Integration von 3D-„Time-of-Flight“-Kameras in Applikationen zur sicheren Steuerung autonomer Roboter

Vom Department Elektrotechnik und Informatik der Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Dipl.-Ing. Jens Bernshausen

1. Gutachter:	Prof. Dr.-Ing. Hubert Roth
2. Gutachter:	Prof. Dr.-Ing. Michael Weyrich
Vorsitzender:	Prof. Dr.-Ing. Dietmar Ehrhardt

Tag der mündlichen Prüfung: 14. Dezember 2011

Vorwort

Die vorliegende Arbeit entstand im Rahmen des BMBF-Projekts „Lynkeus“, während meiner Tätigkeit als wissenschaftlicher Mitarbeiter im Lehrstuhl Steuerungs- und Regelungstechnik an der Naturwissenschaftlichen - Technischen Fakultät der Universität Siegen.

Zunächst möchte ich meinem Erstgutachter und Vorsitzenden des Lehrstuhls Steuerungs- und Regelungstechnik Herrn Prof. Hubert Roth danken, für die Ermöglichung dieser Arbeit und der Unterstützung während meines Promotionsvorhabens. Ebenso möchte ich meinem Zweitgutachter Herrn Prof. Michael Weyrich für die freundliche Bereitschaft zur Übernahme des Zweitgutachtens danken.

Ferner möchte ich meinen besonderen Dank Herrn Dr. Jürgen Wahrburg aussprechen, der mir während meiner Arbeit jederzeit mit Rat und Tat zur Seite stand. Dank seiner fortwährenden Unterstützung und Bemühungen wurde die Fertigstellung dieser Arbeit erst möglich. Natürlich möchte ich auch meinen ehemaligen Kollegen Hans-Christian Schneider, Marc Schlimbach, Dominik Scarpin, Thomas Kerstein, Stephanie Sahm, Hamza Hassan Shah und Raul Armado Castillio Cruces danken, für ihre Hilfsbereitschaft und die guten fachlichen aber auch freundschaftlichen Ratschläge, die mir sehr oft weitergeholfen haben. Auch ein Dankeschön an meine Kollegen und Projektpartner innerhalb des „Lynkeus“-Projekts für die gute Zusammenarbeit.

Danken möchte ich auch meiner Familie, meinen Freunden und ganz besonders meiner Freundin Stefanie Rudolf, die nicht nur meine schlechte Laune und Zeiten der überstrapazierten Nerven ertragen und meine Sorgen geteilt haben, sondern mich auch in jeder Hinsicht Tatkräftig unterstützt haben.

Abstract

Meanwhile laserscanner and stereovision systems have become standard measuring sensors in the field of mobile robotics. They are used for observing the driveway and obstacle recognition. In the field of handling robots these sensors are included to measure, to check or to detect objects for grasping and processing operations. Because of the continual development of the PMD camera and the associated improvement of the measurement quality, the PMD camera offers a good alternative to the mentioned measuring sensors, meanwhile.

This work is occupied in developing different applications for a safety control of robots using the PMD camera. Partially we will use known algorithm, which we want to adept for the use of PMD cameras and partially we have to develop new algorithms designed to the performance of the PMD cameras.

One main aspect in the field of mobile robotic is the integration of the PMD camera technology for driveway observation, to detect objects in front of the robot. Combined with a path planning algorithm, the navigation of the mobile robot system in a partially unknown environment should be allowed with the exclusive use of a PMD camera. Another aspect of the PMD based control of the mobile robot platform is the autonomous docking to a trailer. Therefore we have to implement methods for the recognition of the trailer and for planning and controlling the docking manoevre.

Beside the use of the camera in the area of the mobile robotics, in the second part of this work the camera will be used in the industrial area also for the observation of a robot cell. Humans and unknown objects inside of the robot cell will be detected with the PMD camera and the robot will be react at the objects by stopping the movement or planning an alternative collision free path.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AKF	**A**uto**k**orreations**f**unktion: Verfahren zur Bestimmung des Phasenunterschieds zwischen zwei gleichen Signalen
BwB	**B**all **w**ithin **B**ound – Test ist eine Methode zur Überprüfung ob ein Messwert innerhalb eines kreisförmigen Gebiets liegt
CAN	**C**ontroller **A**rea **N**etwork: asynchrones, serielles Feldbussystem
CCD	**C**harge-**C**oupled **D**evice: integriertes elektronisches Bauelement zum Transport von elektrischen Ladungen
CMOS	**C**omplementary **M**etal **O**xid **S**emiconducter: Verfahren zur Herstellung von komplementären Metalloxid Halbleitern
DLR	**D**eutsches **L**uft- und **R**aumfahrtzentrum
FTS	**F**ahrerloses **T**ransport**s**ystem: autonome mobile Roboterplattform zum Transport von Objekten in der Industrie
ICP	**I**terative **C**losest **P**oint - Algorithmus: iterativer Algorithmus zur Näherung von überlappenden Punktwolken
KKF	**K**reuz**k**orrelations**f**unktion: Verfahren zur Ermittlung der Ähnlichkeit von zwei Signalen
LMS	**L**east-**M**ean-**S**quare: Algorithmus zur Minimierung des kleinsten Fehlerquadrates
PMD	**P**hotonen**m**isch**d**etektor: Ein auf CMOS-Technik basierender dreidimensional messender Sensor
QT	C++ Klassenbibliothek zur Programmierung grafischer Benutzeroberflächen

RANSAC	**Ran**dom **Sa**mple **C**onsensus: Methode zur Schätzung eines Modells anhand von Messwerten durch eine zufällige Stichprobe
SBI	**S**uppression of **B**acklight **I**llumination: integrierte Schaltung zur Unterdrückung von Hintergrundlicht
SVD	**S**ingular **V**alue **D**ecomposition: Singularitätswertzerlegung bezeichnet die Zerlegung einer Matrix als Produkt von drei speziellen Matrizen
TCP	**T**ool **C**enter **P**oint: Koordinatensystemursprung des Werkzeugs von Industrierobotern
ToF	**T**ime-**o**f-**F**light: Verfahren zur Messung der Lichtlaufzeit
TrICP	**Tr**immed **I**terative **C**losest **P**oint: angepasster ICP-Algorithmus
LIFO	**L**ast **I**n **F**irst **O**ut: Methode zur Exploration von topologischen Graphen
KRL	**K**UKA **R**obot **L**anguage: Programmiersprache zur Programmierung von KUKA Robotersystemen
XML	E**x**tensible **M**arkup **L**anguage: Sprache zur Darstellung von Daten in einer hierarisch strukturierten Darstellung

1 Einleitung

Die vorliegende Arbeit beschäftigt sich mit der Integration von PMD-Kameras in Applikationen zur sicherheitsbasierten Steuerung verschiedener Robotersysteme. Die PMD-Kamera (Photonenmischdetektor) ist eine auf dem „Time-of-Flight“ (ToF)-Verfahren basierende dreidimensional messende Kamera. Durch das Aussenden von moduliertem Licht im infrarotem Spektrum kann, mittels eines speziell entwickelten Sensors, durch Korrelation des reflektierten und vom Sensor empfangenen Lichtsignals, mit einem gleich getaktetem Referenzsignal, für jeden Pixel des PMD-Arrays ein Entfernungswert ermittelt werden. Neben dem Entfernungsbild wird zudem ein Intensitätsbild der Szene aufgenommen. Somit kann das Entfernungsbild mit einer Textur belegt werden, welches eine zweidimensionale, sowie dreidimensionale Bilddatenverarbeitung ermöglicht. Verglichen mit alternativen Entfernungsmess-systemen bietet die PMD-Kamera viele Vorteile. Es kann zum Beispiel auf eine softwareseitige Vorverarbeitung zur Generierung von Entfernungsbildern, wie dies bei Stereovisionsystemen der Fall ist, verzichtet werden. Zusätzlich bietet die PMD-Technik eine hohe Bildwiederholfrequenz und gute Integrierbarkeit aufgrund ihrer kompakten Bauweise. Zudem ist sie kostengünstig und robust aufgrund fehlender mechanisch bewegter Bauteile. Nachteilig sind die verhältnismäßig geringe Entfernungsauflösung und der im Vergleich zu Laserscannern geringe Sichtbereich. Im Zuge dieser Arbeit wird auf die Eigenschaften der PMD-Technik, im Vergleich zu Laserscannern und Stereovisionsystemen, detaillierter eingegangen.

Laserscanner und Stereovisionsysteme gehören mittlerweile im Bereich der mobilen Robotik zu den Standardmesssensoren und werden zur Fahrwegüberwachung sowie zur Hinderniserkennung und -vermeidung eingesetzt. In der Handhabungsrobotik werden diese Sensoren zur Vermessung, Überprüfung und Detektion von zu greifenden oder von zu bearbeitenden Objekten verwendet. Aufgrund der stetigen Weiter-entwicklung der PMD-Sensorik und der damit einhergehenden Verbesserung der Messeigenschaften der PMD-Kamera, bietet diese zunehmend eine hervorragende Alternative zu den erwähnten Sensoren.

