiv vom Laserscanner angesteuert und vermessen werden. Aus den damit gewonnenen 3D-Weltkoordinaten der Bildmerkmale kann, wie in der visuellen Odometrie bereits gezeigt, die relative Position im Raum berechnet werden. Probleme können hierbei sowohl bei schlecht trackbaren Merkmalen auftreten sowie bei einer zu ungenauen Bestimmung der 3D-Weltkoordinaten durch die Vermessung mittels Laserscanner.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Selbstlokalisation ist bei heutigen mobilen Systemen eine wichtige Komponente. Zur zentralen Problemstellung, der Positionsbestimmung im Raum, existieren verschiedene Ansätze; einer davon ist beispielsweise die Bestimmung über die Radumdrehungen, die beim Auto oder beim Fahrrad verwendet wird, um den zurückgelegten Weg festzustellen. Probleme treten hierbei durch Reibung und unebene Flächen auf, da bei zu niedriger Bodenhaftung die Räder durchdrehen. Es soll also nicht der zurückgelegte Weg ermittelt werden, sondern die relative Position zur Startposition. Ein weiterer Ansatz ist die der visuellen Odometrie. Hierbei wird die Position unabhängig von der Radumdrehung allein durch die Mittel der Bildverarbeitung bestimmt. Die Schwierigkeit liegt jedoch darin, eine optimale Balance zwischen Geschwindigkeit und Robustheit zu erreichen.

1.1 Aufgabenstellung

Das Ziel dieser Arbeit liegt in der Untersuchung eines Verfahrens zur Selbstlokalisation, das einerseits eine genaue Positionsbestimmung ermöglicht und andererseits schnell und robust ist. Ein Ansatz, der dies ermöglichen soll, ist die Fusionierung von 3D-Laserdaten mit Kamerabildern. Im Detail bedeutet dies, in einem ersten Schritt gut trackbare Merkmale aus den Kamerabildern zu extrahieren und diese vom Laserscanner anzusteuern sowie vermessen zu lassen. Im zweiten Schritt ist mit Hilfe der ermittelten Daten die dazuge-

hörige 3D-Weltkoordinate zu berechnen; diese soll im letzten Schritt die Berechnung der relativen Position ermöglichen.

1.2 Motivation

Das zu Grunde liegende Anwendungsszenario ist das Universitätsprojekt “Robbie” im Bereich der Robotik, welches seit 2001 jedes Jahr mit bestimmten Anwendungsschwerpunkten weiterentwickelt wird; diese sind um nur einige Beispiele zu nennen “Robbie der Empfangsroboter”, “Robbie der Fußballspieler” oder “Robbie der Lebensretter”, siehe Bild 1.1. Nach erster theoretischer und praktischer Auseinandersetzung mit dem Projekt Robbie wird schnell klar, dass im Bereich der Selbstlokalisation noch Verbesserungsbedarf besteht. Im Rahmen des RoboCup-Wettbewerbs bestätigten sich die anfänglichen Annahmen in dem Sinne, dass mit einer genaueren, robusteren Positionsbestimmung ein noch besseres Gesamtergebnis hätte erzielt werden können. Die Aufgabe besteht nun grob darin, bereits existierende Verfahren durch eine Kombination zu verbessern oder lediglich zu beschleunigen. Die Art, womit dies erreicht werden soll, ist die Fusionierung von Kamera- und Laserscannerdaten. In dieser Form besteht bislang keine bekannte Umsetzung, weshalb es erforderlich ist, diese auf ihre Realisierbarkeit hin zu überprüfen. Persönliches Ziel ist die theoretische wie auch praktische Auseinandersetzung mit dieser Aufgabe, die zugleich mehrere Ebenen der Informatik wie das Ansprechen von Hardware bis hin zu softwaretechnischer Integration in ein bestehendes Framework beeinhaltet.

Bild 1.1: Robbie der Uniroboter: Vom Fußballspieler, Containerfahrer zum Lebensretter.

1.3 Aufbau der Arbeit

Kapitel 2 stellt die derzeitigen Methoden und Probleme der Selbstlokalisation anhand von verschiedenen Verfahren vor. Es werden sowohl klassische als auch bildbasierte Verfahren behandelt. Der Einsatzzweck bestimmt dabei die zu verwendete Methode, wie exemplarisch an Beispielen der Robotik und Computergrafik gezeigt wird.

In **Kapitel 3** werden die Komponenten zum bildbasierten 3D-Laserscannen vorgestellt. Der Ansatz hierzu wird sowohl auf die theoretische als auch praktische Umsetzung hin untersucht.

Die Implementierung des bildbasierten 3D-Scannens wird in **Kapitel 4** angesprochen. Die Ansteuerung der Hardware wird vor der Integration in das Robbie-8 Framework in einem davon unabhängigen System getestet.

In **Kapitel 5** werden Experimente hinsichtlich Geschwindigkeit und Genauigkeit, bezogen auf die Realisierbarkeit des bildbasierten 3D-Laserscannens, durchgeführt und analysiert.

Abschließend werden die Ergebnisse und die daraus gewonnenen Erkenntnisse in **Kapitel 6** zusammengefasst sowie über mögliche Verbesserungen und Weiterentwicklungen diskutiert; ausschlaggebend sind diese Erkenntnisse für die weitere Forschung im Bereich des bildbasierten 3D-Laserscannens bezogen auf die Selbstlokalisation.

Kapitel 2

Lokalisation und Kartenerstellung mit mobilen Systemen

Mobile Systeme wie Handys, Navigationssysteme und Spielzeugrobotor sind seit mehreren Jahren in vielen Haushalten vertreten. Immer leistungsfähigere Technik kann auf kleinstem Raum platziert werden, womit immer komplexere Systeme hergestellt werden können. Pionierarbeit leistete Sony 1999 bei der Einführung von Aibo[1], einem der ersten künstlichen Haustierroboter. Wenige Jahre später wurde von der gleichen Firma ein humanoider Roboter, Qrio[2] entwickelt, der mit Hondas ASIMO[3] verglichen werden kann. Neben der Entwicklung von künstlichen Haustieren öffnete sich der Markt für weitere Innovationen wie etwa dem Staubsaugerroboter Trilobite oder der immer wieder aufgegriffenen Idee des selbstfahrenden Autos.

Das autonom navigierende Automobil „Spirit of Berlin“, ein selbstfahrender Mittelklasse-Wagen, ist ein aktuelles Projekt der Freien Universität Berlin [Spi], in welchem die neueste Technologie zur Navigation zum Einstz kommt, siehe Abbildung 2.1. Die vollautonome Steuerung des Wagens wird möglich durch die Auswertung von Informationen aus den verschiedensten technischen Komponenten wie zum Beispiel eines Navigationssystems

[1] Artificial Intelligence Robot
[2] Quest for Curiosity
[3] Advanced Step in Innovative Mobility

Bild 2.1: “Spirit of Berlin”, ein selbstfahrender Mittelklasse-Wagen. Quelle: [Spi].

sowie verschiedenen Lasersensoren und Videokameras. Zur Orientierung dient ein 3D-Laserscanner mit einer Reichweite von bis zu 150 Metern, dessen Laserstrahlen von der Umgebung reflektiert und von Sensoren gemessen werden. Eine daraus generierte Karte veranschaulicht, wo und in welchem Abstand sich sowohl bewegliche als auch statische Objekte zum Auto befinden. Die Position des Autos wird über ein GPS-Signal ermittelt, das bei Signalausfall mittels Informationen aus Kreiselinstrumenten und Beschleunigungssensoren geschätzt wird. Die Fehlertoleranz liegt hier bei maximal einem Meter. Fahrbahnmarkierungen und Fußwege werden mit Hilfe von Videokameras ermittelt, wodurch eine Fahrgeschwindigkeit von bis zu $30km/h$ erreicht werden kann. In den kommenden Jahren gilt es, diese auf $60km/h$ zu verdoppeln sowie die Objekterkennung von beispielsweise Ampelzeichen oder auch Verkehrsschildern [Gol07], [Spi07] zu verbessern.

2.1 Problemstellung

Wie das Beispiel vom selbstfahrenden Auto zeigt, kommen dazu oft mehrere unterschiedliche Methoden zum Einsatz; diese haben dabei jeweils individuelle Eigenschaften, die

möglichst optimal einzusetzen sind. Folgende Fragen sollten bei der Auswahl der geeigneten Methode berücksichtigt werden:

- Wird ein echtzeitfähiges System benötigt?
- In welcher Umgebung (innen, außen) wird das System eingesetzt?
- Wie ist die Beschaffenheit des Bodens hinsichtlich Neigung und Unebenheiten?
- Welchen Bereich (Reichweite) hat das System zu überwachen?
- Werden die Messungen bei Bewegung oder bei Stillstand registriert?
- Wie hoch darf der maximale Fehler bei der Positionsbestimmung sein?
- Welche Komponenten nehmen Einfluss auf die System- und Ablaufgeschwindigkeit?
- Kann bei Sensorausfall die Position aus bereits bekannten Werten geschätzt werden?
- Inwieweit können bereits gemessene Werte als Wissensbasis den Ablauf optimieren?

Anhand des Anwendungsszenarios “Robbie”, siehe Abschnitt 2.4.2, werden im Bereich der Robotik mögliche Antworten auf diese Fragen gegeben und bewertet.

Ein generelles Problem von mobilen Systemen ist die Minimierung des maximalen Fehlers wie etwa bei der Positionsbestimmung; dies stellt eine besonders große Herausforderung für echtzeitfähige und sich schnell fortbewegende Systeme dar. Hierbei gilt es, je nach Anwendungsgebiet, eine möglichst ausgewogene Balance zwischen Geschwindigkeit und Genauigkeit zu finden. Der Funktionsumfang wie auch das Spektrum der Einsatzgebiete und die damit verbundenen Stärken und Schwächen sind je nach Verfahren sehr unterschiedlich. Im Folgenden werden sowohl einige klassische als auch bildbasierte Methoden zur Selbstlokalisation diesbezüglich näher untersucht.

2.2 Klassische Methoden der Selbstlokalisation

Die klassichen Methoden zur Selbstlokalisation unterscheiden sich in Funktionsweise und verwendeter Technik, die oft aus verschiedenen Einzelkomponenten besteht. Die Methoden selbst lassen sich in globale und lokale Verfahren unterteilen. Ein Gesamtsystem kann als kompakt bezeichnet werden, wenn alle dazugehörigen Komponenten zusammenhängend platziert sind, wie beispielsweise bei einem mobilen Roboter. Im Gegensatz dazu benutzt ein verteiltes System wie GPS, siehe 2.2.2, weit voneinander entfernte Komponenten wie Bodenempfänger und Satellit. Je nach Verfahren wird eine absolute oder relative Position zurückgegeben. Im Folgenden werden einige der klassischen Methoden beschrieben, welche in den unterschiedlichsten Bereichen wie Navigation, Kommunikation und Transportwesen eingesetzt werden.

2.2.1 Odometrie

Odometrie oder auch Hodometrie stammt aus dem Griechischen und bedeutet „Wegmessung“. Ein Odometer zeigt an, welche Wegstrecke s ein Objekt wie beispielsweise ein Auto zum Zeitpunkt t zurückgelegt hat. Bei der Berechnung in Gleichung 2.1 wird die Anzahl n der Radumdrehungen mittels Inkrementalgebern und dem Radumfang d bestimmt.

$$s = n * d * \pi \tag{2.1}$$

Man erhält auf diese Weise aber nur dann eine Information über die Position des Autos, wenn es sich geradlinig fortbewegt. Bei Richtungsänderung und damit Änderung des Lenkwinkels β pro Zeit t ergibt sich die Position P wie folgt

$$P(t) \quad = \quad (x(t), y(t), \alpha(t))^T \tag{2.2}$$

mit aktueller Orientierung α.

$$P(t+1) = \begin{pmatrix} x(t) & + & s(t)cos(\beta) \\ y(t) & + & s(t)sin(\beta) \\ \alpha(t) & + & \beta(t) \end{pmatrix} \tag{2.3}$$

Hieraus wird schnell ersichtlich, dass es sich bei der aktuellen Positionsbestimmung mit Hilfe der Odometriedaten um akkumulierte Positionen handelt. Da es keinen festen Bezugspunkt gibt, spricht man hier auch von einer relativen und zugleich lokalen Selbstlokalisation. Veranschaulicht am Beispiel des Autos bedeutet dies, dass eine Veränderung des Standorts, ohne selbsttätige Fortbewegung, keine Positionsänderung bewirkt.

2.2.2 GPS

Das Global Positioning System - allgemein bekannt als GPS - ist ein satellitengestütztes Navigationssystem. Im Gegensatz zur Odometriemethode ist es ein globales und auch absolutes Verfahren. Es sollte so konzipiert werden, dass zu jeder Zeit folgende Informationen - von jeder Position auf der Erde - verfügbar sind [Reu03]:

- Die aktuelle (globale) 3D-Position, also neben Längen- und Breitengrad auch die Höhe.
- Ein Zeitstempel mit Datum und Uhrzeit.
- Höhe der momentanen Fortbewegungsgeschwindigkeit.
- Die aktuelle Ausrichtung (NORD, SÜD, WEST, OST).

Das GPS-System besteht aus drei Basiskomponenten: einem Kontrollsystem, einem Satelliten, sowie einem GPS-Empfänger. Diese Komponenten können natürlich beliebig erweitert werden. Die möglichen Kommunikationswege, siehe Abbildung 2.2, sind abhängig von der jeweiligen Komponente. Das Kontrollsystem kann sowohl Daten an die Satelliten verschicken als auch empfangen, was notwendig ist, da die Berechnungen der Daten durch dieses erfolgen. Die GPS-Empfänger, die an unterschiedlichen Standorten wie etwa

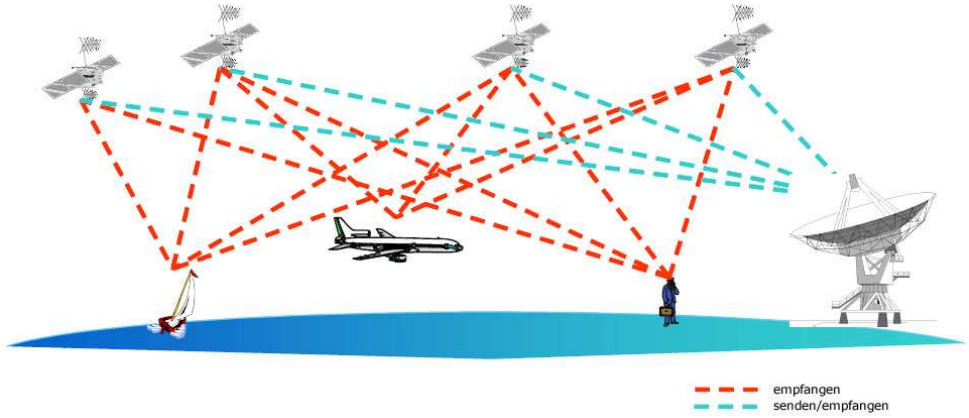

Bild 2.2: Kommunikationswege beim Global Positioning System (GPS). Quelle: [Reu03].

im Flugzeug, Segelschiff oder im Auto eingesetzt werden, empfangen ihr Signal von den Satelliten.

Die Position eines Empfängers lässt sich aus den Laufzeiten der verschieden codierten Signale der GPS-Satelliten berechnen. Es sind dabei Signale von mindestens drei unterschiedlichen Satelliten notwendig. Bei einer nicht eindeutig bestimmbaren Position ist ein vierter Satellit notwendig, dieser korrigiert Ungenauigkeiten bei der Zeitmessung, um die Daten zur Berechnung der Position zu synchronisieren [Reu03]. Für Informationen über den genauen Aufbau eines GPS-Signals sowie den Systemablauf siehe [Alo].

Folgende Faktoren [Reu03] können bei der Positionsbestimmung Fehler hervorrufen:

- Satelliten- und Empfängeruhrfehler
- Satellitenbahnfehler
- Ionosphärische/Troposphärische Refraktion
- Mehrwegeffekte
- Abschattung der Signale durch Horizontüberhöhung

Die Genauigkeit des GPS-Systems von 10 bis 15 Metern reicht bei vielen heutigen Anwendungsgebieten nicht mehr aus. Daher wurde bei der Weiterentwicklung, dem Differentiellen Global Positioning System (DGPS) [DGP], welches auf dem GPS-Grundprinzip aufbaut, versucht, die zuvor genannten Fehlereinflüsse zu umgehen. Eine Ortsgenauigkeit von unter einem Meter ist durch diese geringe Modifizierung des vorhanden Systems möglich.

2.2.3 Proximity Sensing

Proximity Sensing beruht auf der Idee, dass die Positionen von mehreren verteilten Empfängern bekannt sind. Die ungefähre Position ergibt sich dann aus der Koordinate zur nächstgelegenen Empfangsantenne. Natürlich hängt die Positionsgenauigkeit vom Abstand der Antennen zueinander ab. Für zellenbasierte Systeme stellt dieses Verfahren die Grundlage dar und wird bei Mobilfunkdiensten wie zum Beispiel GSM und UMTS eingesetzt. In der Robotik, siehe 2.4.2, wird diese Methode dazu verwendet um festzustellen, welchem Objekt der Roboter am Nächsten ist oder wie weit ein bestimmtes Objekt entfernt ist. Hierdurch kann vermieden werden, dass der Roboter mit Objekten kollidiert [Pro].