Im Zuge dieser Arbeit sollen verschiedene Applikationen zur sicheren Steuerung von Robotern mit der PMD-Kamera realisiert werden. Die Applikationen sind im Bereich der

mobilen Robotik als auch im Bereich der Handhabungsrobotik anzusiedeln. Hierfür wird zum Teil auf existierende Algorithmen zurückgegriffen, welche für die Verwendung der PMD-Kamera angepasst werden, als auch neue auf die PMD-Technik zugeschnittene Algorithmen entwickelt. Ein Hauptaspekt ist die Implementierung von Methoden zur Beobachtung des Roboterfahrwegs. Objekte sollen mit der PMD-Kamera detektiert und so eventuelle Kollisionen mit dem Robotersystem erkannt und vermieden werden. Kombiniert mit einem reaktiven Online-Bahnplanungsverfahren soll es dem mobilen Roboter ermöglicht werden, unter alleinigem Einsatz der PMD-Kamera, autonom in einer nur teilweise bekannten Umgebung zu navigieren. Ein weiterer Aspekt der autonomen Navigation von fahrerlosen Transportsystemen ist die Notwendigkeit Anhänger autonom an- beziehungsweise abzukoppeln. Dies stellt ein, bei fahrerlosen Transportsystemen zurzeit noch nicht vollständig gelöstes, Problem dar. Die Einbindung der PMD-Kamera verspricht die Möglichkeit, Methoden zur Erkennung von Anhängerdeichseln und darauf basierend Verfahren zur Andockplanung und -Regelung zu entwerfen.

Neben der mobilen Robotik stellt die Handhabungsrobotik ein interessantes Anwendungsgebiet für die PMD-Technologie dar. Daher wird im zweiten Teil dieser Arbeit die Überwachung einer Roboterzelle mit einer PMD-Kamera realisiert. Durch die dreidimensionale Aufnahme des Roboterumfelds, sollen Fremdobjekte erkannt und Verfahren zur Kollisionsvermeidung implementiert werden.

Beide angesprochene Themenbereiche waren und sind immer noch Gegenstand von zahlreichen Forschungsprojekten. Bisher wurden jedoch Laserscanner oder Stereo-visionsysteme zur Beobachtung des Roboterfahrwegs beziehungsweise des Roboters, innerhalb der Handhabungsrobotik, eingesetzt. Diese Sensoren haben den Nachteil, dass diese in den beschriebenen Anwendungsgebieten mit weiteren Sensoren fusioniert werden müssen. Einige Projekte befassen sich mit der Verwendung von PMD-Kameras in Teilbereichen, jedoch nicht als alleiniges optisches Sensorsystem. Die Innovation dieser Arbeit im Vergleich zu früheren Arbeiten liegt in der Entwicklung von Methoden zur Steuerung und Beobachtung von Robotern, welche die Verwendung von PMD-Kameras als einziges optisches Sensorsystem ermöglichen.

Die Arbeit ist in die Abschnitte Grundlagen der PMD-Kamera, Integration der PMD-Kamera zur sicheren Steuerung von mobilen Robotern und Integration der PMD-Kamera zur sicheren Steuerung von Handhabungsrobotern gegliedert.

Zunächst wird in Abschnitt 2 die Zielsetzung dieser Arbeit detailliert beschrieben. In Kapitel 3 wird anschließend der aktuelle Stand der Technik dargestellt. Hierzu wird ein

Einblick in die Sensordatenverarbeitung innerhalb der Industrie- und mobilen Robotik gegeben.

Kapitel 4 gibt zunächst einen Überblick über das Funktionsprinzip der PMD-Kamera. Im Anschluss werden die verschiedenen zurzeit erhältlichen Systeme vorgestellt, sowie die Vor- und Nachteile der PMD-Sensorik im Vergleich zu den herkömmlichen, optischen Messsystemen, wie Laserscanner und Stereovisionsystemen, bezogen auf die Zielsetzung dieser Arbeit, herausgestellt.

In Kapitel 5 wird die PMD-Kamera zur Entwicklung von Methoden zur sicheren autonomen Steuerung von mobilen Robotern eingesetzt. Geeignete Algorithmen werden entwickelt und analysiert, welche ein autonomes Navigieren des mobilen Roboters in einer nur teilweise bekannten Umgebung ermöglichen. Kapitel 5 beschreibt den Versuchsaufbau, die Registrierung der PMD-Kamera im Roboterkoordinatensystem und die Methoden zur Selbstlokalisierung des Roboters. Darauf basierend werden Bahnplanungsalgorithmen und Regelmechanismen zum Abfahren von geplanten Trajektorien implementiert.

Die Einbindung der PMD-Kamera in Applikationen zum sicheren Betrieb von Handhabungsrobotern erfolgt in Kapitel 6. Die Anbringung der PMD-Kamera oberhalb des Robotersystems ermöglicht die Implementierung von Methoden zur Überwachung des Roboterarbeitsraums. Algorithmen zur Erkennung von Personen und nicht registrierten Objekten und somit zur Berechnung von deren Lage bezüglich des Roboters werden entwickelt. Dies ermöglicht den gefahrlosen Betrieb von Robotern ohne die Installation von Sicherheitszäunen. Im zweiten Teil von Kapitel 6 wird diese Funktionalität um ein Bahnplanungsverfahren erweitert, welches die Möglichkeit eröffnet alternative Trajektorien zu planen, sollte eine Kollision auf der gegebenen Trajektorie erkannt werden.

Kapitel 7 stellt eine Zusammenfassung der in Kapitel 5 und 6 gewonnenen Erkenntnisse dar. Hier werden zunächst die Ergebnisse zusammengefasst und diskutiert und sich kritisch mit diesen auseinandergesetzt. Abschließend erfolgt ein kurzer Ausblick zu möglichen Verbesserungen der entwickelten Systeme.

2 Zielsetzung

Ziel dieser Arbeit ist das Einbinden einer oder mehrerer 3D-PMD-Kameras, in Applikationen der mobilen Robotik sowie Handhabungsrobotik, zur Realisierung eines sicheren autonomen Betriebs des jeweiligen Robotersystems. Zur Umsetzung dieses Vorhabens müssen neue Verfahren entwickelt werden, die eine Überwachung des Roboterumfelds mittels PMD-Kamera ermöglichen. Bisher gibt es nur wenige Vorarbeiten die sich mit der Verwendung der PMD-Technik, in den beschriebenen Bereichen, beschäftigen. Bekannte Arbeiten, welche die PMD-Technik verwenden, nutzen diese meist als Ergänzung zu herkömmlichen Messverfahren oder nur in Teilbereichen der Robotersteuerung. In dieser Arbeit soll die PMD-Kamera als einziges optisches Messsystem dienen. Somit ist die Entwicklung neuartiger Methoden zur Steuerung der Robotersysteme, welche speziell auf die Erfordernisse der Verwendung von 3D PMD-Bilddaten angepasst werden, notwendig.

Die Arbeit kann in zwei Hauptbereiche unterteilt werden. Diese sind die mobile Robotik sowie die Handhabungsrobotik. Zunächst soll die PMD-Kamera für eine autonome Navigation eines mobilen Roboters verwendet werden. Hierbei sollen nicht nur Hindernisse detektiert oder Umgebungskarten aufgenommen werden, sondern die PMD-Kamera in allen Funktionalitäten des Robotersystems Verwendung finden. Hierzu gehören Algorithmen zur Selbstlokalisierung. Es wird unterschieden zwischen relativer und absoluter Selbstlokalisierung. Die relative Poseschätzung, also die Ermittlung der Roboterpose bezogen auf einen Bezugspunkt, soll durch die Berechnung der Verschiebungen zwischen aufeinander folgenden PMD-Aufnahmen geschehen. Neben der relativen Poseermittlung soll eine absolute Lokalisierung, über die Erkennung von künstlichen Landmarken, deren Positionen innerhalb der vorhandenen Umgebungskarte bekannt sind, stattfinden. Durch die Fusion der beiden Methoden wird versucht eine sehr präzise Selbstlokalisierung zu erreichen. Ebenfalls gilt es eine Methode zur schnellen Detektion von Hindernissen auf Basis der PMD-Daten zu entwickeln. Die 3D-Daten müssen hierzu in einer Art vorverarbeitet werden, so dass sich die Hindernisse in Echtzeit aus den Kamerabildern extrahieren lassen. Detektierte Hindernisse, beziehungsweise Objekte müssen mit den in der Karte eingetragenen Hindernissen abgeglichen werden. Sollten neue Hindernisse erkannt werden, müssen diese in die Umgebungskarte eingetragen und auf Kollision überprüft werden. Falls bevorstehende

Kollisionen erkannt werden, muss die Trajektorie des Roboters basierend auf den Kameradaten neu geplant werden. Ferner gilt es ein Bahnplanungsverfahren zu entwickeln, welches die Möglichkeit einer echtzeitfähigen Bahnplanung bietet, so dass der Roboter ohne Verzögerungen weiterfahren kann.

Ein weiterer Forschungsschwerpunkt bezüglich der mobilen Robotik ist das rückwärts Andocken eines fahrerlosen Transportsystems an eine Anhängerdeichsel. In der Industrie müssen autonome Transportsysteme oft Anhänger an- und abkoppeln. Dies wird zurzeit manuell durchgeführt, da keine zufriedenstellende autonome Lösung existiert. Aus diesem Grund soll, zusätzlich zur PMD basierten Navigation zwischen zwei vorgegebenen Positionen, das selbstständige Andocken eines autonomen Fahrzeugs an eine Anhängerdeichsel implementiert werden. Im Zuge dieser Arbeit werden Algorithmen entworfen, die eine rechtzeitige Erkennung und Positionsbestimmung der Deichsel bezüglich des fahrerlosen Transportsystems mittels PMD-Kamera erlauben. Mit einem nachgeschalteten Bahnplanungs- und Regelungsalgorithmus soll ein präzises Andocken des Robotersystems an die Deichsel gewährleistet werden.

Des Weiteren soll die PMD-Kamera zur Überwachung von Roboterarbeitsräumen im Bereich der Handhabungsrobotik eingesetzt werden. Das Robotersystem soll in die Lage versetzt werden, selbstständig auftretende Hindernisse zu erkennen und auf diese entsprechend zu reagieren. Dazu werden im zweiten Hauptabschnitt der Arbeit die entwickelten Algorithmen für die mobile Robotik soweit möglich für die Verwendung eines Handhabungsroboters erweitert und angepasst. Für Teilbereiche in denen keine Adaption möglich ist, müssen neue Methoden entwickelt werden. In den Arbeiten von [Fischer09] wird eine Hinderniserkennung für Handhabungsroboter basierend auf PMD-Daten behandelt. Dieses System bietet jedoch nicht die Möglichkeit auf Basis der PMD-Daten durch Kollisionserkennung und Trajektorienplanung auf das auftretende Hindernis zu reagieren. Ziel dieser Arbeit ist jedoch, basierend auf der Erkennung von Fremdobjekten, dies können Personen wie auch in der Roboterzelle befindliche Objekte sein, ein System zu entwerfen, welches die Möglichkeit bietet reaktiv auf Hindernisse mittels einer Kollisionserkennung und anschließender Bahnplanung zu reagieren. Ebenso wie in der mobilen Robotik werden Verfahren zur Kameraregistrierung im Roboterkoordinatensystem, zur Lokalisierung des Roboters in den PMD-Aufnahmen, zur Erkennung von Hindernissen und zur reaktiven Bahnplanung entwickelt.

3 Stand der Technik

Das Einbinden von Sensoren zur Umgebungserfassung ist speziell in der mobilen Robotik ein weit verbreitetes Prinzip. Schon zu Beginn der 1950er Jahre beschrieb Grey Walter in [Walter51] die Entwicklung von zwei mobilen Robotern, welche auf dem Prinzip der Phototaxie das autonome Ausweichen von Hindernissen erlernen konnten. Der erste Roboter, welcher seine Umgebung selbstständig wahrnehmen konnte, wurde 1969 am Stanford Research Institute entwickelt. Der Roboter war mit visuellen Entfernungssensoren, einer Kamera und mehreren Berührungssensoren ausgestattet. Somit konnte dieser Hindernissen ausweichen und Objektmanipulationen durchführen. Dies war jedoch nur in einer einfachen und strukturierten Umgebung für spezielle Hindernisse und Objekte möglich. Ebenfalls in Stanford wurde der mobile Roboter CART [Nehmzow02] entwickelt. Dieser war mit einem Kamerasensor ausgestattet und konnte durch die Digitalisierung von Kamerabildern seine dreidimensionale Umgebung erfassen. Die Verarbeitung der Daten forderte jedoch einen erheblichen Zeitaufwand. Um die Bilder für eine Roboterposition zu verarbeiten, wurden mehr als 15 Minuten benötigt, bevor der Roboter um je einen Meter weiterfuhr und neue Bilder aufnahm. Das erste europäische Projekt HILARE [Nehmzow02] folgte Ende der 1970er Jahre. HILARE nutzte Bildverarbeitungsalgorithmen und konnte durch die Kombination von Ultraschallsensoren und Laserentfernungsmesser selbstständig navigieren. Zur Erkennung von Hindernissen wurden die Ultraschallsensoren verwendet. Die Darstellung der Umgebung erfolgte durch eine zweidimensionale polygonale Interpretation des Raums.

Im Zuge der fortschreitenden Entwicklung von mobilen Robotern und der entsprechenden Sensorik zur Umwelterfassung, erhielten diese Einzug in zahlreiche unterschiedliche Anwendungsgebiete. Schnell wurden mobile Roboter eingesetzt als fahrerlose Transportsysteme zum Befördern von Materialien im industriellen Umfeld. Dabei erfolgt die Navigation der Roboterplattformen üblicherweise anhand von in den Boden eingelassener Induktionsschleifen, Transpondern oder durch Fahrbahnmarkierungen. Die Erkennung von Hindernissen geschieht zumeist über die Verwendung von Laserscannern oder 3D-Stereovisionsystemen. Während fahrerlose Transportsysteme in der Industrie meistens nach dem Erkennen von Hindernissen durch Stoppen reagieren, reagieren mobile Roboter im Bereich der Servicerobotik

durch das Planen einer alternativen Trajektorie zum Zielpunkt. Anhand von Entfernungsmesssystemen, wie Laserscanner und Stereovisionsysteme, wird die Umgebung des Roboters detektiert und die Informationen über diese zur Planung kollisionsfreier Routen zwischen vorgegebener Start- und Zielpose verwendet [Brauer07]. Ein weiteres Forschungsfeld der mobilen Robotik ist zurzeit die Generierung von 3D-Umgebungskarten bei gleichzeitiger Lokalisierung des Roboters in unbekannten Umgebungen. In [Cole06] werden hierfür 2D-Laserscanner eingesetzt. Während ein Laserscanner zur Hinderniserkennung dient, ist ein zweiter auf einer Schwenkeinrichtung, zur Aufnahme einer dreidimensionalen Umgebungskarte, befestigt. Um die Karte mit einer Textur zu belegen, dient eine herkömmliche Digitalkamera die ebenfalls rotatorisch bewegt werden kann. In [Garcia04] wird als Alternative zu Laserscannern ein Stereovisionsystem zur Generierung einer 3D-Umgebungskarte verwendet.

Parallel zur Entwicklung von mobilen Robotern wurden Industrieroboter, mit dem Ziel dem Menschen unangenehme, monotone und gefährliche Arbeiten abzunehmen, entwickelt. 1951 begann die Entwicklung von teleoperativ gesteuerten Manipulatoren zur Handhabung radioaktiver Materialien. 1959 wurde das erste kommerziell erhältlichen Robotersystem vorgestellt. Die Steuerung dieses Roboters erfolgte anhand von Kurvenscheiben und Begrenzungsschaltern. Im Jahr 1960 wurde der erste Industrieroboter „Unimate“ vorgestellt. Dieser, basierend auf den Arbeiten von [Devol], wurde hydraulisch bewegt und durch einen Computer mittels numerischer Steuerungsalgorithmen kontrolliert. Ein weiterer Meilenstein war die Präsentation des Roboters „PUMA“ im Jahr 1978, welcher einer der ersten elektrisch angetriebenen Manipulatoren war.

Eine Anwendung die seit den 1980er Jahren Gegenstand von aktuellen Forschungsprojekten ist, ist das sogenannte „bin-picking“ (Griff in die Kiste). Dies entspricht der Erkennung und Entnahme von ungeordneten Teilen aus einem Behälter mit der Hilfe von Handhabungsrobotern. Hierbei werden 3D-Messsysteme, wie Stereovisionsysteme oder Laserscanner, die an dem Roboter angebracht werden, zur Aufnahme des Behälters und der hierin befindlichen Teile, verwendet. Mit speziellen Algorithmen ist es anhand der 3D-Daten möglich, eine Trajektorie zum Greifen der ungeordneten Teile zu planen. Dieses Problem ist jedoch noch immer nicht allgemeingültig gelöst. In [Boug03] wird ein Laserscanner in Kombination mit einer Farbkamera zur Erkennung und Greifplanung von unbekannten Objekten verwendet.

Neben der Erkennung von Objekten zur sensorbasierten Greifplanung spielt die Überwachung der Roboterarbeitsräume mit optischen Entfernungsmessverfahren eine zunehmend stärkere Rolle. Aktuell müssen Roboterarbeitsräume durch Sicherheitszäune gegen unbefugtes Betreten während des Roboterbetriebs gesichert werden. Dies ist jedoch eine unflexible und platzeinnehmende Methode. Änderungen der Roboterzelle sind mit einem hohen Zeit- und Kostenaufwand verbunden. Jedoch gibt es mittlerweile einige Bestrebungen den Robotersystemen durch den Einsatz von Bild- und Entfernungsmessverfahren einen höheren Grad der Autonomie zu verleihen, so dass Fremdobjekte innerhalb der Roboterzelle erkannt werden können und der Roboter entsprechend reagieren kann. So beschäftigte sich Stettmer in seiner Arbeit „Sensorgestützte Kollisionsvermeidung bei Industrierobotern“ [Stettmer94] mit der Verwendung von Laserscannern zur Überwachung der Roboterzelle. Durch eine beschleunigte Signalverarbeitung und dem parallelen Berechnen von Ausweichroutinen wurde ein sensorbasiertes Sicherheitssystem implementiert, welches den gefahrlosen Betrieb eines Industrieroboters ermöglicht. In [Ebert02] wird ein Mehrkamerasystem dazu verwendet, den Roboterarbeitsraum zu überwachen. Hierdurch ist es möglich Kollisionen zwischen dem Robotersystem und Personen, welche sich im Roboterarbeitsraum befinden, zu vermeiden.