2.2.4 Laserscan Matching

Scan-Matching ist ein Verfahren, das versucht, ein Matching zwischen einem Referenz-Scan und einem aktuellen Scan[4] zu finden. Der aktuelle Scan ist so zu transformieren, dass eine maximale Überlappung mit dem Referenz-Scan ensteht. Die damit gefundene Transformation entspricht genau der Positionsänderung des Roboters zwischen den Zeitpunkten der beiden Scans und berechnet sich je nach Ansatz unterschiedlich. Ein Ansatz ist die Verwendung eines Belegtheitsgitters wie in Abbildung 2.3 dargestellt. Die Auflösung des Gitters ist dabei proportional zur Genauigkeit und Geschwindigkeit. Die Berechnung der Transformation nach [ME85] erfolgt über eine Fitnessfunktion wie in [Lin04] beschrieben; diese terminiert dabei im Maximum und liefert die bestmögliche Translation in x- und y-Richtung sowie einen Winkel, um die der aktuelle Scan rotiert worden ist. Da diese

[4] z. B. Daten von einem Ultraschall- oder Laserscanner

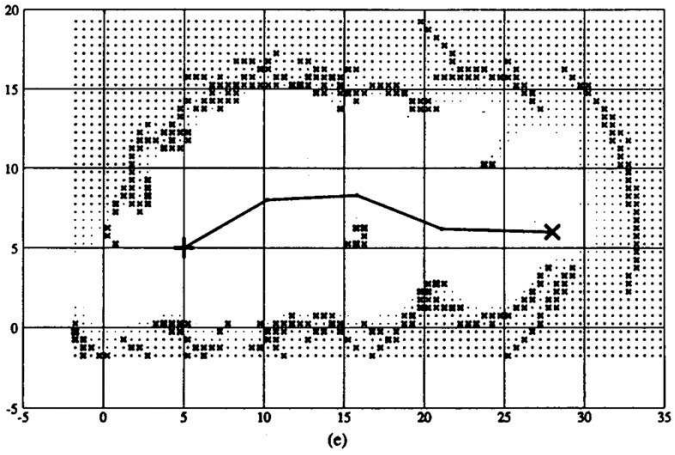

Bild 2.3: Belegtheitsgitter eines zweidimensionalen Scans. Eingezeichnet sind freie, belegte sowie unsichere Bereiche. Quelle: [ME85].

iterative Berechnung bei hoher Auflösung sehr rechenintensiv ist, kann nur ein Ausschnitt eines gesamten Gitters ausgewählt werden. Weitere Ansätze basieren auf Höhenlinien, Histogrammen, direkten Punktvergleichen, sowie Linien [Lin04].

Simultaneous Localization and Mapping (SLAM) ist eines der fundamentalen Probleme, die es in der Robotik zu lösen gilt, um echte autonome mobile Roboter zu entwickeln. SLAM bedeutet nichts anderes als die gleichzeitige Bestimmung der aktuellen Position und Konstruktion einer Umgebungskarte. Worin besteht hier nun die Schwierigkeit? Was bei der Berechnung nur eingeschränkt zur Verfügung steht ist die Zeit, in der die Daten berechnet werden müssen. Dies stößt einerseits an Grenzen der verwendeten Hardware und andererseits an die Geschwindigkeit des verwendeten Matching Verfahrens. Inzwischen konnte dieses Problem auf ebenen, horizontalen Böden in Innenräumen gelöst werden. Bei Veränderung der Rahmenbedingungen durch freies, unebenes Gelände und steigender Fortbewegungsgeschwindigkeit des Roboters gilt es jedoch, bessere Verfahren zu finden. Hierbei ist ein Kompromiss zwischen Genauigkeit von Position und Karte und Zeitverbrauch zu finden. Durch steigende Rechenleistungen und weiter entwickelter Sensoren wird die Forschung in dieser Richtung vereinfacht [GDNC+01]. In seiner Arbeit zeigt [SEGL05], dass auch visuell basierte SLAM Ansätze realisierbar sind.

2.3 Bild-basierte Selbstlokalisation

Neben den bereits beschriebenen Verfahren gibt es auch rein visuelle, kamerabasierte Verfahren, die unter dem Begriff „Visuelle Odometrie“ zusammengefasst werden. Durch die Verwendung von Kameras ist man in der Lage, wie der Mensch auch, die Umgebung durch Farb- und Helligkeitsunterschiede wahrzunehmen. Durch die Bewegung von Objekten in der Szene oder der Kamera selbst entsteht ein räumlicher Eindruck. Die Bestimmung des Ortes ist dabei sowohl mit einer als auch mit zwei Kamerasystemen möglich. Aus den Bilddaten werden Merkmale extrahiert, diese können mit einem bekannten Muster verglichen oder als ein Muster in einer Wissensbasis erfasst werden. Im Folgenden werden diese beiden Verfahren näher erläutert.

2.3.1 Markerbasierte Verfahren

Ein Marker ist ein generiertes Muster was möglichst einfach in Bildern wiedergefunden werden kann. Der Strich- oder Barcode wird zum Beispiel im Einzel- und Großhandel dazu verwendet, um Produkte zu identifizieren. RFID[5]-Tags sind die neuere Generation von Barcodes und stellen in Bezug auf die Ortungsgenauigkeit neue Möglichkeiten dar. In [NLLP04] werden diese Art von Markern zur Lokalisation in Innenräumen verwendet. RFID-Tags können nicht von Kameras erfasst werden sondern nur mit Lesegeräten, die Wellen gewisser Frequenzen aussenden und empfangen können. Abbildung 2.4 zeigt die Anwendung eines kameragestützten Systems, welches an den Wänden angebrachte Marker erkennt, aus denen sich die geometrischen Gegebenheiten der Umgebung und die aktuelle Position des Benutzers ermitteln lässt. Beim Roboterfußball werden hingegen Säulen und Spielfeldmarkierungen als Marker eingesetzt. Findet die Selbstlokalisation in Innenräumen oder in einer überschaubaren Umgebung geringeren Ausmaßes statt, so reichen markerbasierte Verfahren insbesondere hinsichtlich Markerplatzierung aus. Bei Änderung der Bedingungen ist es jedoch sinnvoll, ein markerloses Verfahren einzusetzen. Eine praxisnahe Anwendung findet sich in [Wie04] und [KLK+02].

[5] Funk-Frequenz-Identifizierung

Bild 2.4: Markerbasierte Lokalisation. Quelle: [KLK+02].

2.3.2 Markerlose Verfahren

Markerlose Verfahren ersparen die Platzierung und manuelle Generierung von Markern; diese werden hierbei durch die Gewinnung von Features (Merkmale) aus Bildern ersetzt. Ein Feature beschreibt die Struktur eines Bildes wie zum Beispiel die Position, Farbe und Helligkeit eines Pixels, eine Kante oder auch komplexere Strukturen wie die eines Objektes. Aus jedem Bild lassen sich somit Features extrahieren. Je nach Anwendungszweck werden unterschiedliche Anforderungen an die zu gewinnende Strukturinformation gestellt. Der Moravec-Operator, der Kanade-Lucas-Tomasi Operator (KLT) sowie der SIFT-Operator sind Beispiele für solche Feature-Extraktoren [Ros06]. Arbeiten, die sich mit

markerlosen Verfahren beschäftigen, sind [Dic06], [RD06] und [Str01].

Beide Verfahren werden bereits in der Augmented Reality (AR) eingesetzt und unterstützen die Software dabei, die reale Welt mit der virtuellen zu verschmelzen. Die Gewinnung von Informationen über Position und Lage im Raum sind dabei unerlässlich.

2.4 Anwendungsgebiete

2.4.1 Computergrafik

Die Computergrafik nutzt die Verfahren der Lokalisation beispielsweise zur 3D-Modellierung. Beliebige Objekte werden dabei mit einem 3D-Scanner abgetastet und unter Verwendung einer Kamera als farbiges 3D-Modell am Computer erstellt. In der Augmented Reality, einem Teilbereich der Computergrafik und Bildverarbeitung, werden Informationen über die aktuelle Lage im Raum benötigt, um virtuelle Objekte in einer realen Szene platzieren zu können. Beim Motion-Capturing hingegen wird die Bewegung einer endlichen Menge von Markerpositionen registriert, um daraus möglichst realitätsnahe Bewegungungen simulieren zu können.

2.4.2 Robotik

In der Robotik ist das Wissen über die aktuelle Position Grundlage für viele Verfahren, insbesondere für das Gebiet der Navigation. Hierbei spielt es jedoch keine Rolle, ob es sich um einen autonomen Staubsauger [RO], einen Rollstuhlroboter [LR02] oder allgemein einen Serviceroboter handelt. Je nach Aufgabengebiet gibt es unterschiedliche Gewichtungen und Anforderungen, die an das System gestellt werden und zu erfüllen sind. Eine Positionsungenauigkeit von unter einem Zentimeter oder die Anforderung echtzeitfähig zu sein sind Beispiele hierfür. Oft muss auch ein Kompromiss gefunden werden zwischen Robustheit, Genauigkeit und Geschwindigkeit; bei Letzterem gibt es zwei unterschiedliche Anforderungen, die ab einem gewissen Punkt nicht mehr erfüllt werden können. Auf der einen Seite die Zeit, die das System für die Berechnung braucht und auf der anderen Seite

die Fortbewegungsgeschwindigkeit des Roboters. Gewisse Berechnungen können nur gemacht werden wenn der Roboter sich nicht fortbewegt, wodurch die Echzeitfähigkeit stark eingeschränkt wird.

In Abschnitt 2.1 wurden Fragen formuliert, die anhand des Universitätsprojektes “Robbie-8” im Folgenden konkret beantwortet werden und damit zugleich die Anforderungen an das System festlegen: Robbie-8 ist ein echtzeitfähiges Rettungsrobotersystem, das in der Lage ist, in einer unbekannten Umgebung Opfer zu finden, die in einer selbsterstellten Karte verzeichnet werden. Die Umgebung, ein Labyrinth, siehe Abbildung 2.5, zeigt ein nachgebautes Verschüttetengebiet, was bislang in Innenräumen, insbesondere bei Wettkämpfen wie dem RoboCup German Open 2007 in Hannover und der RoboCup Weltmeisterschaft 2007 in Atlanta, stattfand. Das System ist jedoch auch in Außenbereichen einsetzbar, verzichtet aber auf die Verwendung von GPS-Daten, da diese nicht immer zur Verfügung stehen. Zur Positionsbestimmung wird daher ein Laserscanner eingesetzt, der fortlaufend Messwerte liefert mit Hilfe derer die Karte erstellt wird. Durch die Unebenheiten des Bodens ist es notwendig, die Lage des Laserscanners zu korrigieren um Fehlmessungen zu vermeiden. Der Laserscanner besitzt eine ungefähre Reichweite von 4 Metern die bislang ausreichend war. Ab einer gewissen Fortbewegungsgeschwindigkeit reicht dies nicht mehr aus. Um die Gesamtgeschwindigkeit des Systems zu optimieren und nicht zu verlangsamen wird versucht, möglichst wenig Messungen im Stillstand durchzuführen. Je geringer der Bewegungsspielraum vom Roboter ist (Abstand zu Gegenständen), desto genauer muss die Positionierung der Wände in den Karten sein damit der Roboter nicht gegen eine Wand fährt, die laut Karte nicht existiert. Hieraus wird ersichtlich, dass nur eine gute Navigation möglich ist, wenn die Pfadberechnungen genau sind. Bei Sensorausfall kann Robbie sich bis zu einem gewissen Punkt weiter fortbewegen, da die Änderung der Position auch mit Hilfe der Odometriedaten ermittelt wird. Die Position auf der Karte kann daher aus der alten Position und der zwischenzeitlichen Änderung geschätzt werden. Durch eine präzisere Positionsgenauigkeit kann das komplette System nachhaltig verbessert werden, wobei jedoch die Berechnungsgeschwindigkeit nicht vernachlässigt werden darf.

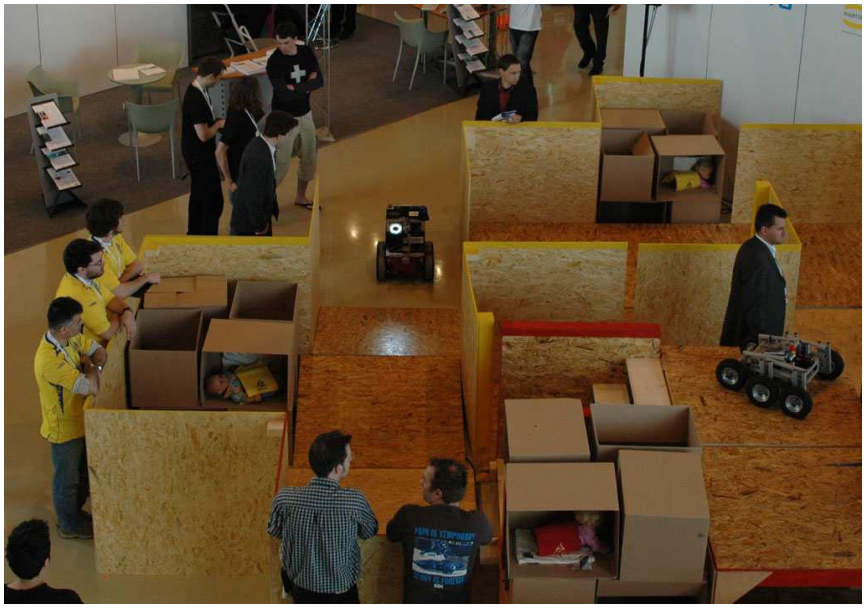

Bild 2.5: Nachgebildetes Verschüttetengebiet beim RoboCup 2007 im Wettbewerb “Rescue League”.

Kapitel 3

Fusionierung von 3D-Laserdaten und Kamerabildern

Die Idee ist, zwei bereits bestehende Ansätze, die Visuelle Odometrie und das 3D-Laserscannen, so zu kombinieren, dass eine verbesserte Positionsbestimmung als in bereits bekannten Verfahren (siehe Abschnitt 2) möglich ist. Ob diese Zusammenführung einen Sinn macht und zu einer Optimierung der Genauigkeit führt, bleibt zu zeigen. Da unterschiedliche Komponenten und Koordinatensysteme verwendet werden, ist die möglichst genaue Kalibrierung des Systems und der Komponenten aufeinander zwingend notwendig. Dies wird in Abschnitt 3.2 genauer behandelt. Anschließend wird auf den konkreten Ablauf eingegangen, angefangen mit der aktiven Vermessung und Verarbeitung der Daten bis hin zur Berechnung der gesuchten Position. Hierbei existiert kein eindeutiger Weg zum Ergebnis, es gibt vielmehr mehrere unterschiedliche Möglichkeiten. Generell zeigt der verfolgte Ansatz starke Parallelen zur Visuellen Odometrie. Dieser stimmt in Bezug auf das Feature-Tracking von Punktkorrespondenzen und Bestimmung der 3D-Weltpunkte mit der visuellen Odometrie überein. Die 3D-Weltpunkte werden jedoch mit Hilfe des 3D-Laserscanners aktiv vermessen, woduch ein Mono-Kamerasystem ausreicht. Der folgende Abschnitt geht auf die technischen Komponenten ein, die dazu notwendig sind.

3.1 Komponenten

Zur Realisierung werden zwei Basiskomponenten benötigt. Ein Kamerasystem um Bilder aufzunehmen, aus denen Merkmale zur weiteren Verarbeitung extrahiert werden können, sowie ein 3D-Laserscanner zur Vermessung ausgewählter Bildmerkmale. In den kommenden Abschnitten werden nur die wichtigsten technischen Daten erwähnt; detailierte Informationen der einzelnen Komponten sind aus der technischen Dokumentation der beigelegten CD-ROM zu entnehmen.

3.1.1 Kamerasystem

Das Kamerasystem besteht aus einer einzelnen perspektivischen Kamera, die möglichst kompakt und günstig sein sollte. Bild 3.1 zeigt zwei Kameras die bei den Experimenten verwendet worden sind. Über die Kamera der Firma “Imaging Source” ([Ima]) sind im Vergleich zur der von “Digital Interface” ([Dig]) sehr viele Informationen verfügbar. Probleme gab es jedoch bei der Ansteuerung und Integration in die Robbie-Software, weshalb die Kamera von Digital Interface letztendlich eingesetzt werden musste. Angeschlossen wird diese dabei über einen Firewire-Anschluss über den, die Bilddaten übertragen werden und gleichzeitig die Stromversorgt erfolgt. Beide Kameras verfügen außerdem über einen Triggerausgang, um synchrone Aufnahmen mehrerer Kameras auszulösen, was beispielsweise bei Stereo-Kamerasystemen notwendig ist.

Es können 30 Bilder pro Sekunde mit einer maximalen Auflösung von $640x480$ Bildpunkten aufgenommen werden, was für die anschließende Verarbeitung ausreichend ist. Da nur Graubilder benötigt werden, ist eine Farbbildkamera nicht notwendig - vielmehr ist die Aufnahmequalität entscheidend. Ein möglichst geringes Bildrauschen und eine Optik, die mit einem möglichst großen Helligkeitsspektrum arbeiten kann, sind hier ausschlaggebende Anforderungen.

Bild 3.1: Links: Digital Interface DFW-VL500. Rechts: Imaging Source DFK 31BF03.

3.1.2 3D-Laserscanner

Der in dieser Arbeit genutzte Laserscanner wird zur Vermessung von Positionen im Raum benötigt. Es gibt schon seit einigen Jahren kommerzielle 2D- und 3D-Laserscanner zu kaufen. [SLNH01] und die Internetseite der Firma thinglab [Thi] geben einen Überblick über unterschiedliche Laserscannervarianten, die aber zum Einen sehr viel teurer und zum Anderen hinsichtlich Größe und Gewicht für mobile Roboter ungeeignet sind. Um dieser Problematik zu begegnen, haben sich verschiedene Wissenschaftseinrichtungen wie beispielsweise das Fraunhofer-Institut für Autonome Intelligente Systeme (AIS) mit der Entwicklung und dem Vetrieb von kompakten 3D-Laserscannern (siehe Bild 3.2) beschäftigt. Die 3D-Laserscanner bestehen dabei aus möglichst kompakten 2D-Laserscannern mit ergänzter Rotationsvorrichtung für die dritte Dimension; diese kosten aber immer noch mehrere tausend Euro.

Um Kosten zu sparen und den zur Verfügung stehenden 2D-Laserscanner - HOKUYO URG-04LX - weiter zu verwenden entstand die Idee [PDMP06], einen “low-cost” 3D-Laserscanner selbst zu entwickeln, insbesondere für zukünftige Wettbewerbe wie den RoboCup. In [Del07] konnte dies bereits in einer Studienarbeit umgesetzt werden und die Ergebnisse zeigen in Bild 3.3, dass ein 3D-Scannen der Umgebung mit leichten Unge-

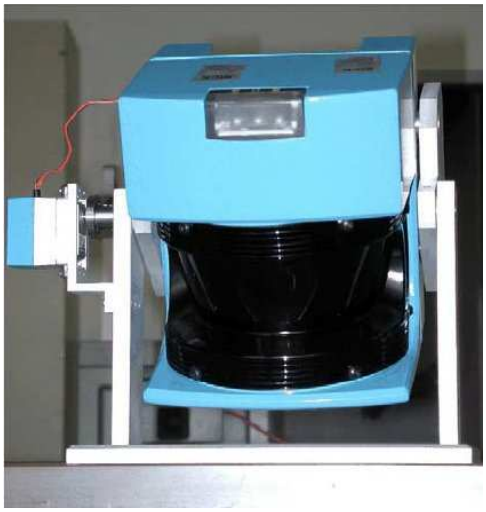

Bild 3.2: AIS 3D-Laserscanner. Quelle: [SNH03].

nauigkeiten möglich ist. Das Vermessen von einzelnen Punkten im Raum ist mit diesem Aufbau jedoch nicht möglich; dennoch können einige Raumpunkte übernommen werden.