Wie aus den beschriebenen Anwendungen ersichtlich ist, werden zumeist Laserscanner oder Stereovisionsysteme als Messinstrumente zur Umgebungswahrnehmung verwendet. Ein bisher selten bis gar nicht verwendeter Sensor zur Umgebungsvermessung im Bereich der Robotik ist der PMD-Sensor. Dieser fand seinen Ursprung Mitte der 1990er Jahre an der Universität Siegen. Der PMD-Sensor funktioniert über ein Time-of-Flight Verfahren und ermöglicht eine hardwarebasierte Berechnung eines Entfernungswertes für ein Pixelarray von bis zu 200x200 Pixeln. Für detaillierte Informationen sei auf Kapitel 4 verwiesen. Mittlerweile gibt es einige Ansätze zur Verwendung des PMD-Sensors in Robotikapplikationen.

Im Bereich der mobilen Robotik gibt es verschiedene Arbeiten, die sich mit 3D-SLAM, der simultanen Lokalisierung und Kartengenerierung, unter Verwendung der PMD-Technik beschäftigen. Zu erwähnen sind hier die Arbeiten von [Joochim08]. Hier wird die geringe laterale Auflösung einer PMD-Kamera durch Fusion mit einer VGA-Kamera erhöht, so dass eine hochauflösende 3D-Karte der Roboterumgebung generiert werden kann.

Ebenso wie in der mobilen Robotik, gibt es in der Handhabungsrobotik erste Versuche zur Verwendung der PMD-Technik. So wird in [Kyrill08] und [Fuchs09] versucht den

PMD-Sensor für das geschilderte Problem, der Aufnahme von unsortierten Teilen aus einer Kiste, zu benutzen. Hierbei ist die Kamera am Endeffektor des Roboters angebracht. Anhand eines gegebenen Objektmodells wird versucht Werkstücke innerhalb des PMD-Bilds zu erkennen und deren Position zu ermitteln. Aufgrund der geringen lateralen Auflösung lassen sich jedoch nur verhältnismäßig große Objekte für eine Greifplanung präzise genug lokalisieren. [Fischer09] zeigt einen ersten Ansatz zur Kollisions- und Hinderniserkennung eines Industrieroboters mit PMD-Kameras. Hier werden, durch Abgleich mit einem CAD-Umgebungsmodell, Fremdobjekte innerhalb der Roboterzelle detektiert. Auf erkannte Objekte kann der Roboter mit dem Anhalten der Trajektorienausführung reagieren.

4 Grundlagen der PMD-Kameratechnik

Das Funktionsprinzip von PMD-Kameras ist Gegenstand zahlreicher Veröffentlichungen. Für eine detaillierte Beschreibung der Funktionsweise sei auf die Arbeiten von [Frey08, Ringbeck01] verwiesen, welche die PMD-Technik eingehend behandeln. Im Folgenden soll ein Überblick über die PMD-Technik gegeben werden, so dass im Weiteren die Problemstellungen und Lösungsansätze im Gebrauch der PMD-Kamera vom Leser gut nachvollzogen und verstanden werden können. Sind die Messprinzipien sowie die Messeigenschaften von PMD-Kameras erläutert, werden diese mit Laserscannern und Stereovisionsystemen, bezüglich deren Eignung, bezogen auf die in Kapitel 2 definierten Ziele, verglichen.

4.1 Aufbau und Funktionsprinzip

Die PMD-Kamera stellt eine neue Generation von „Time-of-Flight“ (ToF)-Sensoren dar. Unter Verwendung der PMD-Technik ist das parallele, pixelweise Vermessen einer Szene ohne Scanvorgang, wie dies mitunter bei Laserscannern der Fall ist, möglich. Dies führt zu einer hohen Bildrate und einer hohen lateralen Auflösung. Zusätzlich zu den Distanzwerten können ebenfalls Intensitätswerte einer Szene aufgenommen werden.

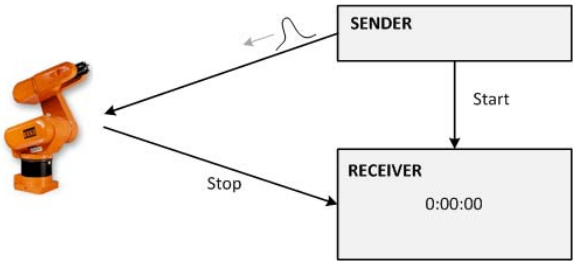


Abbildung 4.1.1: "Time-of-Flight" –Prinzipdarstellung

Abbildung 4.1.1 zeigt das Prinzip eines herkömmlichen ToF-Verfahrens. Hierbei wird ein Lichtimpuls emittiert, welcher von einem Ziel reflektiert und von einem Empfänger detektiert wird. Die Laufzeit des Lichts, die der Impuls vom Zeitpunkt des Sendens bis zum Empfangen benötigt, wird gemessen. Bei bekannter Lichtgeschwindigkeit c kann auf den zurückgelegten Weg s des Lichts und somit auf die Entfernung d des Objekts geschlossen werden.

$$d = \frac{1}{2}s = \frac{1}{2}c \cdot t \tag{4.1}$$

Sollte eine Entfernungsauflösung im Millimeterbereich erforderlich sein, bedeutet dies eine Messgenauigkeit von wenigen Pikosekunden um die geforderte Entfernungsauflösung zu erreichen. Zur Realisierung eines solchen Systems muss jedes Element innerhalb dessen eine extrem hohe Bandbreite aufweisen.

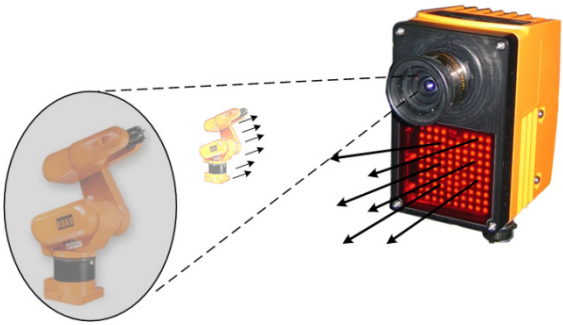


Abbildung 4.1.2: "Time-of-Flight" Messverfahren mit PMD-Technik

Im Gegensatz zu den herkömmlichen ToF-Verfahren basiert das Messverfahren der PMD-Kamera auf der Bestimmung des Phasenunterschieds zwischen dem reflektierten Licht einer kontinuierlich modulierten Lichtquelle und einem auf dem PMD-Sensor gleichartig moduliertem Referenzsignal. Die PMD-Kamera besitzt modellabhängig ein oder mehrere Beleuchtungsmodule, bestehend aus LED-Arrays, welche moduliertes Licht mit einer Wellenlänge im infraroten Bereich von ca. 850nm aussenden. Dieses wird von der Szene vor der Kamera reflektiert und durch eine Empfängeroptik auf den PMD-Chip projiziert. Der Vorteil des PMD-Verfahrens ist, dass jeder Pixel als Korrelationsempfänger zur Ermittlung des Phasenversatzes dient. Ein Distanzbild kann folglich mit einer hohen Geschwindigkeit aufgenommen werden.

4.2 Funktionsweise des PMD-Pixels

Abbildung 4.2.1 zeigt den Querschnitt eines auf CMOS-Technik basierenden PMD-Pixels. Das Pixel besteht aus zwei transparenten Modulationsgateelektroden, welche sich in der Mitte des Pixels befinden. Diese sind durch eine ebenfalls lichtdurchlässige Oxidschicht von dem p^+-dotierten Substrat elektrisch isoliert. Links und rechts von den Gateelektroden befindet sich je eine Auslesediode.

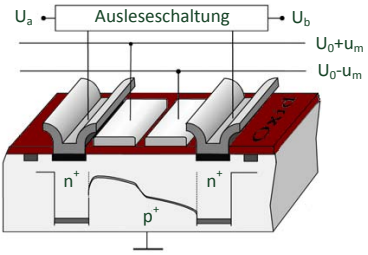


Abbildung 4.2.1: Querschnitt eines PMD-Pixels nach [Ringbeck07]

Dringt Licht durch die Gateelektroden in das Substrat, werden Ladungsträger aus dem Substrat gelöst [Ringbeck07]. Durch Anlegen einer Spannung an die Gateelektroden kann das Potential in dem Substrat so verändert werden, dass die generierten freien Ladungsträger in eine definierte Richtung abfließen und sich je nach Potential in einem der pn-Übergänge, welche die Auslesedioden bilden, sammeln.