Im Folgenden wird nun genauer auf den Aufbau des selbstgebauten 3D-Laserscanners, zur Vermessung von einzelnen Raumpositionen, eingegangen:

In Bild 3.4 ist der angepasste Aufbau zu sehen, der aus einem HOKUYO URG-04LX Laserscanner und HiTEC Digital ROBOT SERVO HSR-8498HB besteht. Die Plattform, auf die der Laserscanner montiert wird, im Bild 3.4 rechts, ist vertikal rotierbar. Eine Aussage über Genauigkeit und Vermessungszeiten wird in Kapitel 5 anhand von Experimenten getroffen.

Der Servomotor kann nicht direkt mit dem Computer verbunden werden, ein Schnittstellenbaustein ist dazu erforderlich. Hierfür wird das Crumb8-USB Modul mit Atmega 8 Mikrocontroller mit der Möglichkeit, Daten über eine USB-Schnittstelle auszutauschen, eingesetzt; dieses Modul wurde bereits in der erwähnten Studienarbeit [Del07] verwendet und konnte durch seine unkomplizierte Programmierung mit Hilfe des CrispAVR-USB und weiteren positiven Eigenschaften wie kompakter Bauweise überzeugen.

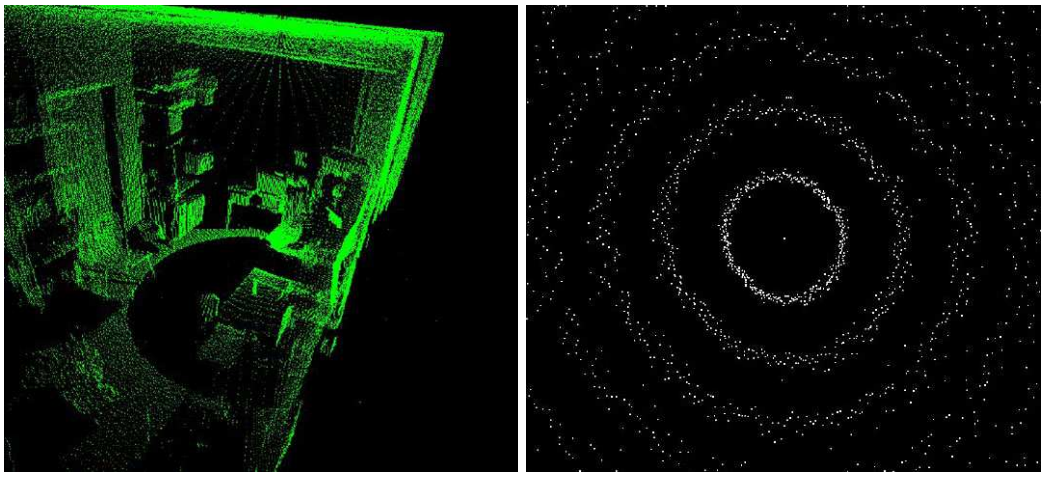

Bild 3.3: Links: Serverraum mit 173.910 Messpunkten. Rechts: Radiale Anordnung der Messpunkte. Quelle: [Del07].

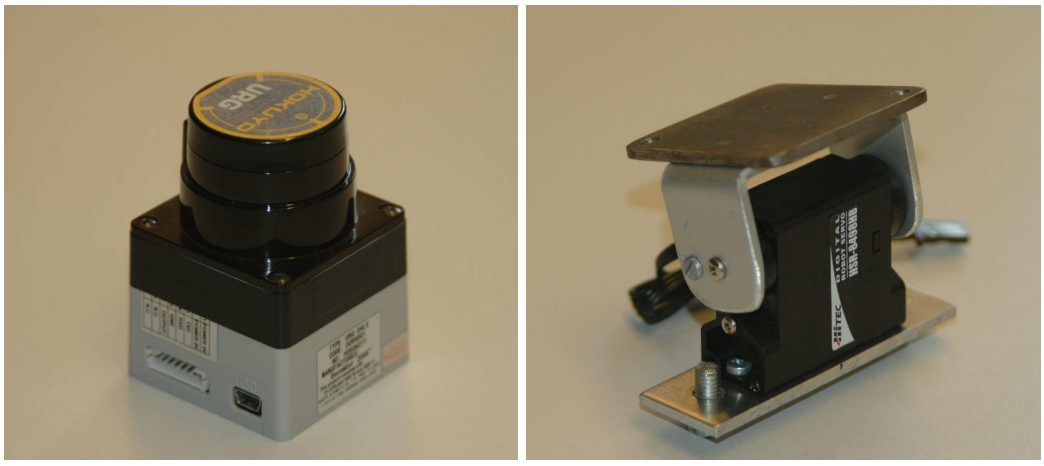

Bild 3.4: Links: HOKUYO URG-04LX Laserscanner. Rechts: Konstruierte vertikale Rotationsplattform.

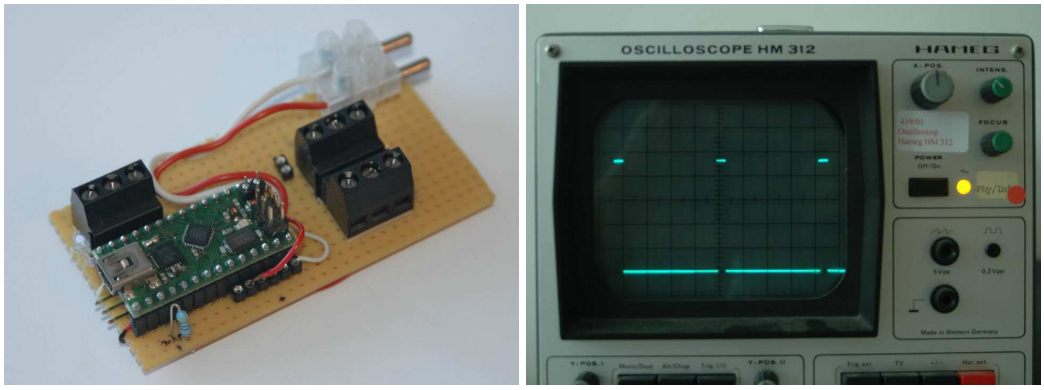

Bild 3.5: Links: Gesockeltes Crumb8-USB Modul. Rechts: Generiertes PWM-Signal vom Mikrocontroller dargestellt am Oszilloskop.

Mikrocontroller Das Crumb8-USB Modul mit Atmega 8 Mikrocontroller stellt neben einer Reihe von PIN Ein- und Ausgängen die einfache Erzeugung eines hardwarebasierten Pulsweitenmodulationssignals (PWM-Signal) bereit. Um diesen Baustein auch für zukünftige Projekte einzusetzen, ist der zu flashende Quellcode in der Programmiersprache C geschrieben. Abbildung 3.5 zeigt die entwickelte Platine mit gesockeltem Crumb8-USB Modul und das generierte PWM-Signal, welches mit Hilfe eines Oszilloskops[1] gemessen und dargestellt wird. Die Frequenz des Signals liegt standardmäßig bei $50Hz$, was der Periodendauer von $20ms$ entspricht. Heutige Servomotoren lassen sich bis zu einer Frequenz von $100Hz$ ansteuern. Der Vorteil liegt dabei bei der schnelleren Umsetzung der geänderten Pulsweite, die bei $100Hz$ nach $10ms$ geändert werden kann und bei $50Hz$ nur alle $20ms$. Die Pulsweite bei Servomotoren liegt im Bereich von ein bis zwei Millisekunden. Durch Änderung dieses Wertes wird der Servomotor bis zum linken Anschlag bei ca. $1ms$ rotiert und bis zum rechten Anschlag bei ca. $2ms$. Je nach gesetzten Parametern im Mikrocontroller steht ein unterschiedlich großer Wertebereich zur Verfügung, um den Servomotor zwischen beiden Endpositionen zu bewegen; dieser bestimmt die Auflösung der ansteuerbaren Positionen. Zur Berechnung dieser Position siehe Anhang A.

[1] Elektronisches Messgerät zur Darstellung des zeitlichen Verlaufs einer Gleich- oder Wechselspannung.

3.2 Kalibrierung

Die Kalibrierung ist eines der umfangreichsten und entscheidensten Verfahren bei der Berechnung und Verarbeitung von Messdaten, insbesondere bei der Verwendung unterschiedlicher Bezugsgrößen und Modelle. Durch eine ungenaue und nicht optimale Abstimmung zwischen Komponenten und diesen selber sind unbestimmte, fehlerhafte Ergebnisse die Folge. Daher ist es notwendig, ein Referenzmodell festzulegen, in das alle anderen verwendeten Komponentenmodelle transformiert werden. Zur Beschreibung der Position im realen Raum wird das rechtshändige Weltkoordinatensystem als Referenzmodell festgelegt. Die nächsten Abschnitte gehen dabei auf die einzelnen Komponentenmodelle und deren Kalibrierung ein, um eine Möglichkeit zu haben, alle Daten miteinander in Beziehung zu setzen und darauf Berechnungen durchzuführen.

3.2.1 Kamera

Das Lochkameramodell ist das übliche Modell zur Beschreibung von perspektivischen Kameras, das mit Hilfe von bestimmten Parametern beschrieben wird; diese lassen sich in extrinsische und intrinsische Paramter aufteilen. Die extrinsischen Parameter definieren die Position und Orientierung der Kamera in Bezug zum übergeordneten Weltkoordinatensystem. Die geometrischen Kameraeigenschaften werden mit den intrinsischen Parametern aus Tabelle 3.2.1 beschrieben. Da die intrinsischen Parameter unabhängig von der Kamerabewegung und -orientierung sind, ist für sie nur eine einmalige Bestimmung notwendig. Nach [TV98] und [Dan07] lassen sich somit folgende Gleichungen (3.1) zur Abbildung eines 3D-Weltpunktes auf ein 2D-Pixel, was das Ziel der Kalibrierung selbst ist, im Bild aufstellen.

- $\boldsymbol{R}$: Orthogonale $3x3$ Rotationsmatrix ($\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$).
- $\boldsymbol{t}$: 3D Translationsvektor, der die Verschiebung zum Ursprung zwischen beiden Koordinatensystemen beschreibt.
- WCS, CCS, ICS und PCS: 3D Weltkoordinatensystem, 3D-Kamerakoordinatensystem, 2D-Bildkoordinatensystem und 2D-Pixelkoordinatensystem.

Intrinsische Kameraparameter	
F	Brennweite - Abstand zwischen dem optischem Zentrum und der Bildebene.
α	Seitenverhältnis zwischen der vertikalen u. horizontalen Ausdehnung eines Pixels.
c_x, c_y	Bildhauptpunkt des Schnittpunktes der Bildebene mit der optischen Achse.
β	Winkel zwischen x- und y-Achse der Bildebene bei windschiefem Pixel.
κ_1, κ_2	Radiale Verzerrungskoeffizienten

Tabelle 3.1: Intrinsische Kameraparameter.

$$\begin{aligned} WCS \;\Rightarrow\; CCS: \quad \boldsymbol{p}^c &= \boldsymbol{R}^T(\boldsymbol{p}^w - \boldsymbol{t}) \\ CCS \;\Rightarrow\; ICS: \quad \boldsymbol{p}^i &= \begin{pmatrix} p_x^i \\ p_y^i \end{pmatrix} = \begin{pmatrix} F\frac{p_x^c}{p_z^c} \\ F\frac{p_y^c}{p_z^c} \end{pmatrix} \\ ICS \;\Rightarrow\; PCS: \quad \boldsymbol{p}^p &= \begin{pmatrix} \frac{x^i}{dx} + c_x \\ \frac{y^i}{dy} + c_y \end{pmatrix} \end{aligned} \tag{3.1}$$

Um direkt von Pixelkoordinaten in Weltkoordinaten umrechnen zu können, ist die Aufstellung einer Matrixmultiplikation (3.2) notwendig und eine Überführung der euklidischen 2D-Bildkoordinaten in das entsprechende homogene Koordinatensystem.

$$\begin{aligned} \hat{\boldsymbol{p}}^i &= \begin{pmatrix} \hat{p}_x \\ \hat{p}_y \\ \hat{p}_z \end{pmatrix} = \boldsymbol{K}\boldsymbol{P}_{pers}\boldsymbol{D}\begin{pmatrix} p_x^w \\ p_y^w \\ p_z^w \end{pmatrix} \\ &= \begin{pmatrix} F & \beta & c_x \\ 0 & \alpha F & c_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} & \boldsymbol{R} & & \boldsymbol{t} \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} p_x^w \\ p_y^w \\ p_z^w \\ 1 \end{pmatrix} \\ \boldsymbol{p}^i &= \begin{pmatrix} \frac{\hat{p}_x}{\hat{p}_z} \\ \frac{\hat{p}_y}{\hat{p}_z} \end{pmatrix} \end{aligned} \tag{3.2}$$

Das bisher verwendete Lochkameramodell geht von idealisierten Kamerabildern aus, die keinerlei Verzerrungen aufweisen. Praktische Untersuchungen haben gezeigt, dass bei

CCD-Kameralinsen meist zwei verschiedene Arten von Linsenverzerrungen auftreten können. Zum Einen eine radiale Verzerrung und zum Anderen eine tangentiale, die normalerweise vernachlässigt werden kann. Aus den idealisierten Koordinaten (p_x^i,p_y^i) lassen sich die realen (unverzerrten) Bildkoordinaten ($\tilde{p}_x^i$,$\tilde{p}_y^i$) aus den Gleichungen (3.3) berechnen.

$$\begin{aligned} \tilde{p}_x^i &= p_x^i + (p_x^i - c_x)(\kappa_1 r^2 + \kappa_2 r^4) \\ \tilde{p}_y^i &= p_y^i + (p_y^i - c_y)(\kappa_1 r^2 + \kappa_2 r^4) \\ r &= \sqrt{(p_x^i - c_x)^2 + (p_y^i - c_y)^2} \end{aligned} \tag{3.3}$$

Kalibrierung nach Tsai Bei der Kalibrierung nach Tsai [Tsa87] werden die im Kamerabild detektierten Punkte eindeutig den bekannten Weltkoordinaten der entsprechenden Punkte des Kalibriermusters zugeordnet. Dabei werden optimale Kameraparameter gefunden, welche die Eigenschaft haben, 3D-Weltpunkte auf 2D-Bildpunkte mit minimalem Fehler abzubilden. Bei der Rückprojektion von 2D nach 3D bleibt jedoch die Tiefe des 3D-Weltpunktes unbekannt; diese kann aber beispielsweise mit Hilfe eines Laserscanners ermittelt werden.

3.2.2 3D-Laserscanner

Der 3D-Laserscanner ist wie in 3.7 zu sehen auf einer Rotationsplattform montiert. Für die Berechnung der korrekten Entfernung zwischen dem Ursprung der Rotationsachse und des 3D-Weltpunktes ist die Bestimmung des Abstandes zwischen Rotationachsen- und Laserscannerursprung notwendig. Die Koordinate des 3D-Weltpunktes im Koordinatensystem des Laserscanners kann zuvor mit Gleichung (3.5) berechnet werden.

Die Transformation von einem beliebigen Punkt vom Laserscanner-Koordinatensystem mit Ursprung des Laserstrahls $P_{laser}^{L'}$ zu einem Punkt mit Ursprung der Rotationsachse $P_{rotation}^{L}$ kann folgendermaßen berechnet werden:

Als Ausgangspunkt wird die mathematische Gleichung (3.4) zur Berechnung eines Punk-

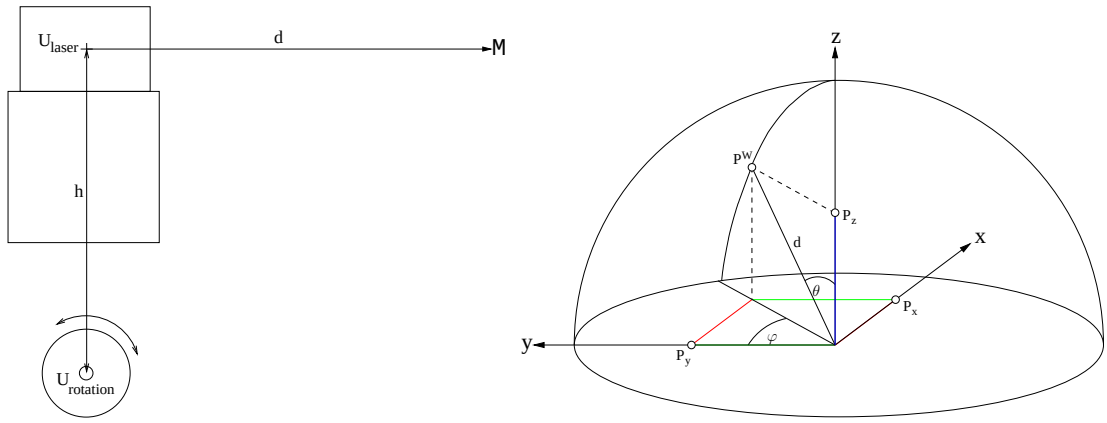

Bild 3.6: Links: Abstand h zwischen Rotationsachsenursprung $U_{rotation}$ und Laserstrahlursprung U_{laser}. Rechts: 3D-Punkt auf einer Kugeloberfläche.

tes auf einer Kugeloberfläche (Bild 3.6) herangezogen, mit d als Abstand zwischen Ursprung des Laserscanners und einem Objektpunkt M.

$$\begin{aligned} p_x^{L'} &= d * sin(\theta) * sin(\varphi) \\ p_y^{L'} &= d * sin(\theta) * cos(\varphi) \\ p_z^{L'} &= d * cos(\varphi) \end{aligned} \tag{3.4}$$

Da die Rotationsachse $U_{rotation}$ nicht im Ursprung des Laserstrahls U_{laser} liegt, ist es notwendig, Gleichung (3.4) um die Entfernung h in Abhängigkeit zum Rotationswinkel θ zu erweitern.

$$\begin{aligned} P_x^L &= P_x^{L'} + cos(\theta)h = P_x^{L'} + \Delta x \\ P_y^L &= P_y^{L'} \\ P_z^L &= P_z^{L'} + sin(\theta)h = P_z^{L'} + \Delta z \end{aligned} \tag{3.5}$$

$$
\begin{aligned}
sin(\alpha) &= \Delta x/h \\
\alpha &= 180 - 90 - \theta
\end{aligned}
$$

$$
\begin{aligned}
\Delta x &= sin(90 - \theta) * h = cos(\theta) * h \\
\Delta z &= sin(\theta) * h
\end{aligned}
$$