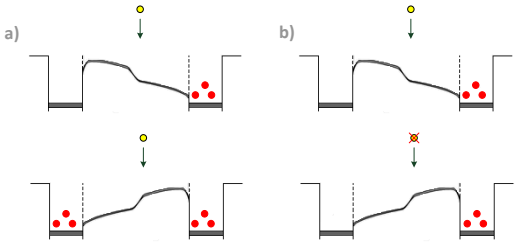


Abbildung 4.2.2: Erzeugung von Elektronen innerhalb des Substrats

Dieser Sachverhalt ist in Abbildung 4.2.2 für den Fall der Ansteuerung der Gateelektroden mit einer modulierten Spannung dargestellt. Die Gates werden im Gegentakt angesteuert. Es kommt zur Bildung eines Potentialgefälles im Substrat. Bei konstanter Beleuchtung, siehe Abbildung 4.2.2a, sammeln sich die Ladungsträger

gleichmäßig in den pn-Übergängen. Bei moduliertem Licht, welches von dem Pixel empfangen wird, ist es abhängig von der Phasenverschiebung zwischen Lichtsignal und elektrischem Modulationssignal, wie viele Ladungsträger sich innerhalb der Auslesedioden ansammeln. Abbildung 4.2.2b zeigt den Fall einer von Null unterschiedlichen Phasenverschiebung. Dies zeigt, dass die Spannung an den jeweiligen Auslesedioden abhängig von der Phasenverschiebung zwischen dem eintreffendem Licht und elektrischem Signal ist.

4.3 Messverfahren und Messgenauigkeit

Zur Messung des Phasenunterschieds zwischen elektrischem Referenzsignal und dem empfangenen Lichtsignal muss deren Autokorrelationsfunktion (AKF) berechnet werden. Diesbezüglich müssen nach [Schwarte98] vier einzelne Messungen A_1, A_2, A_3 und A_4 durchgeführt werden. Bei jeder Messung wird das Referenzsignal, mit dem die Photogates angesteuert werden, um 90° verschoben. Jede Messung entspricht folglich einem Abtastwert der AKF. Über die Gleichung:

$$\varphi = arctan\left(\frac{A_1 - A_3}{A_2 - A_4}\right) \tag{4.2}$$

kann die Phasenverschiebung φ ermittelt werden. Aufgrund des Messprinzips der PMD-Kamera können, neben der Entfernung der jeweiligen Objekte, sowohl Amplitude als auch Intensität der Szene aufgenommen werden. Die Amplitude a kann ermittelt werden durch:

$$a = \frac{\sqrt{(A_1 - A_3)^2 + (A_2 - A_4)^2}}{2} \tag{4.3}$$

Die Intensität beziehungsweise der Grauwert g berechnet sich zu:

$$g = \frac{1}{4}(A_1 + A_2 + A_3 + A_4) \tag{4.4}$$

Aus der Phasenverschiebung kann auf die Distanz d zum Objekt vor der Kamera geschlossen werden [Ringbeck01]:

$$d = \frac{c \cdot \varphi}{4\pi \cdot f_{mod}} \tag{4.5}$$

Die gemessene Entfernung ist abhängig von der Modulationsfrequenz f_{mod}. Bei einer Modulationsfrequenz von $f_{mod} = 20MHz$ beträgt die Wellenlänge $\lambda_{mod} = 15m$. Da das Licht den Weg zweimal zurücklegen muss, folgt für die maximal messbare Distanz:

$$d_{max} = \frac{\lambda_{mod}}{2} = 7{,}5m \tag{4.6}$$

Wichtiger Faktor zur Verwendung der PMD-Kamera in der Praxis ist die Messungenauigkeit der PMD-Kamera. Nach [Ringbeck01] kann bei Betrachtung folgender Gleichung eine Aussage über die Messungenauigkeit dR eines PMD-ToF-Systems getroffen werden:

$$dR = \frac{1}{\sqrt{N_{Phase}}} \cdot \frac{1}{k_{tot} \cdot \frac{S}{N}} \cdot \frac{\lambda_{mod}}{\sqrt{8}\pi} \tag{4.7}$$

Hierbei entspricht k_{tot} dem Modulationskontrast, S der Anzahl der von dem modulierten Licht generierten Elektronen und N der Anzahl der durch Rauschen und Hintergrundlicht sowie einiger Halbleitereffekte generierten Elektronen. Es ist ersichtlich, dass das Rauschen durch die Parameter S, N und λ_{mod} aktiv beeinflusst werden kann.

Durch die Reduzierung der Wellenlänge λ_{mod} des modulierten Lichtsignals wird die Messgenauigkeit erhöht, gleichzeitig vermindert sich jedoch die maximal messbare Entfernung. In der Praxis ist je nach Anwendungsgebiet ein Kompromiss zwischen Entfernung und Entfernungsauflösung zu finden. Die Wellenlänge kann jedoch nur soweit verringert, beziehungsweise die Modulationsfrequenz erhöht werden, wie es das Beleuchtungsmodul erlaubt.

Die Anzahl der durch das aktive Licht emittierten Elektronen kann durch eine höhere Integrationszeit t_{int} gesteigert werden, welches zur Reduzierung der Messungenauigkeit dR führt. Die Integrationzeit ist die Zeit, in welcher sich die Elektronen innerhalb der Auslesedioden ansammeln, also die Spannung aufintegriert wird. Dies ist äquivalent zu der Beleuchtungszeit einer herkömmlichen 2D-CCD Kamera. Allerdings können zu hohe Integrationszeiten zur Überbelichtung und somit zur Sättigung der PMD-Pixel führen. Dies resultiert in einer Verfälschung der zu messenden Distanzwerte. Die Integrationszeit ist folglich auf den Messbereich der Kamera und auf die Reflektivität der Szene anzupassen.

Zur Reduzierung der durch Fremdlicht und Rauschen generierten Elektronen, welche zu gleichen Teilen in den Auslesedioden gespeichert werden, besitzen die PMD-Pixel nach [Frey07] eine integrierte Schaltung zur Minimierung des Gleichanteils beider Auslesedioden. Auf diese Weise ist der PMD-Pixel, bis zu einer Lichtstärke von 8000lux [IFMO3D200], unempfindlich gegenüber Hintergrundlicht. Hierdurch kann die Messungenauigkeit, hervorgerufen durch Fremdlicht, maßgeblich reduziert werden. Dies ermöglicht den Betrieb der PMD-Kamera bei Anwendungen im Außenbereich unter Einstrahlung von Sonnenlicht. Nachteilig ist jedoch, dass die Aufnahme eines Grauwertes, durch das Löschen des Gleichanteils mit der Hintergrundlichtunterdrückung (SBI – Suppression of Backlight Illumination), nicht mehr möglich ist.

4.4 Verfügbare PMD-Kameramodelle

	3kS	19k	O3D-100	S3	Lyn-Cube	SR-3000	SR-4000
Eindeutigkeitsbereich	7,5m	7,5m	7,5m	7,5m	7,5m	5m	10m
Auflösung	64 x 48	160 x 120	64 x 50	64 x 48	200 x 200	176 x 144	176 x 144
Bildrate	< 15	< 15	< 20	< 20	< 20	< 25	< 50
Messgenauigkeit	> 6mm	> 6mm	> 5mm	> 5mm	k.A.	> 15mm	> 15mm
Öffnungswinkel	40°	40°	k.A.	30°(h) x 40°(v)	40°(h) x 40°(v)	44°(h) x 35°(v)	44°(h) x 35°(v)
SBI	ja	nein	ja	ja	ja	nein	Nein
Schnittstelle	Fire-Wire	Fire-Wire	Ethernet	Ethernet	USB	Ethernet	USB

Tabelle 4.1: Gegenüberstellung von ToF-Kameras

Tabelle 4.1 gibt einen Überblick über die, während der Projektlaufzeit erhältlichen, mit dem ToF-Verfahren arbeitenden, 3D-Kameramodelle und stellt die jeweiligen Eigenschaften der verschiedenen Systeme gegenüber. Die verschiedenen Modelle unterscheiden sich in ihrer lateralen Auflösung, Bildrate, Messgenauigkeit und im Objektivöffnungswinkel.

Zu Beginn dieser Arbeit waren von der Firma PMDTec lediglich die Modelle 3kS und 19k der [PMDVision]-Baureihe erhältlich.

Abbildung 4.4.1: [PMDVision] 3kS mit einer Auflösung von 64 x 48 Pixeln

Abbildung 4.4.1 zeigt die *[PMDVision]3kS*. Diese weist eine laterale Auflösung von 64x48 Pixeln auf. Das Objektiv besitzt einen Öffnungswinkel von 40°. Messungen zeigten jedoch, dass der effektive Öffnungswinkel nur 12(h) x 10(v) beträgt. Die Messauflösung der Kamera ist abhängig von der Entfernung und den Reflexionseigenschaften des zu messenden Objekts. Bei optimalen Bedingungen ist laut [PMD3kS_05] eine minimale Messgenauigkeit von 6mm zu erreichen. Diese kann in der Realität jedoch stark abweichen.