Der Abstand zwischen Laserscanner- und Rotationsachsenursprung kann mittels zweier unterschiedlicher trigonometrischer Verfahren bestimmt werden. Zum Einen bei bekannter Entfernung des horizontalen Laserstrahls d_0 (mit $\alpha = 0$) und eines weiteren Laserstrahls d_1 (mit $0 < \alpha < 90$) oder zum Anderen mittels zweier Laserstrahlen l_1 (mit $0 < \alpha < 90$) und l_2 (mit $0 < \beta < 90$), siehe Zeichnung (Bild A.1). Die Gleichung (3.6) zeigt die Berechnung des Abstandes h mit Hilfe des zweiten Verfahrens, der auch bei den Experimenten in Kapitel 5 verwendet wird. Für weiter Details siehe Anhang A.

$$h = s * sin(\delta)/sin(\alpha + \beta) \tag{3.6}$$

3.2.3 Gesamtsystem

Das Gesamtsystem ist, wie Bild 3.7 zeigt, mit den zuvor beschriebenen Kalibrierungen und Modellen theoretisch in der Lage, Punkte im Raum zu vermessen und diese auf Bildpunkte abzubilden. Legt man das Kamerakoordinatensystem als das Weltkoordinatensystem fest, so kann das Laserscannerkoordinatensystem mit Hilfe einer Rotationsmatrix R und einer Translation t ins Weltkoordinatensystem, beziehungsweise in diesem Fall auch ins Kamerakoordinatensystem konvertiert werden. Dies wird durch die Orientierung des Laserscanners, der die gleiche Ausrichtung hat wie die Kamera, sichergestellt. Entsprechend dem praktischen Aufbau kann folgende Transformation aufgestellt werden:

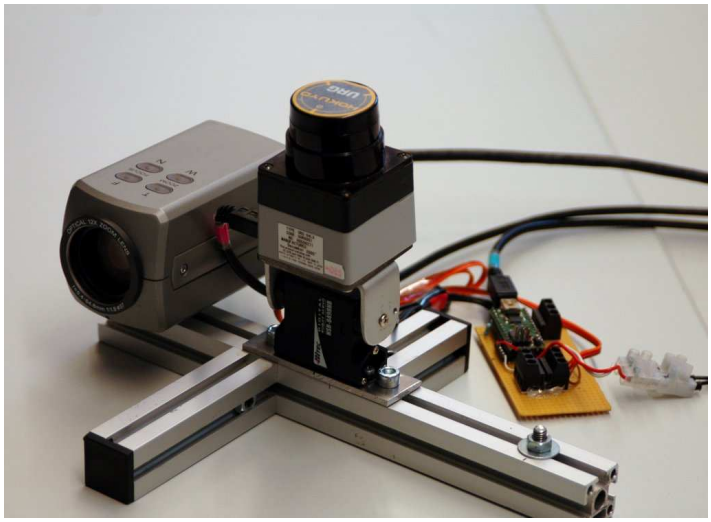

Bild 3.7: Flexibler Gesamtaufbau zur Vermessung von 3D-Weltpunkten, zur einfachen Montage auf den Roboter ”Robbie-8“.

$$\boldsymbol{P^w} = \boldsymbol{R}(\boldsymbol{P}^{L'} - \boldsymbol{t})$$

$$\boldsymbol{P^w} = \begin{pmatrix} x^W \\ y^W \\ z^W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(\frac{1}{2}\pi - \theta)cos(\varphi) & cos(\theta) & t_x \\ sin(\theta)cos(\varphi) & 0 & t_y \\ cos(\varphi) & sin(\theta) & t_z \end{pmatrix} \begin{pmatrix} d \\ h \\ 1 \end{pmatrix}$$

3.3 Bildbasiertes aktives 3D-Scannen

Die Visuelle Odometrie mit einem Stereo-Kamerasystem nutzt zur Berechnung der Positionsänderung die Triangulation von 3D-Weltpunkten ausgewählter Features aus den getrackten Stereobildern. Bei einem Mono-Kamerasystem ist die Berechnung von 3D-Weltpunkten der Features nicht erforderlich, um die Rotation und Translation zu erhalten. Diese lässt sich durch Berechnung von Fundamentalmatrix und Essentialmatrix und anschließender Faktorisierung berechnen. Beim bildbasierten aktiven 3D-Scannen hingegen werden ebenfalls Features aus Bildern extrahiert, wobei ein Mono-Kamerasystem hierbei wiederum ausreicht. Unterschiede liegen zum Einen in der Auswahl und Anzahl der

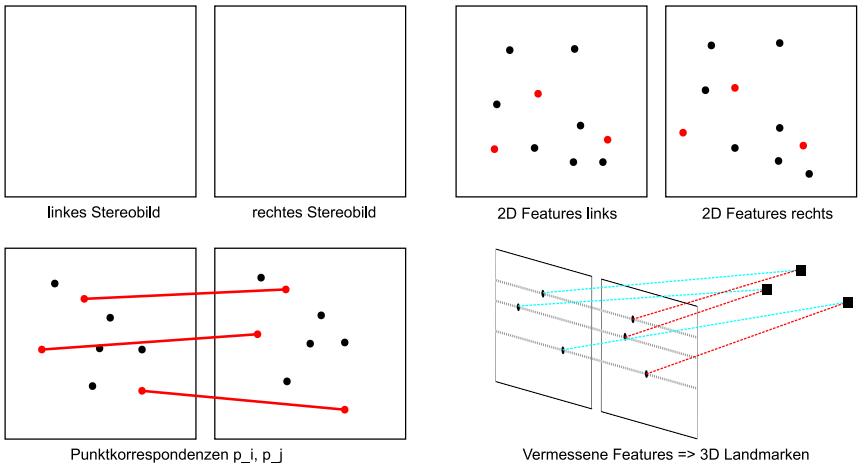

Bild 3.8: Kernpunkte der visuellen Odometrie und bildbasierten 3D-Laserscannens.

notwendigen Features was in Abschnitt 3.3.2 näher beschrieben wird, sowie in der Vermessung der 3D-Punkte, siehe Abschnitt 3.3.3. Bild 3.8 zeigt eine Veranschaulichung der Kernpunkte beider Verfahren. Im Folgenden werden mögliche Gesamtabläufe des fusionierten Verfahrens erläutert.

3.3.1 Ablauf

Der Gesamtablauf des bildbasierten aktiven 3D-Scannens lässt sich wie folgt realisieren: Der Ablauf des Verfahrens ist in zwei Phasen, eine Lern- und Navigationsphase, unterteilt. In der ersten Phase wird ein Bild aufgenommen und aus diesem eine Menge von Features extrahiert, welche bestimmten Kriterien unterliegen. Im nächsten Schritt werden ausgewählte Features mit Hilfe des 3D-Laserscanners vermessen, die von hier an als Landmarken bezeichnet werden sollen. Abschließend werden die gewonnenen Informationen in einer Wissensdatenbank gespeichert, um zu einem späteren Zeitpunkt darauf zurückgreifen zu können. Die Navigationsphase beginnt zunächst ebenfalls mit der Aufnahme eines Bildes und der Extraktion von Features. In einem weiteren Schritt wird versucht, die zuvor ausgewälten Features der Lernphase zu tracken, um damit Punktkorrespondenzen zwischen beiden Aufnahmen zu finden. Als Nächstes werden die gespeicherten Landmarken, die 3D-Positionen der Features in der Welt, aus der Wissensdatenbank geladen. In

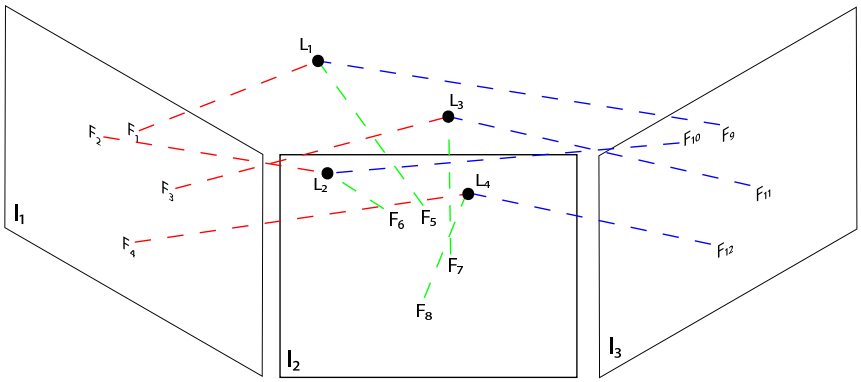

Bild 3.9: Gewonnene Informationen durch mehrmalige Durchläufe des ersten Ansatzes. I_n: Bild n einer Bildersequenz, F_n: Feature n und L_n: Landmarke n (3D-Weltkoordinate).

einem letzten Schritt wird versucht, eine Möglichkeit zu finden, um aus den gewonnen Informationen eine eindeutige Position zu schätzen. Der wesentliche Kern dieses Ansatzes ist die einmalige Vermessung der Features in der Lernphase, die nur dann wieder durchgeführt wird, wenn eine ausreichende Anzahl von Features nicht mehr trackbar ist, da diese sich nicht mehr im aktuellen Bildausschnitt befinden. Bild 3.9 veranschaulicht, welche Informationen durch mehrmalige Durchführung des gerade beschriebenen Ablaufes gewonnen werden.

In einer Bildsequenz von drei Bildern werden jeweils vier Features getrackt, wobei aber nur die 3D-Weltposition der Features aus dem ersten Bild bekannt sind. Damit erhält man eine Zuordnung von 2D-Bildpunkten auf 3D-Weltpunkte im ersten Bild und in allen folgenden Bildern eine Zuordnung zwischen den 2D-Bildpunkten der Features durch das Tracking. Eine mathematische Lösung, um daraus eine eindeutige Position berechnen zu könnnen, konnte bislang nicht gefunden werden, was zu einer leichten Abänderung des ersten Ansatzes führt. In der Navigationsphase ist es nun ebenfalls notwendig, die getrackten Features zu vermessen und die Position mit Hilfe der damit gewonnenen 3D-Punktemengen zu berechnen. Das dies möglich ist zeigen bereits Arbeiten im Bereich der visuellen Odometrie mittels Stereo-Kamerasystemen [ZON+06]. Der Nachteil des veränderten Ansatzes liegt in der deutlich höheren Bearbeitungszeit, die durch die Vermessung der getrackten Features aller Bilder zustande kommt. Zu den Details der Vermessungszeiten siehe Kapitel 5.

3.3.2 Bestimmung und Tracking von Features

Es gibt mehrere Algorithmen zur Extraktion von Features aus Bildern, wie in Abschnitt 2.3.2 bereits erwähnt. In dieser Arbeit wurde der SIFT-Extraktor [SLL01] als Beispieloperator ausgewählt, da er zwar rechenintensiv ist, aber eine Reihe von positiven und sehr wichtigen Eigenschaften besitzt. In erster Linie erzeugt dieser Operator eine sehr zuverlässige Feature-Beschreibung[2], die besonders bei einer geringen Anzahl von benötigten stabilen Features Sinn macht. Er ist außerdem Rotations- und Skalierungsinvariant und relativ unempfindlich gegenüber Beleuchtung und Kamerastandpunkt. Da in jedem aufgenommenen Kamerabild ein gewisses Grundrauschen vorhanden ist, sollte vor jeder Extrahierung eine Bildglättung mit Hilfe eines Medianfilters durchgeführt werden. Dies verhindert die Extraktion von Features, die in den kommenden Bildern, die ein anderes Rauschen besitzen, erst gar nicht vorkommen. J.C. Rosenthal [Ros06] beschäftigt sich ausführlich mit den Themen Feature-Extraktion und -Tracking. Bei der Verwendung des SIFT-Operators und Verarbeitung einer Mono-Bildsequenz läuft der Vorgang ähnlich ab, wie er es für den Fall einer Stereo-Bildsequenz beschreibt. Es werden zwei Listen von extrahierten Features erstellt und Punktkorrespondenzen gesucht. Das Maß für die Ähnlichkeit zweier Features, aus dem ersten und zweiten Bild, wird mit Hilfe des euklidischen Abstandes in Gleichung A.1 bestimmt; dieser lässt sich aus den Deskriptordaten der Features berechnen. Das Punktpaar mit dem kleinsten Abstand der beiden Listen ist damit die gesuchte Punktkorrespondenz des ersten Features. Dieser Vorgang wird für alle Features berechnet, sodass als Ergebnis eine optimale Zuordnung aller Features aus dem jeweiligen Bild gefunden wird, siehe Bild 3.10. Durch Setzen des Fehlers, der maximalen Distanz zwischen zwei Punktpaaren lässt sich die Anzahl und Genauigkeit der getrackten Features festlegen. Werden nach einiger Zeit und mehrerer Bildaufnahmen unterschiedlichen Standorts zu wenige Korrespondenzen gefunden, so wird eine Vermessung neu ausgewählter, stabiler Features notwendig. Bei fehlerhaften Punktzuordnungen werden die Daten unbrauchbar und die darauf aufbauenden Ergebnisse verfälscht. Die Schwierigkeit dabei ist, dies möglichst früh zu erkennen um daraufhin andere Punktkorrespondenzen zu verwenden.

[2] Feature-Deskriptor

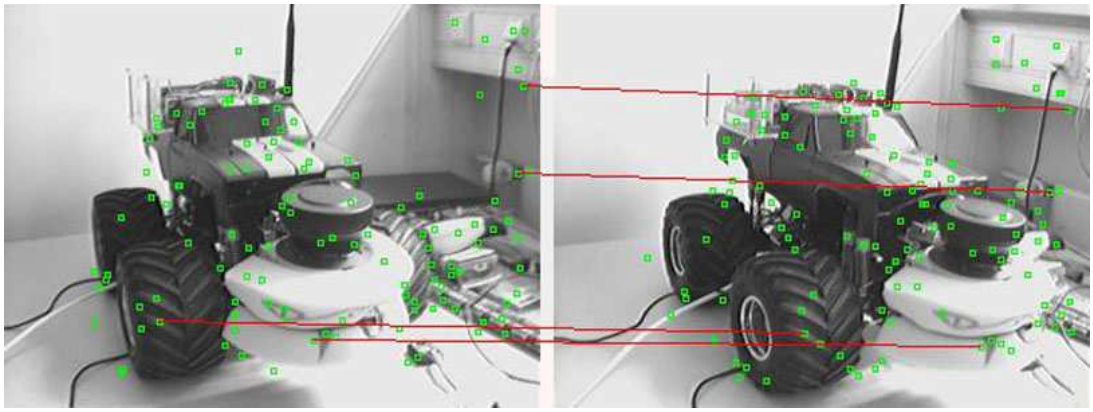

Bild 3.10: Feature Tracking: 4 Punktkorrespondenzen, Fehlerschwellwert 140 (euklidischer Maximalabstand). Quelle: EnhancedReality T.Feldmann.

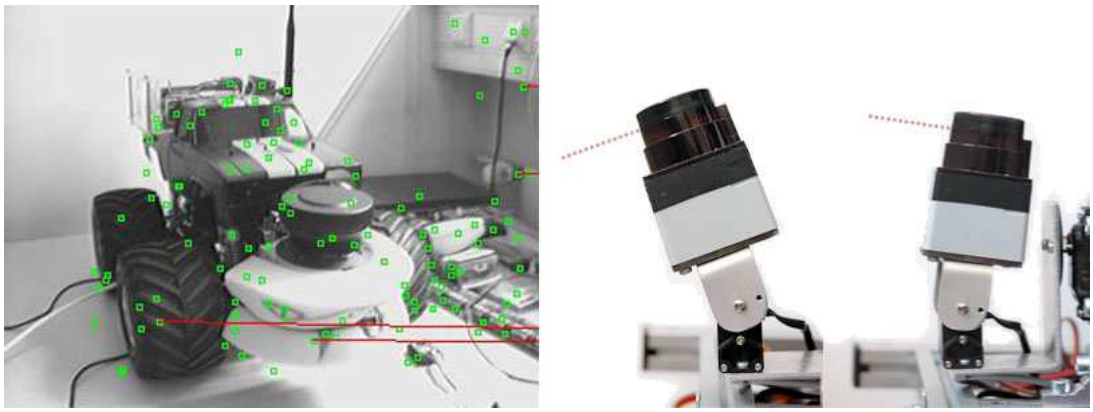

Bild 3.11: Feature-Vermessung mittels 3D-Laserscanner.

3.3.3 Bestimmung der 3D-Position von Features

3D-Positionen von Features werden bei der visuellen Odomotrie mit Stereo-Kamersystemen mittels Triangulation berechnet; dies ist hier durch die aktive Vermessung mittels entwickeltem 3D-Laserscanner ersetzt, siehe Bild 3.11. Von den ausgewählten Features sind die beiden Winkel zur horizontalen Achse ϕ und zur vertikalen Achse θ zu berechnen (siehe Abschnitt 3.2.1), bevor eine Ansteuerung möglich ist. Dabei ist die Genauigkeit der Winkelberechnung ausschlaggebend für die Korrektheit der 3D-Positionen und der daraus geschätzten relativen Position. Um ein Feature zu vermessen wird der Servomotor auf den berechneten Winkel θ positioniert und ein 2D-Laserscan aufgenommen. Der zweite Winkel ϕ gibt nun an, welcher Indexposition des Scans die entsprechende Entfernung steht. Idealerweise kann aus der Entfernung, den beiden Winkeln und der Projektionsmatrix, die während der Kalibrierung bestimmt worden ist, die 3D-Position des Features berechnet werden.

Kapitel 4

Integration in Robbie-8

Die Softwareentwicklung hat mehrere Entwicklungsstufen; diese teilen sich grob in Entwurf, Realisierung und Testphase auf. Bei größeren Projekten ist es sinnvoll, mehrere eigenständige Programmeinheiten zu entwickeln, die an festgelegte Rahmenbedingungen wie zum Beispiel Systemarchitektur, Schnittstellen, Design-Patterns oder Programmiersprachen gebunden sind. Dabei werden diese unabhängig voneinander realisiert und getestet. In der Endphase werden sie in ein Gesamtsystem eingebunden. Diese Art der Entwicklung hat sowohl Vor- als auch Nachteile. Ein Nachteil kann etwa die notwendige Zeit zur Integration ins Gesamtsystem sein. Weiter können bei den Schnittstellenspezifikationen Probleme auftreten oder aber allgemeine Inkompatibilitäten. Der Vorteil bei der Einteilung in Programmeinheiten besteht jedoch beispielsweise in der sichergestellten Funktionalität der einzelnen Teile und deren Wiederverwendbarkeit; dies macht sie flexibel einsetzbar.