Die *[PMDVision]19k* besitzt eine laterale Auflösung von 160x120 Pixeln. Die höhere Auflösung ermöglicht eine bessere zweidimensionale Bildverarbeitung mit den von der Kamera generierten Grauwertinformationen. Das Defizit dieser Kamera ist die fehlende, sensorintegrierte Schaltung zur Hintergrundlichtunterdrückung (SBI - Suppression of Backlight Illumination). Hierdurch ist die Kamera den Einflüssen von Hintergrundlicht unterlegen, welches zur Erhöhung des Rauschens der Entfernungswerte führt.

Aus diesem Grund wurde anfangs die 3kS verwendet, da diese eine Schaltung zur Hintergrundlichtunterdrückung besitzt. Im weiteren Zuge der Entwicklung neuer Kameramodelle wurde jedoch zu diesen übergegangen.

Abbildung 4.4.2: O3D-100 mit einer Auflösung von 64x50 Pixeln

Die *O3D-100* ist eine mit dem PMD-Sensor ausgestattete Kamera, welche von der Firma ifm electronics entwickelt wurde. Die Kamera hat eine laterale Auflösung von 64x50 Pixeln. Laut Tabelle 4.1 weist sie einen Eindeutigkeitsbereich von 7,5m auf. Der maximale Messbereich beträgt, aufgrund des kleinen aktiven Beleuchtungsmoduls und der damit verbundenen geringen Lichtleistung, abhängig von der Reflektivität der

Szene zwischen 2,5m und 3,5m. Aufgrund der kompakten Bauweise, siehe Tabelle 4.1, ist diese jedoch gut in beliebige Anwendungen integrierbar. Die maximal erreichbare Bildfrequenz beträgt 20fps.

a) b)

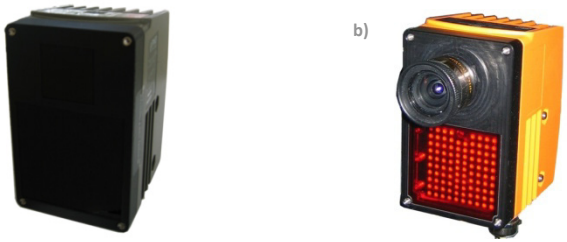


Abbildung 4.4.3: TOF-Kameras a) [PMDVision] S3 mit einer Auflösung von 64x48 Pixeln b) LynCube mit einer Auflösung von 200x200 Pixeln

Die *[PMDVision]S3* besitzt ebenfalls eine laterale Auflösung von 64x48 Pixeln bei einem Öffnungswinkel von 30°(h)x40°(v) und hat aufgrund des weiterentwickelten Sensors und der höheren optischen Leistung eine höhere Entfernungsauflösung. Die Genauigkeit der Entfernungsmessung ist auch hier abhängig von der Entfernung und der Reflektivität des Objekts. Diese liegt für ein Objekt mit einer Reflektivität von 90% und einer Entfernung bis zu $4m$ unterhalb von $\pm 5mm$. Der Messbereich der Kamera beträgt maximal 7,5m und die Bildrate bis zu 20fps. Im Verlauf dieser Arbeit wurde die [PMDVision]3kS, wegen des geringen Sichtfelds, durch die [PMDVision]S3 ersetzt.

Die *LynCube*, Abbildung 4.4.3b, ist ein Prototyp, welcher im Rahmen des Lynkeus-Projekts entwickelt und von der Firma ifm electronics, im Rahmen des Lynkeus-Projektes, zur Verfügung gestellt wurde. Die Kamera besitzt mit 200x200 Pixeln, im Bereich der Time-of-Flight Kameras, die höchste erhältliche laterale Auflösung. Das Objektiv besitzt einen Öffnungswinkel von 40°(h)x40°(v). Allerdings weist dieses Modell, durch eine nicht an das Objektiv und den Sensor angepassten Beleuchtung, eine sehr inhomogene Ausleuchtung des Sichtbereichs auf, welches sich auf die Qualität der 3D-Messwerte negativ auswirkt. Die Bildrate der LynCube liegt bei maximal 20fps. Die Kamera wurde im Rahmen dieser Arbeit zur Umgebungsüberwachung im Bereich der mobilen Robotik verwendet.

Der Vollständigkeit halber seien die Produkte der Firma Mesa Imaging erwähnt. Diese vertreiben ebenfalls ToF-Kameras, welche in dieser Arbeit jedoch keine Verwendung finden.

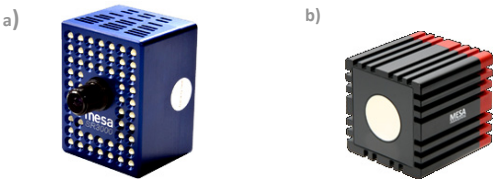


Abbildung 4.4.4: TOF-Kameras der Firma Mesa Imaging a) SR-3000 b) SR-4000 [MESA10]

Abbildung 4.4.4a zeigt eine *SwissRanger-SR3000* der Firma Mesa Imaging. Dieser Typ besitzt eine laterale Auflösung von 176x144 Pixeln und ist spezifiziert für einen Messbereich bis zu 5m. Die Kamera weist im Vergleich zu der [PMDVision]S3 oder der LynCube, resultierend aus der geringeren optischen Leistung und der höheren Verstärkung, ein wesentlich höheres Rauschverhalten auf. Nach [Rapp07] haben Tests gezeigt, dass die Anfälligkeit der Kamera gegen Hintergrundlicht vergleichbar mit der der [PMDVision]19k ist.

Abbildung 4.4.4b zeigt die *SwissRanger-SR4000*. Diese besitzt ebenfalls eine laterale Auflösung von 176x144 Pixeln. Die Kamera ist in mehreren Ausführungen erhältlich, mit Objektivöffnungswinkeln von 44°(h)x35°(v) und 69°(h)x56°(v), sowie spezifiziert für einen Messbereich von 5,0m oder von 10,0m. Bei einer 100%igen Reflektivität des Zielobjekts werden die Kameras mit einer Messgenauigkeit bis $\pm 10mm$ für das kleinere Objektiv und bis $\pm 15mm$ für das größere Objektiv angegeben [Mesa10]. Die aufgeführte maximale Framerate beträgt 54fps.

4.5 Vergleich mit alternativen 2D/3D-Messverfahren

Zur Objekt- und Hinderniserkennung sowie zur Selbstlokalisierung von mobilen Robotern werden ebenfalls, wie zur Überwachung von Roboterzellen im Bereich der Handhabungsrobotik, größtenteils Laserscanner und Stereovisionsysteme, zum Teil auch in Kombination, eingesetzt. Um die Vor- und Nachteile der PMD basierten Überwachung im Vergleich zu diesen Sensoren herauszustellen, werden im Folgenden zunächst die Funktionsweise und die Eigenschaften von Laserscannern und Stereovisionsystemen behandelt. Darauffolgend werden die jeweiligen Eigenschaften der vorgestellten Systeme, im Hinblick auf die Zielstellung dieser Arbeit, verglichen.

4.5.1 Laserscanner

Laserscanner können aufgrund ihres Messprinzips zwischen Pulslaufzeitmessverfahren und Phasenlaufzeitmessverfahren unterschieden werden. Bei Pulslaufzeitmessverfahren wird von einer Laserdiode ein kurzer Lichtimpuls ausgesendet. Dieser wird auf der Objektoberfläche diffus reflektiert, über eine Optik refokussiert und von einem Detektor empfangen. Über die Messung der Lichtlaufzeit kann die Entfernung berechnet werden.

$$d = \frac{c \cdot \Delta t}{2} \tag{4.8}$$

Durch die Ablenkung des Laserstrahls, mittels eines rotierenden Spiegels, können 2D-Entfernungswerte der Objektoberfläche aufgenommen werden. Daraus resultierend kann ein Profil der Oberfläche in Polarkoordinaten erstellt werden. Da die Entfernung von der Lichtgeschwindigkeit abgeleitet wird, bewegen sich die zu messenden Zeiten im Bereich von einigen Pikosekunden. Dies erfordert eine sehr hohe Bandbreite der Empfangseinheit, aufgrund derer die minimale Entfernungsauflösung stark begrenzt wird. Daher sind Laserscanner, welche auf der Messung der Phasenlaufzeit beruhen, besser geeignet. Hier wird ähnlich der PMD-Kamera ein moduliertes Lasersignal emittiert und die Phasenverschiebung zwischen empfangenem Lichtsignal und Referenzsignal mit einem Detektor ermittelt. Die Kombination beider Methoden, bei der ein Lichtimpuls mit einer modulierten Lichtfolge überlagert wird, hat sich als besonders geeignet herausgestellt.

Laserscanner sind geeignet für große Entfernungen und haben üblicherweise einen Messbereich bis zu 100m. Der Öffnungswinkel liegt modellabhängig zwischen 10° und 190°.