Der nun folgende Abschnitt 4.1 geht auf das in der Anfangsphase erstellte, eigenständige Toolkit ein; dies soll hauptsächlich den Einstieg erleichtern und sollte nicht zwangsweise an die Robbiearchitektur, die in Abbschnitt 4.2 beschrieben ist, gebunden sein. Dennoch orientiert sich die Implementation an deren Grundrahmen, um die Integrationszeit zu verkürzen. Abschnitt 4.3 und 4.4 geben eine Übersicht über die umgesetzte Implementierung in Robbie-8 sowie die bereits vorhandenen und erweiterten Komponenten, die für diese Arbeit relevant sind.

4.1 Toolkit

Am Anfang der Arbeit war noch unklar, wie genau die einzelnen Hardwarekomponenten angesprochen und kombiniert werden sollen; daher war es nötig, ein eigenständiges Toolkit zu erstellen. Da abzusehen war, dass eine möglichst baldige Implementierung im Robbie-8 Framework fortgesetzt werden würde, sollte es nicht zu komplex werden und beschränkt sich daher auf die Grundfunktionen zur Ansteuerung von Kamera, Laserscanner und Servomotor. Der größte Vorteil, den das Toolkit hat, ist die Unabhängigkeit von Nachrichten (Messages) und Modulen, die im Robbieframework zur Datenübertragung und zur eigentlichen Datenverarbeitung genutzt werden. Auf diesen Punkt wird in Abschnitt 4.3 und 4.4 genauer eingegangen. Um die Hardwareansteuerung zu testen, sind Unit-Tests in den Quelldateien implementiert. Listing 4.1 zeigt ein Beispiel für einen solchen Unit-Test zur Ansteuerung des Servomotors aus der Mikrocontrollerklasse, die sich somit unabhängig von anderen Klassen ausführen und testen lässt. Zur Verarbeitung von SIFT-Features wird die Implementierung von David G. Lowe in der Klasse „Sift“ verwendet, siehe UML Diagramm (Bild 4.1). Neben der Aufnahme und Verarbeitung der Daten ist es hierzu notwendig, eine Möglichkeit zur Speicherung zu bieten, um Berechnungen auch im “offline” Modus durchführen zu können. Am Beispiel der Laserscannerdaten sind diese in der Datenstruktur STLmap gespeichert. Diese Daten können als binäre Dateien gespeichert und geladen werden, siehe Quellcodeauszug in Listing 4.2. Außerdem können die Daten einer STLmap als Gnuplot- oder Octave-Textdateien zu Visualisierungszwecken ebenfalls gespeichert werden.

Listing 4.1: Unit-Test zur Ansteuerung des Servomotors.

```
/* UnitTest uController */
int main( int argc, char** argv ){
  // Legt ein neues Objekt vom Typ uController
  uController* uC = new uController();
  if ( uC->init() ){  // Initialisiert den Mikrokontroller
    // Ändert die Pulsweite und damit die Position des
       Servomotors
    uC->setHighPWMdegree( 20 );  // um 20 Grad
    my_sleep( 1000 ); // Wartet eine Sekunde
    uC->setHighPWMdegree( -20 ); // um -20 Grad
  }
  return 0;
}
```

Listing 4.2: STLmap Datenstruktur zur Verwaltung von Laserscannerdaten.

```
map<KEY, VALUE >
//   KEY = int (Winkel des Servomotors)
// VALUE = vector<T > > (n-te Messung)
//     T = vector<int > (682 Werte eines Laserscans)
map<int, vector<vector<int > > > m_scanMap;
```

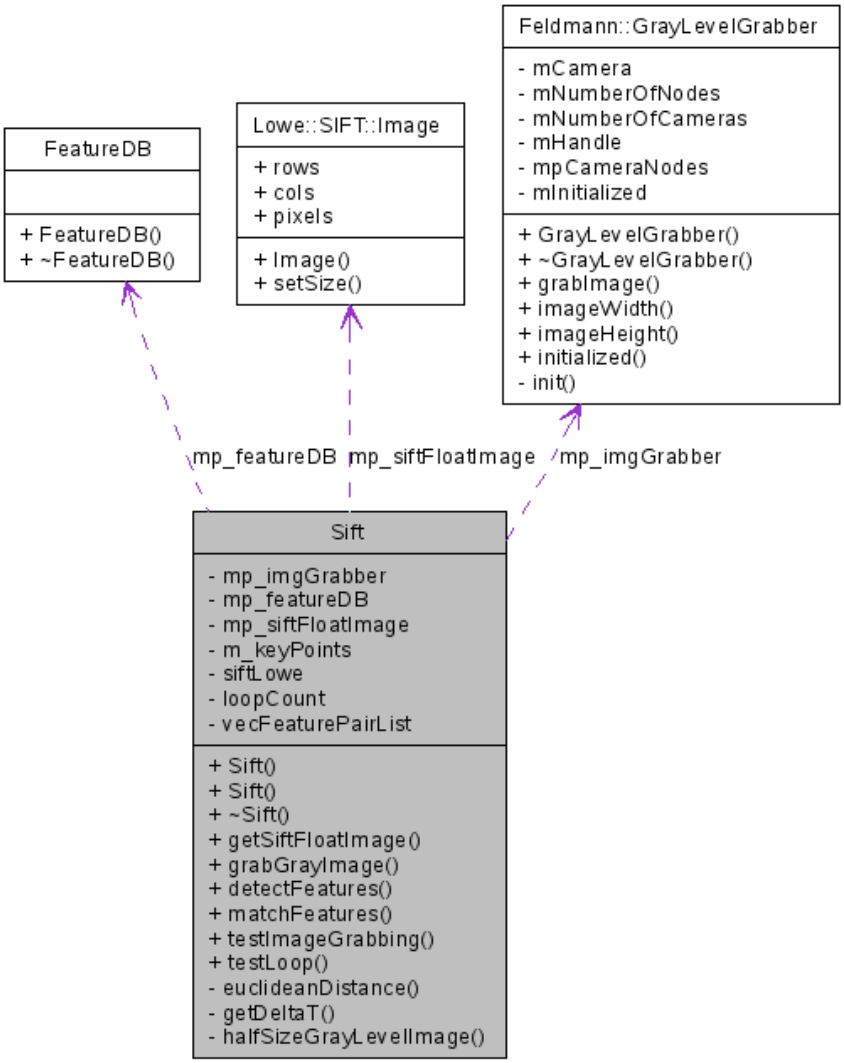

Bild 4.1: UML Klassendiagramm der Klasse Sift.

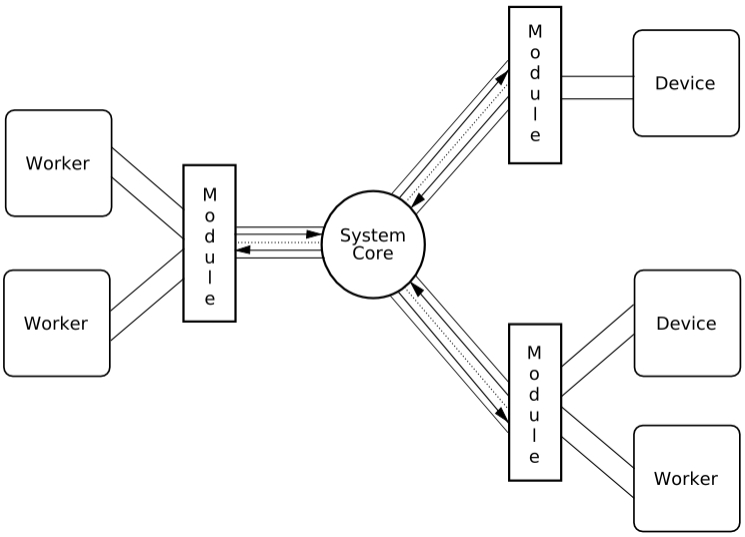

Bild 4.2: Schematisches Modell der “Robbie-8” Systemarchitektur.

4.2 Architektur

Bevor die bereits realisierten Implementierungen in das Robbie-8 Framework integriert werden können, ist es notwendig, sich einen Überblick der Systemarchitektur zu verschaffen. Bild 4.2 zeigt dies anhand eines vereinfachten, schematischen Modells. Der “System Core” ist die zentrale Komponente des Systems; an dieser werden beliebige Module angemeldet und stellen die Verbindung zu “Workern” und “Devices” her. “Worker” verarbeiten dabei Daten des Systems oder der aufgenommenen Daten der “Devices” wie beispielsweise Laserscanner oder Kameras. Da jedes Modul in einem separaten Thread abgearbeitet wird, ensteht somit ein parallel arbeitendes, flexibles System.

Um die Performance der Anwendung zu optimieren, wird eine “Dual-Core-Architektur[1]”, siehe Bild 4.3, verwendet und die gleiche Software auf zwei Rechnern ausgeführt, jedoch mit unterschiedlich angemeldeten Modulen. Damit kann der Server (Robbie) sich um die Aufnahme der Hardwaredaten kümmern und der Client führt intensive Berechnungen aus. Um Daten zwischen Modulen und den zwei Rechnern auszutauschen, werden diese in Nachrichten gekapselt und über eine Zweiwege-Schlange verschickt. Inzwischen können

[1] Client-Server-Architektur

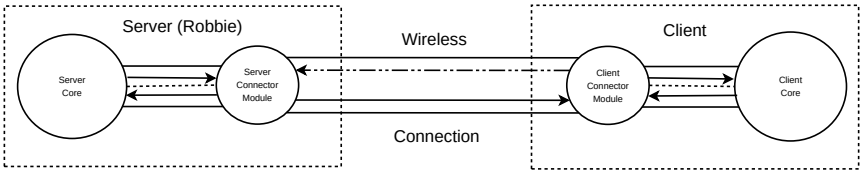

Bild 4.3: Client-Server-Architektur. Quelle: [Tec07].

die Nachrichten auch über eine stabile, drahtlose Verbindung verschickt werden, was die Flexibilität und Mobilität wesentlich erhöt.

4.3 Messages

Um Daten zwischen den Modulen austauschen zu können ist es notwendig, diese beim Senden in Nachrichten zu kapseln und beim Empfangen wieder zu entpacken. Für den Transport zwischen verschiedenen Clients müssen die Daten jedoch zuerst noch serialisiert werden, um Datenverluste bei der Übertragung zu vermeiden. Die erstellten Nachrichten können mit Hilfe eines Loggers gespeichert und geladen werden, um weitere Verarbeitungen mit den Daten offline durchzuführen. In Tabelle 4.1 ist eine Übersicht der wichtigsten Nachrichtentypen, die zur Vermessung von Features notwendig sind, mit der jeweiligen Semantik aufgelistet.

Ein Problem, das bei der Verwendung einer parallelen Architektur in Verbindung mit Nachrichten entstehen kann, ist die der Abarbeitung einer Sequenz von Einzelschritten. Möchte man zum Beispiel zwei Laserscans von zwei verschiedenen Winkeln empfangen, so wird zuerst eine SetServoM-Nachricht erstellt, wartet dann auf den Empfang einer LaserDataM-Nachricht mit passendem Zeitstempel und wiederholt diesen Vorgang. Die Implementierung ist relativ komplex sowie verschachtelt und führt zu Performanceeinschränkungen. Um das Gesamtsystem durch die Nachrichtenarchitektur zu entlasten, haben die Nachrichten nur eine beschränkte Lebensdauer; diese wird festgelegt durch eine Anzahl von Zugriffen, nach deren Erreichung sie gelöscht und der betreffende Speicher wieder freigegeben wird. Die Nachrichtenarchitektur in Verbindung mit der parallelen Verarbeitung der Module ist optimal solange keine zeitkritischen Abläufe auftreten.

ImageM (e)	Enthält Daten eines Farbbildes und dessen Typ (REAR_CAM, LEFT, etc.).
GetImageM (u)	Fordert eine ImageM an.
SetServoM (e)	Enthält Daten einer Servonummer und dessen Winkel.
LaserDataM (u)	Enthält Daten eines Laserscans (682 Werte).
LaserscanThetaListM (n)	Enthält Daten einer Liste von Winkeln und einer STLmap, in der zu jedem Winkel der entsprechende Laserscan gespeichert ist.
GetLaserscanThetaListM (n)	Fordert eine LaserscanThetaListM mit Daten zu einer übergebenen Liste von Winkeln an.
FeaturesM (e)	Enthält Daten einer Liste von Features aus Einzel- oder Stereobildern.
SetFeatureDetectorM (u)	Wählt einen Feature Detektor Typ aus (SIFT, KLT, etc.).

Tabelle 4.1: Nachrichtenübersicht (n: neue Nachricht , e: erweiterte Nachricht und u: unveränderte Nachricht).

4.4 Module

Das “Core System” hat keinerlei Informationen über die Funktionalität des Programms; diese wird durch die vorhandenen Module festgelegt, die neben der Erstellung von Nachrichten auch solche empfangen können, was in anderen Teilen des Frameworks nicht möglich ist. Da nicht jedes Modul alle Nachrichten empfangen und darauf reagieren soll, kann jedes Modul individuelle Nachrichten abonnieren. Listing 4.3 zeigt anhand des “ActiveLaserScannerModules” ein Beispiel hierfür; dieses Modul dient dazu, zu einer übergebenen Liste von Winkeln die zugehörigen Laserscans zu beschaffen. Dabei wird innerhalb des Moduls die Position des Servomotors gesetzt und anschließend der Laserscan angefordert. Die Anforderung eines “dummy” Laserscans ist notwendig, da der Servomotor sich während des ersten Scans bewegen kann und somit die Daten verfälscht- siehe Abschnitt 5.2 für genauere Zusammenhänge. Würde man die Vermessung mit Nachrichten setServoM und LaserDataM umsetzen, kann keine zuverlässige Aussage über die notwendige Zeit zur Vermessung dieser Winkel getroffen und die Prozedur kann zudem nicht schnellstmöglich durchgeführt werden. Abschließend stellt Tabelle 4.2 alle wichtigen Module in einer Übersicht mit kurzer Beschreibung des Verwendungszweckes vor.

Listing 4.3: Beispiel zur Abonnierung von Nachrichtentypen eines Moduls.

```
/* Abonnierte Nachrichtentypen im Konstruktor */
ActiveLaserScannerModule::ActiveLaserScannerModule(){
  addMessageType( MessageTypes::SYSTEM_RESET_M );
  addMessageType( MessageTypes::GET_LASERSCAN_THETALIST_M );
}
```

Listing 4.4: Beispiel zum Laserscannen bestimmter Winkel.

```
// 1. Setzt die Position des Servomotors auf den übergebenen
   Winkel der Liste
mp_servoController->setHighPWMtime( mp_servoController->
   degToTime( ( double ) * m_thetaAngle_it ) );
// 2. Anforderung von zwei Scans, dabei den Ersten verwerfen
vector<int > dummy = mp_laserScanner->getScan();
m_thetaListMap[ *m_thetaAngle_it ] = mp_laserScanner->getScan
   ();
m_thetaAngle_it++;
```

ActiveLaserScannerModule (n)	Direkte Kommunikation (ohne SetServoM) mit dem ServoController- und LaserScannerdevice; empfängt “GetLaserscanThetaListM” Nachrichten und versendet eine LaserscanThetaListM.
ServoControllerModule (e)	Empfängt und verarbeitet “SetServoM” Nachrichten zur Änderung des PWM Signals vom Mikrocontroller, um die Position des Servomotors zu ändern.
LaserScannerModule (n)	Versendet ohne Aufforderung “LaserDataM” Nachrichten in gewissen Zeitabständen.
FireGrabberLightModule (u)	Empfängt “GetImageM” Nachrichten und versendet daraufhin “ImageM”.
FireGrabberModule (u)	Empfängt “GetImageM” und versendet “ImageM” Nachrichten, jedoch im Vergleich zum “LightModule” mit nur definierten Kameras.
PositionEstimationModule (n)	Zur Berechnung der Roboterpositionen mit Hilfe verschiedener Verfahren.
FeaturesModule (e)	Empfängt “ImageM” Nachrichten und detektiert in diesen Features.
CalibrationModule (e)	Bestimmung der Servopositionierungsgenauigkeit.
TsaiCalibrationModule (u)	Testversion einer Kalibrierung nach Tsai zur Abbildung von 2D- auf 3D-Koordinaten.
MessageLoggerModule (u)	Empfängt und speichert Nachrichten in einer Logdatei.
MessagePlaybackModule (u)	Lädt Nachrichten aus einer Logdatei und spielt diese ab.

Tabelle 4.2: Modulübersicht (n: neues Modul, e: erweitertes Modul und u: unverändertes Modul).

Kapitel 5

Experimente und Ergebnisse

Die theoretischen Aspekte und Probleme der Selbstlokalisation sowie der Idee des kombinierten Ansatzes sind bereits aus Kapitel 2 und 3 bekannt. In diesem Kapitel soll anhand durchgeführter Versuche und deren Ergebnisanalyse gezeigt werden, ob es mit den zur Verfügung stehenden Komponenten möglich ist, bildbasiertes 3D-Laserscannen in die Praxis umzusetzen. Dabei geht es um den Versuch, eine Antwort auf die Fragen: "'Wie groß oder klein darf ein Objekt in der Realität und im aufgenommenen Bild sein, um es vermessen zu können?" sowie: "'Welche Fehlerquellen (Messfehler) beeinträchtigen den Gesamtablauf einschließlich der Berechnung der 3D-Weltposition?" zu finden. Die in den folgenden Abschnitten durchgeführten Versuche sollen Antworten auf diese Fragen liefern.

In Abschnitt 5.1 werden Genauigkeitsmessungen von Servomotor und Laserscanner behandelt. Im Anschluss daran wird die Performance von SIFT-Features sowie der Ansteuerung des Servomotors und des aktiven 3D-Laserscannens in Abbschnitt 5.2 analysiert. Abschließend folgt eine Bewertung der gewonnenen Erkenntnisse. Tabelle 5.1 gibt eine Übersicht über die verwendeten Komponenten der Versuche.

Kamera	Digital Interface DFW-VL500 (optischer 12x Zoom, $f = 5.4 - 64.8mm$)
2D-Laserscanner	HOKUYO URG-04LX
	($LxBxH = 50x50x70mm$, $160g$, $240°$ Scan, $0.36°$ Auflösung, $+/-1cm$ Genauigkeit)
Servomotor	HiTEC Digital ROBOT SERVO HSR-8498HB
	(PWM-Interface, Betriebswinkel: $0 - 180°$, Betriebsspannung: $6 - 7.4V$)
Mikrocontroller	Crumb8-USB (ATmega8, USB-UART Wandler, $14.7456MHz$ XTAL Quarz)
Laserpointer	Laserdiode (Wellenlänge: $630 - 680mm$, Ausgangsleistung: max.$1mW$)

Tabelle 5.1: Komponentenübersicht

5.1 Genauigkeit

In der Praxis treten folgende Genauigkeitsfehler auf, die Auswirkungen auf die Größe des vermessenen Objektes haben:

- Die Pixelgröße des Features im Bild, die abhängig von der Bildauflösung ist.
- Eine fehlerhafte Bestimmung der Punktkorrespondenzen - Featurepaar zweier Bilder mit vorgegebener minimaler euklidischer Distanz.
- Der berechnete vertikale und horizontale Winkel (θ,ϕ) des 3D-Laserscanners durch Kalibrierungsungenauigkeiten.
- Positionierungsfehler des Servomotors - bei dem hier verwendeten Aufbau vom vertikalen Winkel (θ).
- Laserscannerrauschen und -messfehler in den Entfernungsdaten.

Welche Fehlerquellen den Gesamtablauf beeinflussen ist damit beantwortet, aber die Frage, wie groß der jeweilige Fehler und die Auswirkungen sind, bleibt bislang ungeklärt.

5.1.1 Servomotor

Eine Untersuchung der Stellgenauigkeit des Servomotors ist demzufolge notwendig, da der Positionswinkel θ den Winkel festlegt um den sich der Servomotor zur horizontalen

Horizontaler Abstand [cm]	100	100	100	100	200	200	200	200
Vertikaler Winkel [grad]	5	5	10	10	5	5	10	10
Winkelungenauigkeit [grad]	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0
Abweichung [cm]	0.88	1.76	0.90	1.81	1.76	3.52	1.80	3.61

Tabelle 5.2: Abweichung der y-Koordinate eines Weltpunktes in Abhängigkeit vom horizontalen Abstand, vertikalen Winkel (0 Grad $\hat{=}$ Horizontalachse) und dessen Winkelungenauigkeit bei der Positionierung des Servomotors.

Achse drehen soll und mit diesem weitere Berechnungen durchgeführt werden, zum Beispiel bei der Vermessung eines 3D-Weltpunktes mit errechnetem Winkel θ. Je weiter der zu vermessene Objektpunkt vom Laserscanner entfernt ist, desto größer sind die Folgen einer Abweichung ϵ vom eigentlich zuvor berechneten Winkel. Diese Vermutung zeigt sich auch anhand der Bestimmung des Laserscannerabstandes zur Rotationsachse, die mit Hilfe zweier Laserstrahlen durchgeführt und wie bereits in Abbschnitt 3.2.2 behandelt, mathematisch berechnet wird. Durch die anscheinend zu großen Positionierungsungenauigkeiten des Servomotors lässt sich der Abstand nicht eindeutig bestimmen, da die Ergebniswerte zwischen 35 und 125 Millimeter liegen. Der von Hand gemessene Wert mittels Zentimetermaß ergibt einen ungefähren Abstand von 85 Millimeter.

In Tabelle 5.2 sind einige Abweichungen in Abhängigkeit zur Entfernung berechnet worden jedoch unter Vernachlässigung des Abstandes zwischen Laserscanner und Rotationsachse. Bei Berücksichtigung dieses Abstandes ist die positive Abweichung im Vergleich zur negativen noch größer.

Der Versuchsaufbau zur Berechnung der Stellgenauigkeit benutzt den bereits bestehenden Aufbau und ersetzt lediglich den Laserscanner durch einen Laserpointer, wie in Bild 5.1 zu sehen ist. Der Laserpointer wird danach senkrecht zu einer möglichst hellen und einfarbigen Wand mit einem Abstand von 100 cm positioniert. Der Abstand sollte so gewählt sein, dass im Kamerabild alle angesteuerten Positionen des Laserpunktes auf der Wand zu sehen sind und gleichzeitg die gesamte Bildhöhe ausgenutzt wird. Bevor auf die einzelnen Schritte des Versuchsablaufes eingegangen wird, sind die anzusteuernden Positionswinkel festzulegen, zum Beispiel von -10 bis $+10$ Grad in 5 Grad-Schritten.

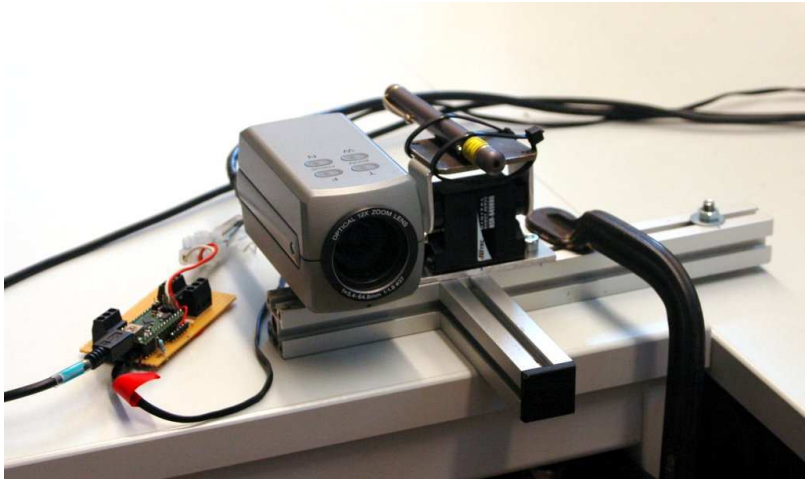

Bild 5.1: Versuchsaufbau zur Berechnung der Stellgenauigkeit des Servomotors.

Die Versuchsdurchführung lässt sich in folgende Schritte aufteilen:

1. Ein Referenzbild der Wand, ohne Laserpunkt, aufnehmen.
2. Position des Laserpointers ändern (z. B. -10 Grad).
3. Ein Laserpointerbild der Wand, mit Laserpunkt, aufnehmen.
4. Berechnung der Position des Laserpunktes im Bild.
   - Ein Differenzbild aus Laserpunkt- und Referenzbild erstellen.
   - Den Schwerpunkt einer Region bestimmter Größe und Mindesthelligkeit berechnen, die dem Laserpunkt entspricht. Da Graubilder verarbeitet werden, ist eine Farbdetektion des Laserpunktes nicht möglich.
5. Berechnung des Mittelwertes und der Standardabweichung der Positionen im Bild pro Winkel.
6. Den Vorgang ab Schritt 2 wiederholen bis die Berechnung in Schritt 5 mit einer festgelegten Fehlertoleranz konvergiert oder nach einer vorgegebenen Anzahl von Wiederholungen.

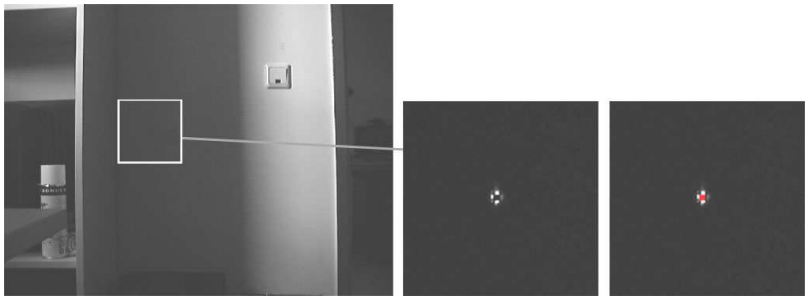

Bild 5.2: Referenzbild mit Bildausschnitt des Differenzbildes und des gefundenen Laserpunktes mit eingezeichnetem Schwerpunkt (v.l.n.r).

Bild 5.2 zeigt ein Beispiel für ein Referenz-, Laserpointer- und Ergebnisbild sowie Tabelle 5.3 die Werte des durchgeführten Versuches. Aus den Werten kann entnommen werden, dass die Standardabweichnung zwar gering, die Streuung aber sehr groß ist. Um die Einheit der Abweichung von Pixel in beispielsweise Zentimeter umrechnen zu können ist eine Kamerakalibrierung notwendig.

Die dargestellten Graphen in Bild 5.3, 5.4 veranschaulichen die erhobenen Daten und machen deutlich, dass je nach Winkelabstand unterschiedliche Abweichungen vom eigentlich anzusteuernden Winkel auftreten. Aufgrund von Hardwareproblemen konnten nicht mehr als 10 Einzelmessungen durchgeführt werden, wodurch der Mittel- und Standardwert nicht optimal bestimmt werden kann. Bei der Ansteuerung von Positionen sollte der minimale und maximale Abstand zum Mittelwert möglichst nahe am eigentlichen Wert liegen, da die Positionen nur einmal und nicht mehrfach angesteuert werden sollen. Aus

Winkel [grad]	Y-Koordinate [pixel]				
θ	Min	Max	Mittelwert	Standardabweichung	Streuung (Max-Min)
-10	31.0	32.2	31.7	0.5	1.2
-5	105.9	110.4	107.3	1.4	4.5
0	172.6	174.5	173.4	0.7	1.9
5	246.7	248.5	247.4	0.6	1.8
10	314.0	316.7	315.3	0.6	2.7

Tabelle 5.3: Messwerte der berechneten Schwerpunktkoordinate des Laserpointers zur Bestimmung der Positionierungsungenauikgeiten.

diesem Grund ist es sinnvoll, einen anderen Motor zur Ansteuerung der Position zu verwenden, beispielsweise einen Schrittmotor. Genauere Analysen und Aussagen über das Positionierungsverhalten kann mit Hilfe eines Drehwinkelgebers gegeben werden. Solange die Positionierung des Laserscanners nicht auf mindestens 0.5 Grad genau ist, kann davon ausgegangen werden, dass eine fehlerfreie Vermessung kaum realisierbar ist.

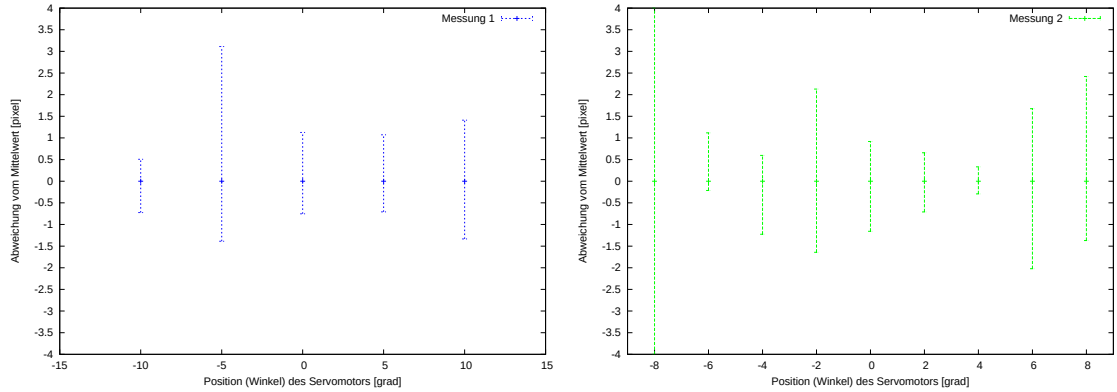

Bild 5.3: Maximale und minimale Abstände zum Mittelwert, berechnet aus 10 Einzelmessungen, mit Winkelabständen von 5 Grad im linken Graphen und 2 Grad im Rechten. Der linke Graph visualisierst außerdem die in Tabelle 5.3 berechneten Werte.

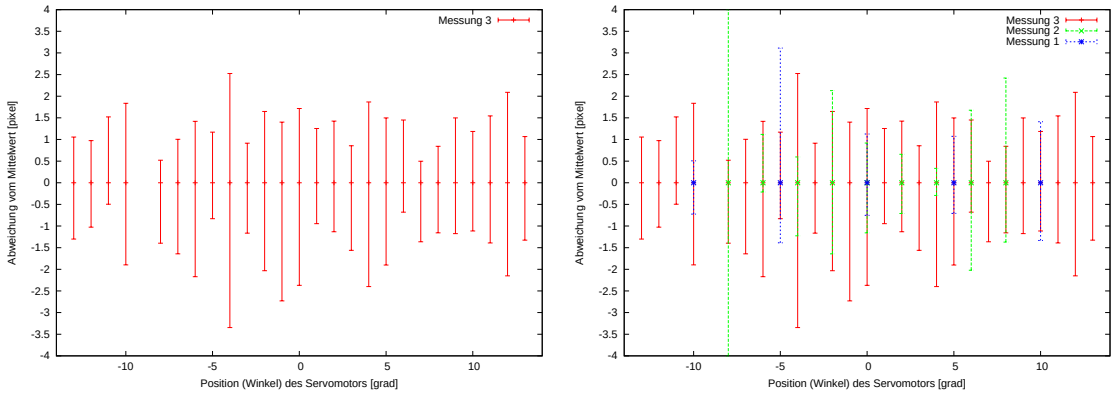

Bild 5.4: Graph links: Maximale und minimale Abstände zum Mittelwert, berechnet aus 10 Einzelmessungen, mit Winkelabständen von einem Grad. Graph rechts: Alle drei Messungen in einem Graph übereinandergelegt mit teilweise extremen unterschieden.

Scanindex	209	210	211	212	213	214	215	216	217
Mittelwert	908.1	920.3	924.4	924.4	923.3	913.9	911.5	910.4	911.1
Standardabweichung	5.1	4.3	3.8	4.0	4.1	4.7	3.5	3.4	3.3
Scanindex	218	219	220	221	222	223	224	225	226
Mittewert	912.0	912.9	912.6	912	911.6	911.5	910.8	910	910.5
Standardabweichung	3.0	3.0	3.1	3.4	3.7	3.4	3.5	3.6	3.5

Tabelle 5.4: In dieser Tabelle ist ein Ausschnitt aus Bild 5.7 ausgewählt und die jeweiligen Entfernungen des Mittelwertes sowie der Standardabweichung aus 100 Scans ausgewertet.

5.1.2 Laserscanner

Die Genauigkeit eines Laserscanners kann gemessen werden durch Betrachtung der Ausdehnung des Laserstrahls und der dabei verwendeten Technik, die an dieser Stelle unberücksichtigt bleibt und der gemessenen Enfernungsdaten; letztere gilt es nun genauer zu betrachten. Tabelle 5.4 stellt eine Auswertung eines Winkelausschnittes zwischen 58 und 61 Grad (Indexposition 209 und 217) von 100 Einzelscans dar. Der Laserscanner befand sich zur Zeit der Aufnahme vor einer Pappwand mit einer Entfernung von ca. einem Meter und war parallel zu dieser ausgerichtet.

Aus den Werten der Tabelle und Gnuplot-Darstellungen - Bild 5.5, 5.6 und 5.7 - derselben Messdaten (100 Scans) ist ein sehr gleichmäßiges Rauschen sowie eine fast anhaltende mittlere Position des Mittelwertes zwischen den Mindest- und Maximalentfernungen erkennbar. Die Standardabweichung hingegen schwankt sehr stark zwischen 3 und 5 Millimetern, wodurch es sehr schwer ist, eine Anpassungsfunktion zu finden, die diese Schwankungen abbildet. In Bild 5.6 und 5.7 ist eine Anpassungsfunktion vom Polynom 7. Grades dargestellt, die sich an die Werte der Standardabweichung bestmöglichst annähert aber nicht ausreicht. Eine Möglichkeit, um eine bessere Anpassung zu erreichen, ist die Trennung in mehrere Winkelbereiche (Scanindex) wie zum Beispiel von 100 bis 200, 200 bis 400 und 400 bis 600 und jeweils eine zu diesen Werten individuelle Abbildungsfunktion zu finden. Außerdem sollte eine Liste generiert werden, die Ausreißerwerte in den Scans findet und als defekt markiert. Diese defekten Scanindexwerte werden bei der Weiterverarbeitung der Laserscannerdaten dann nicht mehr verwendet.

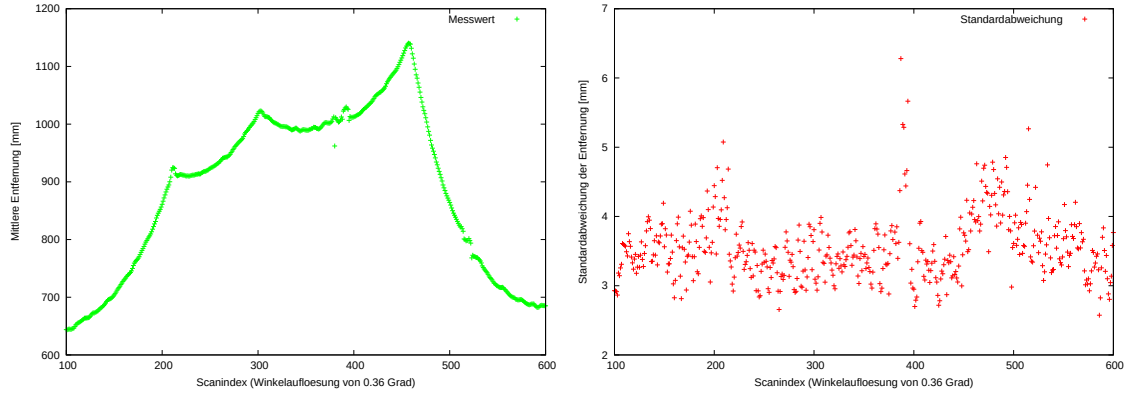

Bild 5.5: L: Mittelwert des jeweiligen vertikalen Winkels. R: Standardabweichung.

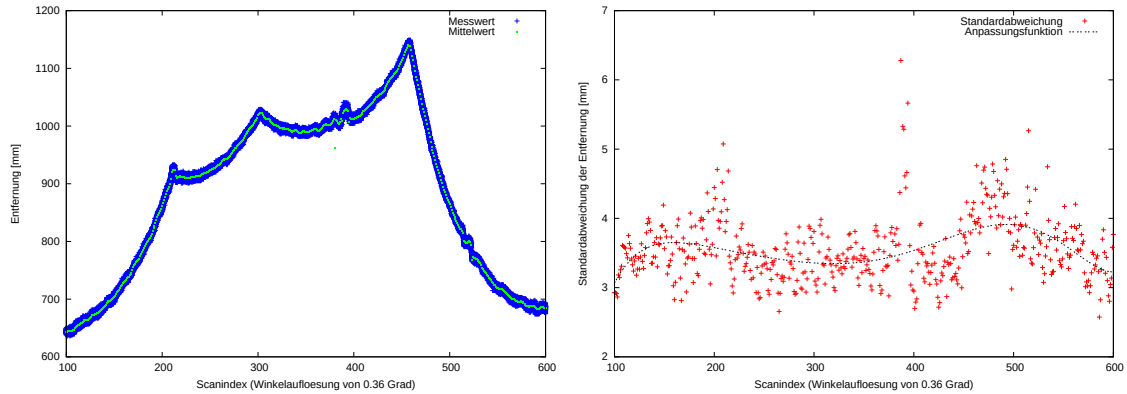

Bild 5.6: L: Mittelwert und alle Messwerte. R: Anp.funktion vom Polynom 7.Grades.