Abbildung 4.5.1: Laserscanner mit Phasenlaufzeitverfahren a) SICK LMS291-S05 b) rotoScan ROD-4 der Firma Leuze Electronic c) G 43600XA der Firma Götting

Beispielhaft zeigen die Abbildungen 4.5.1a-c verschiedene Ausführungen von Laserscannern, die auf dem Phasenlaufzeitverfahren basieren. So besitzt der Scanner

der Firma SICK einen Messbereich von 80 Metern und einen Öffnungswinkel von 180°. Die erreichbare Scanrate liegt bei 75Hz. Der systematische Messfehler wird mit ±35mm und der statistische mit 10mm [SICK10] angegeben. Der Laserscanner in Abbildung 4.5.1b besitzt einen Öffnungswinkel von 190° und eine Reichweite bis zu 65m. Die Wiederholgenauigkeit beträgt bei einer Entfernung von 4 Metern ±15mm.

Alle bisher vorgestellten Laserscanner sind 2D-Scanner. 3D-Daten werden üblicherweise durch Zusammenfügen mehrerer Scans erzeugt, welche während einer Dreh- oder Translationsbewegung des Scanners aufgenommen werden. Diese Bewegungen erfolgen durch Anbringung des Laserscanners an eine Dreh- oder Verschiebevorrichtung. Somit kann die relative Position des Scanners zur Startposition nachvollzogen werden. Die Aufnahmen können über die bekannte Lage des Scanners zum Zeitpunkt der Datengenerierung mittels einer Software fusioniert werden. Die Zeit für eine 3D-Aufnahme ist abhängig von der Rasterung und der Größe der Dreh- oder Translationsbewegung und liegt im Bereich von wenigen Sekunden bis hin zu einigen Minuten.

Vorteile	Nachteile
+ Großer Öffnungswinkel	- Bewegliche Teile
+ Großer Messbereich	- Nur zweidimensionale Datengenerierung
+ Hohe Wiederholfrequenz	- Hoher Preis
	- Größe und Gewicht

Tabelle 4.2: Vor- und Nachteile von Laserscannern

Tabelle 4.2 führt die Vor- und Nachteile im Gebrauch von Laserscannern auf. Die Vorteile eines Laserscanners liegen im großen Öffnungswinkel in Kombination mit einem großen Messbereich. Nachteilig sind jedoch mechanische, bewegliche Teile, wie zum Beispiel ein rotierender Spiegel, welche zu einer erhöhten Störanfälligkeit führen können, sowie der hohe Preis leistungsfähiger Laserscanner. Die Größe und das Gewicht führen zu einer schwierigen Integrierbarkeit bei kleinen Robotern mit einer nur geringen Zuladung. Die zweidimensionale Datengenerierung schränkt die Einsetzbarkeit der Laserscanner, im Vergleich zu dreidimensional messenden Systemen, zusätzlich ein. Durch Fusion mit einer Dreheinrichtung können zwar dreidimensionale Bilder erstellt werden, die resultierende Bildrate ist für eine Online-Kollisionserkennung jedoch zu gering.

4.5.2 Stereovisionsysteme

Stereovisionsysteme beruhen auf dem Prinzip der Stereotriangulation [Hartley03]. Hierzu werden zwei oder mehrere Kameras verwendet. Die Geräte werden üblicherweise in einem Abstand von wenigen Dezimetern parallel zueinander montiert, so dass die Sichtbereiche der Kameras einen großen Überlappungsbereich aufweisen. Unter Verwendung von speziellen Kalibrierverfahren können die intrinsischen und extrinsischen Parameter bestimmt werden. Dies ermöglicht die Berechnung korrespondierender Punkte mittels Korrelationsverfahren. Unter Ausnutzung der Epipolargeometrie [Kraus97] ist eine Beschleunigung dieser Berechnungen erzielbar. Für komplexe Szenen benötigt diese jedoch zum Teil hohe Berechnungszeiten.

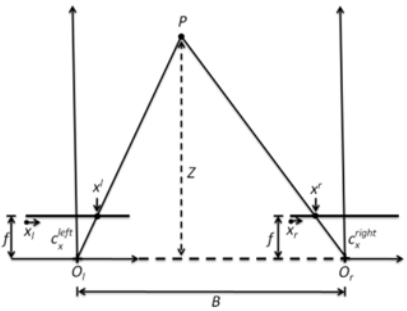


Abbildung 4.5.2: Triangulation für korrespondierende Punkte bei Stereokameras

Abbildung 4.5.2 zeigt eine Skizze zur Berechnung der Objektentfernung für einen beliebigen korrespondierenden Punkt *P*. Über die intrinsische Kalibrierung werden die Brennweite f und die Verschiebung (c_x/c_y) des Bildkoordinatensystems zur optischen Achse berechnet. Die Verschiebung *B* zwischen den optischen Achsen der Kameras wird mittels extrinsischer Kalibrierverfahren ermittelt. Dies bietet die Möglichkeit 3D-Bildpunkte über das Triangulationsprinzip zu berechnen.

$$\frac{B - \left(x^l - x^r\right)}{Z - f} = \frac{B}{Z} \tag{4.9}$$

Aus Gl.(4.9) folgt für die Entfernung Z von P:

$$Z = \frac{f \cdot B}{x^l - x^r} \tag{4.10}$$

Die Differenz zwischen den Pixeln $x^l - x^r$ entspricht der Disparität.

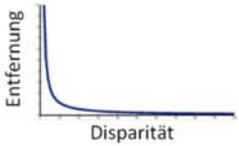


Abbildung 4.5.3: Zusammenhang zwischen Entfernung und Disparität

Die Disparität weist eine nichtlineare umgekehrte Proportionalität gegenüber der gemessenen Entfernung auf. Für die Qualität des Entfernungswertes bedeutet dies, dass Objektpunkte mit geringer Entfernung zur Kamera höhere Genauigkeiten aufweisen und somit die Kamera für die Messung von Distanzen im Nahbereich der Messeinrichtung besser geeignet ist.

Vorteile	Nachteile
+ Geringer Hardwareaufwand	- Tiefenauflösung entfernungsabhängig
+ Geringe Kosten	- Geringer Messbereich
+ Hohe laterale Auflösung	- Benötigt Software zur 3D-Datengewinnung
+ Farbinformationen	- Erzielbare Framerate abhängig von Kameraauflösung und Szenenkomplexität
+ Keine aktive Beleuchtung	- Laterale Auflösung der 3D-Daten ist abhängig von den Strukturen in der Szene

Tabelle 4.3: Vor- und Nachteile von Stereovisionsystemen

In Tabelle 4.3 sind die Vor- und Nachteile bei der Verwendung von Stereovisionsystemen aufgeführt. Vorteile liegen im geringen Hardwareaufwand in Verbindung mit geringen Kosten. Durch die Fusion der Bilder kann eine 3D-Punktwolke generiert werden, die mit Farbinformationen belegt werden kann. Dies ermöglicht die Entwicklung von robusten und einfachen Methoden zur Objekterkennung. Nachteilig jedoch sind, abhängig vom Einsatzbereich, die entfernungsabhängige Tiefenauflösung des Kamerasystems und der daraus resultierende geringe Messbereich. Zudem wird eine spezielle Software benötigt, welche die 3D-Daten aus den zwei zueinander verschobenen Bildern extrahiert. Hierfür werden wertvolle Rechenkapazitäten benötigt, die für die spätere Datenverarbeitung nicht mehr zur Verfügung stehen. Ferner ist die Berechnungszeit und somit die erzielbare Framerate stark abhängig von der Struktur und der Komplexität der aufgenommenen Szenen. Bei Bildern die eine Szene mit geringer Struktur und großen Flächen zeigen, können nur wenige korrespondierende Punkte gefunden werden. Die Berechnungszeit der 3D-Daten ist folglich kürzer. Ein weiterer hieraus resultierender Nachteil ist, dass bei der Messung

mit einem Stereovisionsystem keine homogene Punktwolke entsteht. Die Anzahl der generierten 3D-Daten nimmt mit der Komplexität der Szene zu beziehungsweise ab.

4.5.3 Vergleich der optischen Entfernungsmesssysteme

Die Realisierung der in Kapitel 2 definierten Ziele im Bereich der Handhabungs- und mobilen Robotik stellt spezielle Anforderungen an das zu verwendende Entfernungsmesssystem. Für die mobile Robotik sollte der Messbereich des Entfernungsmesssystems im Bereich von 5m - 20m liegen. Dies ermöglicht das rechtzeitige Erkennen von Hindernissen. Im Nahbereich ist zudem eine Messungenauigkeit von wenigen Millimetern gefordert, so dass eine präzise Selbstlokalisierung durchgeführt werden kann. Für größere Entfernungen darf diese zunehmen, da für die Hindernisbestimmung lediglich eine Genauigkeit im Bereich von wenigen Zentimetern gewünscht ist. Allerdings bedarf es einem möglichst großen Öffnungswinkel, damit Objekte seitlich des Fahrwegs auch bei geringer Entfernung noch zu detektieren sind. Ebenso sind eine hohe Bildrate, sowie eine geringe Baugröße des Sensors zur leichten Integrierbarkeit gefordert. Die Handhabungsrobotik stellt andere Anforderungen an das Messsystem. Der geforderte Messbereich beträgt hier maximal 4m. Die Messgenauigkeit sollte im Bereich 0,5cm bis 2cm liegen. Ein größerer Öffnungswinkel ist ebenfalls wünschenswert, damit der gesamte Arbeitsbereich des Roboters überwacht werden kann. Unabdingbar für die Aufgabenstellung ist die Aufnahme von dreidimensionalen Bildern. Die Bildrate sollte möglichst hoch sein. Die Baugröße ist jedoch an dieser Stelle zweitrangig.