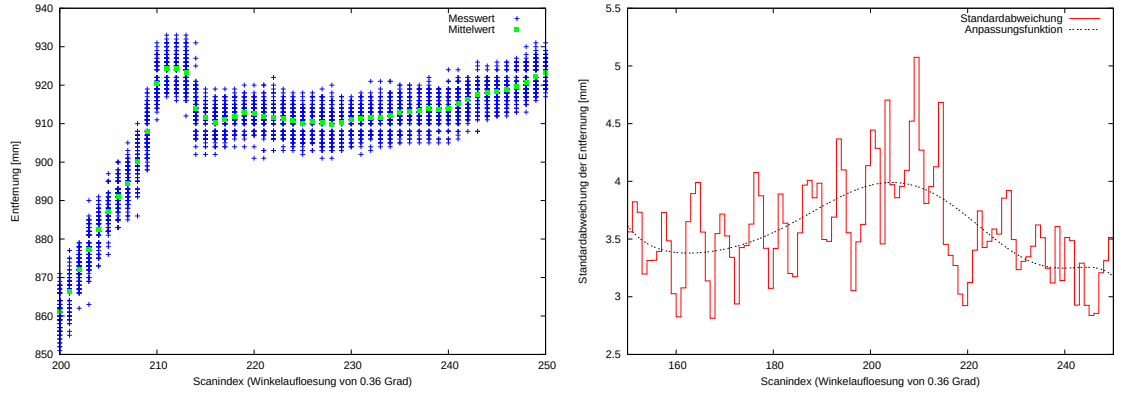

Bild 5.7: L,R: Ausschnitte aus Abbildung 5.6.

5.2 Performance

Neben der Genauigkeit und Robustheit des 3D-Laserscannens ist es ebenso wichtig, die Geschwindigkeit des Verfahrens zu untersuchen. Zwar kann zu diesem Zeitpunkt noch keine Aussage über die Gesamtgeschwindigkeit gegeben werden, jedoch über die meisten Teilprozesse. Dabei wird zunächst die Performance von Featuredetektion und -matching analysiert, die unabhängig vom eigentlichen Vermessen mittels Laserscanner ist. Die weiteren Zeitmessungen sind von der Ansteuerung des Servomotors und des aktiven 3D-Laserscannens sowie der verwendeten Komponentenhardware abhängig; womit nicht die Rechenleistung selbst gemeint ist.

5.2.1 Detektion und Matching von SIFT-Features

Die Geschwindigkeit der SIFT-Feature Algorithmen sind, wie die Ergebnisse in Tabelle 5.5 zeigen, von der Bildauflösung und der Anzahl gefundener Features abhängig. Die gemessenen Zeiten wurden auf einem Intel Centrino Notebook mit $1.8GHz$ und dem Betriebsystem Linux (OpenSuse 10.2) erhoben und verwenden die SIFT Implementierung von David G. Lowe. Da die Performance neben der Auflösung des Bildes auch sehr stark von der Rechenleistung abhängt, ist es sinnvoll, die Berechnungen nicht auf dem Roboter durchzuführen, sondern auf den Clientrechner auszulagern. Die Bilder mit größerer Auflösung als $640x480$ sind mittels kubischer Interpolation skaliert worden, da die Digital Interface Kamera nur Aufnahmen mit maximaler Auflösung von $1024x768$ liefert. Vergleicht man die Rechenzeiten und Faktorwerte aus den Tabellen 5.5 und 5.6 miteinander, dann kann festgestellt werden, dass das Matching der Features mit ansteigender Auflösung um ein Vielfaches stärker anwächst als bei der Featuredetektion. Um die Peformance hierbei zu verbessern, sollte schon vor dem Matching versucht werden, die Featureanzahl durch eine Vorauswahl zu reduzieren. Ein Kriterium für solch eine Vorauswahl kann zum Beispiel die Feature-Position im Bild sein, die einen Mindestabstand zum Bildrand oder einer Bildfläche erfüllen muss.

Kamerabild	Auflösung	Featureanzahl	Rechenzeit	Faktor
robbie9_01_1024.pgm	1024x768	2315	12.12s	7.2
robbie9_02_1024.pgm	1024x768	2076	11.15s	9.4
robbie9_01_800.pgm	800x600	1314	6.75s	4.0
robbie9_02_800.pgm	800x600	1314	6.90s	5.8
robbie9_01_640.pgm	640x480	761	4.74s	2.8
robbie9_02_640.pgm	640x480	633	4.48s	3.8
robbie9_01_320.pgm	320x240	186	1.69s	1.0
robbie9_02_320.pgm	320x240	193	1.19s	1.0

Tabelle 5.5: Detektion von SIFT-Features unterschiedlicher Graubildaufnahmen und Auflösung. Der Faktor ist das Verhältnis der Rechenzeit des Bildes mit der Auflösung $320x240$ zur aktuellen Auflösung. Quelle: VPEToolkit.

Auflösung	Featureanzahl		Feature-Matching			Rechenzeit	Faktor
	Bild 1	2	Bild 1→2	Bild 2→1	Gesamt		
1024x768	2315	2076	66	58	36	214.2s	142.8
800x600	1314	1087	14	17	12	62.4s	41.6
640x480	761	633	9	11	8	21.2s	14.1
320x240	186	193	2	2	2	1.5s	1.0

Tabelle 5.6: Matching von SIFT-Features zweier Bilder aus Tabelle 5.5 mit gleicher Auflösung und maximalen, euklidischen Abstand 100. Der Faktor ist das Verhältnis der Rechenzeit des Bildpaares mit der Auflösung $320x240$ zur aktuellen Auflösung. Quelle: VPEToolkit.

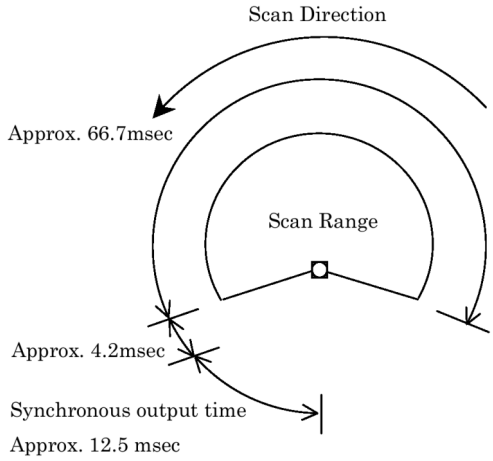

Bild 5.8: Laserscannerzeiten eines Scandurchlaufes. Quelle: [URG05].

5.2.2 Aktives 3D-Laserscannen

Bevor die Zeitmessungen analysiert werden können ist es notwendig, dass vom 2D-Laserscanner gesendete Sychronisationssignal und das hardwarebedingte Zeitverhalten eines Scandurchlaufes zu kennen. Bild 5.8 gibt einen Überblick über die verschiedenen Teilzeiten eines Scans, die sich aus dem eigentlichen Scan ($\sim 66.7ms$), einer kurzen Pause ($\sim 4.2ms$), dem Sync-Signal ($\sim 12.5ms$) und einer Restzeit ($\sim 16.6ms$) zusammensetzt. Dabei hat das ausgegebene Sync-Signal während der 12.5 Millisekunden einen Pegel von annähernd 0 Volt und vor, beziehungsweise nach dieser Zeit einen Pegel von ca. 5 Volt. Ist die Sychronisation beim Mikrokontroller eingeschaltet, so wird die Änderung des PWM Signals erst bei fallender Flanke, also beim Übergang von 5 auf 0 Volt gesetzt, was zur Folge hat, dass die Position des Servomotors erst alle 100 Millisekunden geändert werden kann.

In Tabelle 5.7 wird die erforderliche Zeit gemessen, die benötigt wird, das PWM-Signal und damit eine Positionsänderung des Servomotors zu bewirken, was die zuvor gemachte Annahme bestätigt. Die Dauer einer Positionsänderung des Servomotors vom Start der Bewegung bis zum Erreichen der Endposition ist damit aber nicht bestimmt. Dies kann aber mit Hilfe eines Drehwinkelgebers berechnet werden, um beispielsweise herauszufinden wie lange eine Positionsänderung von 45 Grad dauert. Tabelle 5.8 zeigt die Messergeb-

nisse eines aktiven 3D-Scans, also die Positionierung des Servomotors auf der vertikalen Achse und die Auswertung eines empfangenen 2D-Laserscans. Da im Moment nicht gewährleistet werden kann, dass der Servomotor zum Zeitpunkt des Laserscans seine Endposition erreicht hat, werden zwei Laserscans durchgeführt und nur der zweite ausgewertet, was etwa 200 Millisekunden dauert.

Messzeit [ms]					Laserscanner
Messungen	Min	Max	Mittelwert	Standardabweichung	Synchronisation
10	8	8	8.00	0.00	off
100	7	9	8.00	0.20	off
100	4	8	4.25	0.88	off
200	3	5	4.00	0.17	off
10	60	101	95.60	12.58	on
100	63	101	99.27	3.87	on

Tabelle 5.7: Die Tabelle zeigt die erforderliche Zeit, um von der Robbie-Software aus das PWM-Signal am Mikrocontroller zu ändern. Die Position des Servomotors ändert sich aber erst in der nächsten Periode des PWM-Signals, die mit der derzeitigen Konfiguration bei max. $2ms$ liegt. Quelle: Robbie8-Framework

Messzeit [ms]						
Messungen	Position		Scan		Mittelwert	Standardabweichung
	Min	Max	Min	Max		
10	57	67	33	36	98.8	2.82
100	60	68	33	38	99.57	0.82
10	60	67	31/97	37/100	198.4	2.99

Tabelle 5.8: Im Vergleich zur Tabelle 5.7 wird hier die Dauer gemessen, die notwendig ist, um die Position des Servos (PWM-Signal) zu ändern und anschließend einen Laserscan durchzuführen. Das Sync-Signal vom Laserscanner wird ungefähr alle $100ms$ mit einer fallenden Flanke ausgelöst. In der letzten Zeile sind die Messwerte einer Positionsänderung und zwei Laserscans aufgeführt.

5.3 Validierung

Die Ergebnisse der vorherigen Abschnitte zeigen, dass eine Aussage über die Robustheit und Geschwindigkeit der Einzelkomponenten und damit eine Einschätzung dieser in Bezug auf den Gesamtablauf gemacht werden kann. Die Ungenauigkeiten der Servopositionierung in Abhängigkeit zur Entfernung der Objektpunkte (der Szene) scheinen größere Auswirkungen auf das Gesamtergebnis zu haben als die des Laserscanners. Beim Laserscanner muss jedoch versucht werden, defekte Indexpositionen eines Scans zu erkennen damit diese in weiteren Berechnungen nicht verwendet werden. Das Laserscannerrauschen ist überwiegend gleichmäßig und damit berechenbar. Die Performance von SIFT ist, solange keine hohen Auflösungen und viele SIFT-Features getrackt werden müssen, sehr effektiv. Das aktive Vermessen von beispielsweise 4 Objektpunkten erfordert bei idealen Bedingungen, also keinen Mehrfachmessungen aufgrund von Genauigkeitsfehlern, etwa 0.8 Sekunden. Aus den praktischen Experimenten zeigt sich, dass zwar eine Umsetzung des Verfahrens möglich ist, aber unter unzureichenden Performance- und Genauigkeitsbedingungen leiden würde.

Tiefenbilder Eine Aussage über den Informationsgehalt und ein ungefähres Genauigkeitsmaß von 3D-Laserscans kann mit Hilfe von Tiefenbilder gemacht werden, siehe Bild 5.9. Die drei Aufnahmen sind dabei mit unterschiedlichen Auflösungen des vertikalen Winkels - vom Servomotor angesteuert - erstellt worden. Die horizontale Auflösung bildet dabei die 682 Werte eines Scans ab.

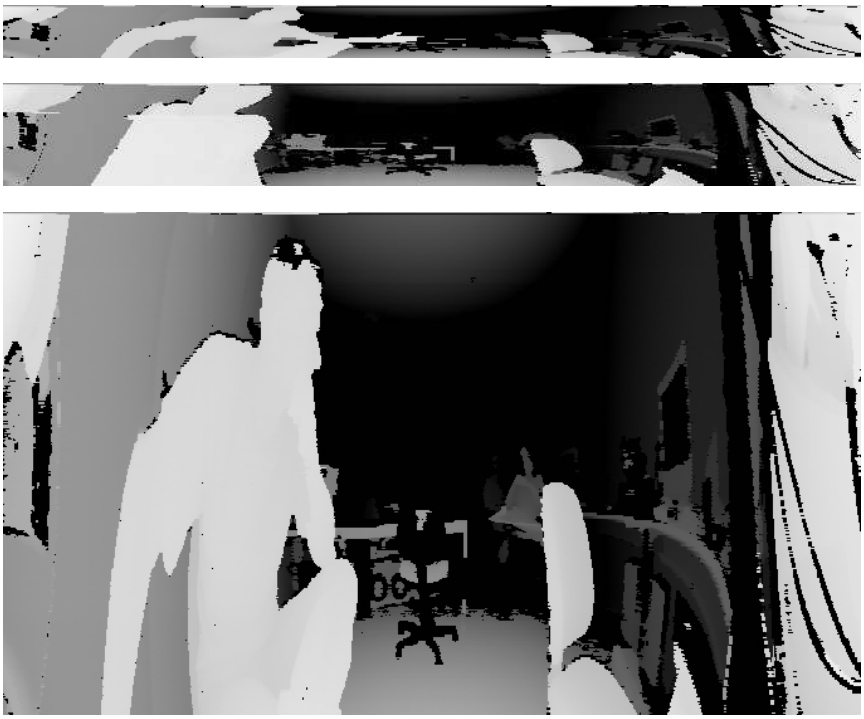

Bild 5.9: Visualisierung der 3D-Laserscans mit unterschiedlicher vertikaler Auflösung. Der Grauwertebereich von 0 bis 255 bildet dabei weit entfernte oder unmessbare Werte auf nahe Objekte ab.

Kapitel 6

Diskussion und Ausblick

In dieser Arbeit konnte gezeigt werden, dass es zwar theoretisch möglich ist, durch die Kombination von Informationen aus Kamerabildern und Laserscannerdaten, 3D-Weltpunkte zu vermessen. Bei der praktischen Umsetzung traten allerdings zu viele Ungenauigkeiten auf. Das kombinierte Verfahren sollte außerdem zeigen, dass es zur Selbstlokalisation von mobilen Robotern verwendet werden kann. Anderen Verfahren, wie beispielsweise der "Visuellen Odometrie" mittels Stereokamerasystem, soll es hinsichtlich Geschwindigkeit und Robustheit überlegen sein. Da bislang aber kein direkter Vergleich vorlag, kann nur eine Prognose abgegeben werden, die auf den durchgeführten Experimenten beruht. Diese zeigen die Tendenz, dass 3D-Weltpunkte nur sehr unpräzise angesteuert und vermessen werden können. Da die Lokalisationbestimmung von der Genauigkeit der 3D-Weltpunkte abhängt, scheint das hier entwickelte Verfahren zumindest in dieser Form noch nicht einsetzbar.

Für die Detektion und das Matching von Bildmerkmalen erwies sich der anfangs ausgewählte SIFT-Operator als durchaus geeignet. Es galt wenige Features zu detektieren, diese jedoch robust, was der Operator leistete. Ob er für diese Aufgabe die optimale Wahl darstellt wurde nicht bewiesen, die einhelligen Meinungen der Fachliteratur weisen jedoch darauf hin, dass er trotz seiner geringen Geschwindigkeit für das gestellte Problem eine geeignete Lösung darstellt.

Im Robbieprojekt hat sich die Verwendung eines selbst entwickelten “low-cost” Wärmesensors bereits bestens bewährt und große Kosten eingespart, weswegen ein individueller Aufbau des 3D-Laserscanners angedacht wurde. In der Arbeit wurde dieser Aufbau theoretisch und praktisch realisiert, was sowohl die Konstruktion eines Rotationsgelenks und die Montage des 2D-Laserscanners darauf beinhaltete. Insgesamt stellt dies einen flexiblen Aufbau dar, der dadurch auf dem mobilen Roboter “Robbie-8” leicht angebracht werden kann.

Die Ansteuerung des Servomotors wurde durch die Programmierung des Mikrocontrollers umgesetzt. Die erreichte Schrittgenauigkeit der Servopositionierung ist vergleichbar mit anderen kommerziellen Produkten, mit ähnlich eingesetzten Bauteilen sowie einer kompakten Bauform. Die erzielten Ergebnisse des Aufbaus sind also mit denen eines kommerziellen 3D-Laserscanners vergleichbar, zeigen aber, dass diese für das Verfahren zu ungenau sind.

Die erreichte Schrittgenauigkeit des konstruierten “low-cost” 3D-Laserscanners reicht für das Verfahren nicht aus. Die kommerziell erhältlichen Alternativen jedoch besitzen gut vergleichbare technische Randdaten was darauf schließen lässt, dass diese keine wesentlich besseren Ergebnisse liefern würden. Der Austausch des Servomortors durch einen Schrittmotor würde die Genauigkeit jedoch steigern. In Kombination mit einem Winkelgeber könnte eine genauere Positionierung des Laserscanners erreicht werden beziehungsweise eine Korrekturmöglichkeit bieten. Die Messung der Ansteuerungsgenauigkeit wurde in dieser Arbeit mit Hilfe eines Laserpointers und dessen Positionsbestimmung im Bild bestimmt, die zwar eine erste Aussage über die Genauigkeit ermöglicht, aber mittels Winkelgeber genauer wäre. Ein Vorteil, der sich aus der eigenen Umsetzung des 3D-Laserscanners ergibt, ist die enorme Kosteneinsparung und Flexibilität. Im Vergleich zu einem kommerziellen, in sich geschlossenen System, können außerdem Teile des Aufbaus an wechselnde Ansprüche angepasst und weiterentwickelt werden.

Tests des Verfahrens wurden noch nicht auf dem mobilen Roboter selbst durchgeführt, da die Kalibrierung von 3D-Laserscanner und eines Monokamerasystems zum Ende der Arbeit noch aussteht beziehungsweise nicht zur Verfügung stand.

Abschließend kann gesagt werden, dass eine Verbesserung der Positionierungsgenauigkeit nur durch weitere Forschungen im Bereich der bildbasierten Verfahren erreicht werden kann. Dabei sind nicht nur Verfahren mit Stereo-Kamerasystemen zu verfolgen sondern auch Mehrfach-Kamerasysteme. Je mehr Kameras eingesetzt werden können, desto genauer ist die Berechnung der 3D-Weltpunkte möglich. Durch die aktive Vermessung mit Hilfe eines Laserscanners kann dieses Ziel, nach den Erkenntnissen der vorliegenden Arbeit, so nicht erreicht werden.

Anhang A

Berechnungen

Euklidisches Abstandsmaß

Der euklidische Abstand d in einem n-dimensionalen Raum für zwei Punkte oder Vektoren lässt sich wie folg berechnen:

$$\mathrm{d}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{A.1}$$

mit zwei beliebigen Punkten gleicher Dimension x und y gegeben durch die Koordinaten $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_n)$.

Auflösung der ansteuerbaren Servopositionen

Die Auflösung, mit der sich der Servomotor positionieren lässt, hängt von verschiedenen Hardwarefaktoren ab. Zum Einen von der maximalen Frequenz des Servomotors und zum Anderen von Rahmenbedingunen des verwendeten Mikrocontrollermoduls hinsichtlich Quarz, Prescaler, Timer und TOP-Wert. Mit den Gleichungen A.2 und A.3, die aus dem jeweiligen Mikrocontrollerdatenblatt [ATM04] entnommen werden können, lassen sich zum Einen die Frequenz F in Abhängigkeit zum TOP-Wert berechnen und zum An-

deren die Auflösung R des PWM-Signals. Hierbei ist zu beachten, welche Art von PWM-Signal generiert wird. In diesem Fall wird ein phasen- und frequenzkorrektes PWM-Signal erzeugt was auf Grund seiner Eigenschaften besonders gut für die Ansteuerung von Motoren geeignet ist.

$$F_{PWM} = \frac{Quarz}{2 * N * TOP} \tag{A.2}$$

$$R_{PWM} = \frac{log(TOP + 1)}{log(2)} \tag{A.3}$$

Mit N: Prescalerteiler ($1, 8, 64, 256$ oder 1024) und TOP: < 256 (8-Bit Timer) und < 65536 (16-Bit Timer).

$$T[Hz] = \frac{1}{F[sec]} \tag{A.4}$$

Die maximale Auflösung bei einem Quarz von $14,7456$MHz, kann mit $N = 1$ und $TOP = 65535$ erreicht werden. Man erhält damit eine Frequenz von $112,5$ Hz. Da nicht alle Servomotoren dieser Frequenz standhalten, ist es notwendig, den nächst höheren Prescalerteiler $N = 8$ auszuwählen. Mit $TOP = 65535$ erhalten wir eine Frequenz von $14,06$Hz. Diese relativ kleine Frequenz hat zur Folge, dass die Zeit in der sich das Signal ändern kann auf ca. 71ms anwächst; bei 112.5Hz sind dies nur ca. 9ms, siehe Gleichung A.4. Hieraus wird ersichtlich, dass ein an den Anwendungszweck gebundener Kompromiss zwischen einer möglichst kurzen Reaktionszeit T (Periodendauer) und einer hohen Auflösung gemacht wird.

Abstand zwischen Rotationsachsen- und Laserstrahlursprung

Der Abstand h beziehungsweise Radius r zwischen Ursprung der Rotationsachse und Laserscanners lässt sich mit trigonometrischen Funktionen berechnen. In Gleichung A.1 sind alle notwendigen Größen zur Berechnung eingetragen. Gegeben sind die beiden Winkel α und β sowie die Entfernungen l_1 und l_2. Da das Dreieck mit den Punkten R, U_1 und U_2 gleichschenklig ist, kann die Länge des Radius r_1 und r_2 über die Berechnung der Seitenlänge s mit Hilfe des Sinussatzes erfolgen. Um die Strecke s, also die Hypothenuse,

des gleichschenkligen Dreiecks zu bestimmen, ist der Winkel γ und die Strecke d_1 mit Gleichung A.5 zu berechnen.

$$\begin{aligned} d2 &= cos(\alpha) * l_1 \\ d0 &= cos(\beta) * l_2 \\ d1 &= d2 - d0 \\ gamma &= (alpha - beta)/2 \\ s &= d1/cos(gamma) \end{aligned} \tag{A.5}$$

Mit der Länge von s lässt sich dann der Abstand h wie folgt berechnen:

$$\begin{aligned} delta &= (180 - (\alpha + \beta))/2 \\ h &= s * sin(\delta)/sin(\alpha + \beta) \end{aligned}$$

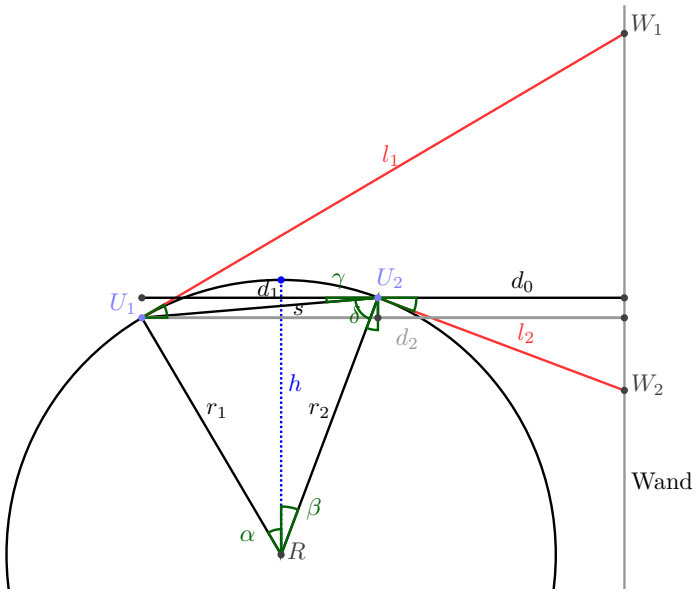

Bild A.1: Berechnung des Abstandes zwischen Rotationsachsen- und Laserscannerursprung mittels trigonometrischer Funktionen.

Anhang B

Verschiedenes

Software

LaTeX-Editor	Kile 1.9.2
Bildverarbeitung	Gimp 2.2.13, Photoshop CS
Vektorgrafiken	Xfig, Illustrator CS
Konvertierung	jpeg2ps, fig2dev
Entwicklungsumgebung	KDevelop 3.5.5
Framework	Robbie-8, VPEToolkit
Betriebsystem	OpenSuse 10.2
Berechnungen/Graphen	Octave, Gnuplot, GeoGebra

Tabelle B.1: Softwareübersicht.

Bilder

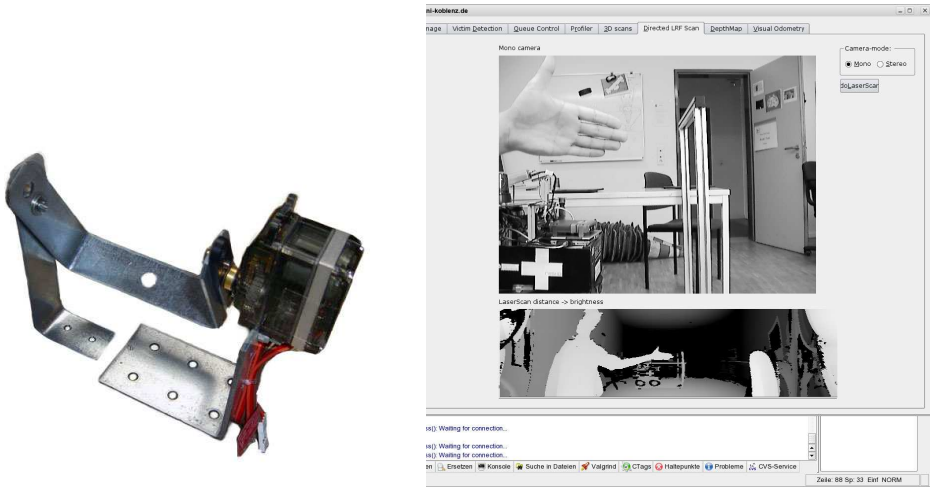

Bild B.1: Links: Rotationsgelenk Prototyp1. Rechts: Robbie-8 GUI mit dargestelltem Kamera- und 3D-Laserscanner Tiefenbild.

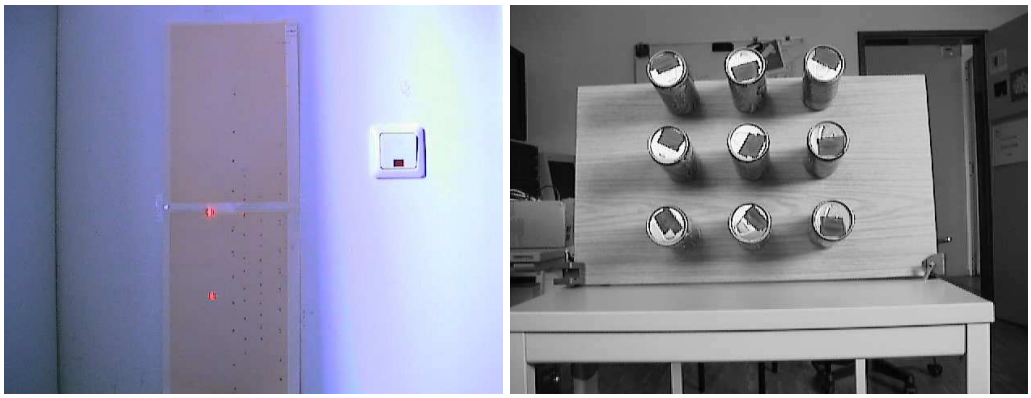

Bild B.2: Links: Manuelle Messung der Servogenauigkeit. Rechts: Kalibrierungsmuster.

Inhalt der beiligenden CD-ROM

Vezeichnis	Inhalt
Ausarbeitung/	Diese Ausarbeitung in elektronischer Form, mit LaTeX-Quellcodes und verwendeten Bildern.
Vortrag/	Folien zum Oberseminarvortrag vom 8.August 2007, mit dazugehörigen LaTeX-Quellcodes und verwendeten Bildern u. Templates.
Technische_Dokumente/	Datenblätter der unterschiedlichen Hardwarekomponenten wie Kamera, Laserscanner, Servomotor, Mikrocontroller und Robbie-8 Architektur.
Experimente/	Alle erstellten und verwendeten Messdaten, die als (binäre-) Objekt-Daten gespeichert sind (.obj), Octave-Daten (.dat), Octave-Quellcode (.m), Gnuplot-Daten (.plot) und Gnuplot-Quellcode (.gnu).
Programmcode/	Implementierte Klassen - Quellcodes zur Ansteuerung des Servomotors, VPEToolkit und Teile des Robbie-8 Frameworks.
Bilder/	Diverse Bilder von den Experimenten, Anwendungsbeispielen.

Tabelle B.2: Inhalt der beiligenden CD-ROM.

Verzeichnis der Bilder

Verzeichnis der Tabellen

Literaturverzeichnis

[Alo] ALOUINI, Mohamed-Slim: *Global Positioning System: An Overview*. citeseer.ist.psu.edu/31114.html,

[ATM04] *ATMEL, Atmega8 Datenblatt*. 2004. – Technische_Dokumente/Mirkrocontroller/Atmega8.pdf

[Dan07] DANG, Thao: *Kontinuierliche Selbstkalibrierung von Stereokameras*, Universität Karlsruhe, Diss., 2007

[Del07] DELIS, Christian: *Entwicklung einer Rotationsplattform für den Hokuyo URG-04LX Laserscanner*. 2007

[DGP] *GPS Technology - DGPS*. http://www.hr-tews.de/GPS/dgps.htm, . – letzter Zugriff: 24. Sep. 2007

[Dic06] DICKSCHEID, Timo: *Markerlose Selbstlokalisation durch Fusion von Sensordaten*, Universität Koblenz, Diplomarbeit, 2006

[Dig] *Sony, Digital Camera Module - Specification*. http://www.inrialpes.fr/grimage/documentation/camera_sony/dfwvl500_ds.en.pdf, . – letzter Zugriff: 24. Sep. 2007

[GDNC+01] GAMINI DISSANAYAKE, M.W.M. ; NEWMAN, Paul ; CLARK, Steven ; DURRANT-WHYTE, Hugh F. ; CSORBA, M.: A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. In: *IEEE Transactions on Robotics and Automation* 17 (2001), Nr. 3

[Gol07] GOLEM.DE (Hrsg.): *Fahrerloses Auto orientiert sich per Laser-3D-Scanner*. http://www.golem.de/0705/52277.html, 2007. – letzter Zugriff: 24. Sep. 2007

[Ima] *Imaging Source, Firewire Color Camera*. http://www.theimagingsource.com/en/products/cameras/firewire_color/dfk31bf03/overview/,. – letzter Zugriff: 24. Sep. 2007

[KLK+02] KALKUSCH, Michael ; LIDY, Thomas ; KNAPP, Michael ; REITMAYR, Gerhard ; KAUFMANN, Hannes ; SCHMALSTIEG, Dieter: Structured Visual Markers for Indoor Pathfinding. In: *The First IEEE International Augmented Reality Toolkit Workshop 29 September 2002 Darmstadt Germany*, 2002

[Lin04] LINGEMANN, Kai: Schnelles Pose-Tracking auf Laserscan-Daten für autonome mobile Roboter. In: *Fraunhofer Series in Information and Communication Technology* Bd. 7. Shaker Verlag, 2004

[LR02] LANKENAU, Axel ; RÖFER, Thomas: Mobile Robot Self-Localization in Large-Scale Environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2002)*, 2002, S. 1359–1364

[ME85] MORAVEC, H.P. ; ELFES, A.: High Resolution Maps from Wide Angle Sonar. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '85)*, 1985, S. 116–121

[NLLP04] NI, Lionel M. ; LIU, Yunhao ; LAU, Yiu C. ; PATIL, Abhishek P.: LANDMARC: indoor location sensing using active RFID. In: *Wirel. Netw.* 10 (2004), Nr. 6, S. 701–710

[PDMP06] PELLENZ, Johannes ; DELIS, Christian ; MIHAILIDIS, Ioannis ; PAULUS, Dietrich: Low-Cost 3D-Laserscanner für mobile Systeme im RoboCup Rescue Wettbewerb. In: *9. Anwendungsbezogener Workshop zur Erfassung, Modellierung, Verarbeitung und Auswertung von 3D-Daten* (2006)

[Pro] *Proximity Sensing*. http://whatis.techtarget.com/definition/0,,sid9_gci523549,00.html,. – letzter Zugriff: 24. Sep. 2007

[RD06] REITMAYR, Gerhard ; DRUMMOND, Tom: Going Out: Robust Model-based Tracking for Outdoor Augmented Reality. In: *IEEE ISMAR'06*, 2006

[Reu03] REUELS, Christian: *Global Positioning System*. Universität Osnabrück : `www-lehre.inf.uos.de/mc/material/reuels.pdf`, 2003

[RO] ROHR, Jan F. ; ORLOWSKI, Maximilian: *Autonome Staubsauger*. `ots.fh-brandenburg.de/downloads/vacposter/Poster-09-2005.pdf`, . – Fachhochschule Brandenburg, letzter Zugriff: 24. Sep. 2007

[Ros06] ROSENTHAL, Jean-Claude: *Feature Detection und Matching Verfahren zur Position- und Lagebestimmung*, Universität Koblenz, Diplomarbeit, 2006

[SEGL05] SIM, Robert ; ELINAS, Pantelis ; GRIFFIN, Matt ; LITTEL, James J.: Vision-based SLAM using the Rao-Blackwellised Particle Filter. In: *IJCAI Workshop Reasoning with Uncertainty in Robotics*, 2005, S. 8

[SLL01] SE, S. ; LOWE, D. ; LITTLE, J.: Vision-based Mobile robot localization and mapping using scale-invariant features. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Seoul, Korea, May 2001, S. 2051–2058

[SLNH01] SURMANN, Hartmut ; LINGEMANN, Kai ; NÜCHTER, Andreas ; HERTZBERG, Joachim: *Aufbau eines 3D-Laserscanners für autonome mobile Roboter*. `https://www.uni-koblenz.de/~agas/Pool/Surmann2001AE3.pdf`, 2001

[SNH03] SURMANN, Hartmut ; NÜCHTER, Andreas ; HERTZBERG, Joachim: An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. In: *Robotics and Autonomous Systems* 45 (2003), Nr. 3-4, S. 181 – 198

[Spi] *Autonomous Car Project: "Spirit of Berlin"*. `http://www.spiritofberlin.eu`, . – Freie Universität Berlin, letzter Zugriff: 24. Sep. 2007

[Spi07] SPIEGEL ONLINE (Hrsg.): *Fahrerloses Auto: Forscher testen "Spirit of Berlin"*. `http://www.spiegel.de/auto/aktuell/0,1518,482854,00.html`, 2007. – letzter Zugriff: 24. Sep. 2007

[Str01] STRICKER, Didier: Tracking with reference images: a real-time and markerless tracking solution for out-door augmented reality applications. In: *Virtual Reality, Archeology, and Cultural Heritage (VAST)*, 2001, S. 77–82

[Tec07] *Technical Design Robbie*. 2007. – `Technische_Dokumente/Robbie/TecDesignRobbie.pdf`

[Thi] *Kommerzielle Laserscanner*. `http://www.thinglab.co.uk/compare.php?SubCatID=30`, . – letzter Zugriff: 24. Sep. 2007

[Tsa87] TSAI, Roger Y.: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. In: *IEEE Journal of Robotics and Automation* 3 (1987), Nr. 4, S. 323–344

[TV98] TRUCCO, E. ; VERRI, A.: *Introductory Techniques for 3-D Computer Vision*. New York : Prentice Hall, 1998

[URG05] *Scanning Laser Range Finder URG-04LX Specifications*. 2005. – `Technische_Dokumente/Laserscanner/URG-04LX_Spec.pdf`

[Wie04] WIENDL, Volker: *Robuste Markererkennung für Augmented Reality Anwendungen*. 2004. – Multimedia Praktikum im WS 2003/04 am Lehrstuhl für Multimediakonzepte und Anwendungen der Universität Augsburg

[ZON+06] ZHU, Zhiwei (Hrsg.) ; OSKIPER, Taragay (Hrsg.) ; NARODITSKY, Oleg (Hrsg.) ; SAMARASEKERA, Supun (Hrsg.) ; SAWHNEY, Harpreet S. (Hrsg.) ; KUMAR, Rakesh (Hrsg.): *An Improved Stereo-based Visual Odometry System*. 2006