	PMD-Kamera	**Laserscanner**	**Laserscanner** mit Schwenkeinrichtung	**Stereovision-systeme**
Reichweite	+	++	++	+
Messgenauigkeit	+	+	+	0
Öffnungswinkel	0	++	++	0
3D-Bild	++	-	++	+
Bildrate	+	+	-	+
Baugröße	++	0	-	-

Tabelle 4.4: Vergleich der Entfernungsmesssysteme

Eine Beurteilung der Sensoren, im Hinblick auf deren Eignung, kann nur in Bezug auf die Aufgabenstellung getroffen werden. Aus diesem Grund werden die Messsysteme

nach Tabelle 4.4 zunächst nach ihren Eigenschaften im Hinblick auf die mobile Robotik untersucht.

Im Bereich der mobilen Robotik soll die Kamera zur Selbstlokalisierung und Hinderniserkennung eingesetzt werden. Um Objekte rechtzeitig zu erkennen, ist ein Messbereich von mindestens 5 Meter gefordert. Dies wird von allen Systemen erfüllt. Der Laserscanner bietet mit bis zu 100m die größte Reichweite. Allerdings kann dieser die Umgebung zur zweidimensional vermessen. Die Höhe von Hindernissen, sowie negative Hindernisse (Löcher, Treppen, Abhänge) können nicht erkannt werden. Aufgrund der dreidimensionalen Datenaufnahme sind PMD-Kameras und Stereovisionsysteme besser geeignet. Vorteil bei der Verwendung von Laserscannern ist der im Vergleich zur PMD-Technik größere Öffnungswinkel. Dies ermöglicht die Überwachung des Bereichs seitlich vor dem Roboter. Der Sichtbereich der PMD-Kamera und der Stereovisionsysteme ist stark beschränkt. Die Erkennung von künstlichen Landmarken, sowie die Berechnung von Bewegungsvektoren zur Selbstlokalisierung sind bei Gebrauch eines Laserscanners nicht beziehungsweise nur in Verbindung mit einem Stereovisionsystem möglich. Hier ist die PMD-Technik im Vorteil. Die zusätzliche Aufnahme von Grauwerten ermöglicht die Verwendung von Standardbildverarbeitungsalgorithmen zur Erkennung von definierten Objekten, sowie zur Berechnung von deren Lage durch Berücksichtigung der 3D-Werte der jeweiligen Pixel.

Innerhalb der Anwendungen zur Handhabungsrobotik soll das Sensorsystem zum einen zur Überwachung der Roboterzelle dienen, so dass Kollisionen zwischen Roboter und Personen beziehungsweise beliebigen Objekten vermieden werden können. Zum anderen soll unter Verwendung der 3D-Daten die Trajektorie des Roboters auf Kollision geprüft und gegebenenfalls eine neue Bahn geplant werden können.

Die Verwendung von Laserscannern ist hier nur bedingt geeignet. 3D-Daten können nur mit zusätzlicher Schwenkeinrichtung generiert werden. Dies führt zu einer geringen Bildwiederhohlfrequenz. Somit können Laserscanner lediglich zur Überwachung der Arbeitsraumgrenzen eingesetzt werden und nicht zur vollständigen Überwachung der Roboterzelle, sowie zur Aufnahme bahnplanungsrelevanter 3D-Daten. Für diese Aufgabe ist die Verwendung von PMD-Kamera oder Stereovisionsystem besser geeignet. Sollte ein Stereovisionsystem aus zwei oder mehreren Kameras aufgebaut werden, besteht die Schwierigkeit einen möglichst großen gemeinsamen Sichtbereich zu gewährleisten, so dass zur Überwachung des Roboterarbeitsraumes genügend 3D-Daten generiert werden können. Die Verwendung der PMD-Kamera ist aufgrund ihrer Eigenschaften und einfachen Integrierbarkeit die geeignetere Wahl.

Der aufgeführte Vergleich der Eigenschaften der verschiedenen optischen Messsysteme mit denen der PMD-Kamera verdeutlicht das Potential einer innovativen Nutzung letztgenannter, im Bereich des dreidimensionalen Sehens autonomer Robotersysteme, insbesondere im Hinblick auf die in Kapitel 2 definierten Ziele.

5 Die PMD-Kamera als neuartiges Sensorkonzept für eine sichere autonome Navigation mobiler Roboter

Üblicherweise müssen Industrieprodukte nach ihrer Herstellung zwischengelagert werden. Dies erfordert eine Beförderung der produzierten Teile von der Fertigungshalle in ein Zwischenlager. Zur Verringerung von Kosten und Fehlern die Menschen durch eine solch hochmonotone Arbeit verursachen, ist man in der Vergangenheit dazu übergegangen diesen Prozess zu automatisieren. Für die Navigation autonomer Transportsysteme werden verschiedene Methoden verwendet. Die am meisten verbreiteten Methoden sind die Navigation anhand von auf den Boden befindlicher Linien oder in den Boden eingelassener Transponder, welche mit Sensoren erkannt werden und an denen das System entlang fährt. Sollte die Navigation außerhalb von Gebäuden stattfinden, wird häufig zusätzlich GPS zur Selbstlokalisierung verwendet. Die Transportsysteme navigieren oft in sehr dynamischen Umgebungen. Um plötzlich auftretende Hindernisse zu erkennen und um Kollisionen mit diesen zu vermeiden, werden die Fahrzeuge üblicherweise mit Laserscannern ausgerüstet. Diese registrieren fremde Objekte vor dem Fahrzeug, welches infolgedessen stehen bleibt, bis das Hindernis entfernt wird. Hierbei wird darauf verzichtet, dass das Fahrzeug dem Hindernis autonom ausweicht.

Dieses Kapitel beschreibt die Verwendung der PMD-Technologie zur Selbstlokalisierung und Steuerung eines fahrerlosen Transportsystems. Hierzu soll die PMD-Kamera sowohl zur Hindernis- und Kollisionsvermeidung verwendet werden, als auch durch Fusion mit Radencodern zur Verbesserung der Genauigkeit bei der Selbstlokalisierung beitragen. Nachfolgend wird der Aufbau einer mit PMD-Kameras versehenen, mobilen Versuchsplattform beschrieben.

Der mit PMD-Sensorik ausgestatte Roboter soll autonom in einer nur teilweise bekannten Umgebung navigieren. Hierunter ist zu verstehen, dass dem Roboter eine CAD-Karte der Umgebung zur Bahnplanung und zur autonomen Navigation zur Verfügung steht. Allerdings besteht die Möglichkeit, dass sich Hindernisse in dem von dem Roboter zu befahrenen Gebiet befinden können, die nicht in der Karte verzeichnet sind.

Somit ist ein geeigneter Algorithmus zur schnellen Bahnplanung anhand von 2D-Umgebungskarten zu entwickeln, welcher die Planung von kollisionsfreien Trajektorien zwischen beliebigen Start- und Zielpunkten innerhalb der Karte ermöglicht. Zum autonomen Abfahren der geplanten Trajektorie soll ein echtzeitfähiger Bahnregelungsalgorithmus entwickelt werden, der ein möglichst genaues Abfahren der geplanten Bahn gewährleistet. Die Roboterposition und Orientierung soll durch eine Fusion zwischen Radencoderdaten und den aus den 3D-Aufnahmen des Kamerasystems berechneten Bewegungsvektoren ermittelt werden. Eine Korrektur dieser relativen Pose-Schätzung soll mit der Hilfe von künstlichen Landmarken, welche in der Umgebungskarte verzeichnet sind, ausgeführt werden. Dies erfordert die Implementierung von PMD spezifischen Methoden zur Erkennung geeigneter Landmarken, bei gleichzeitiger Berechnung von deren Lage bezüglich des Roboterkoordinatensystems.

Bei der Navigation durch die nur teilweise bekannte Umgebung muss der Bereich vor der mobilen Roboterplattform mit der Kamera beobachtet werden. Die einzelnen Aufnahmen sind in das Weltkoordinatensystem zu transformieren und dort in eine geeignete Darstellungsform zu konvertieren, um diese schnell und effektiv mit der Umgebungskarte abzugleichen. Neu erkannte Hindernisse sind in eine zweite Hinderniskarte einzutragen. Die geplante Trajektorie muss daraufhin „online“ auf eventuelle Kollision mit dem neuen Hindernis überprüft werden. Sollte die Überprüfung der Trajektorie eine mögliche Kollision erkennen, muss der Bahnplaner, wenn dies möglich ist, eine neue Bahn von der Ist- zur Zielposition berechnen. Sollte keine Neuplanung zum Ziel möglich sein, muss der Roboter stehen bleiben und eine Meldung an den Benutzer ausgeben, damit das Hindernis entfernt wird und der Roboter mit dem Abfahren der ursprünglichen Trajektorie fortfahren kann.

Zusätzlich sollen Algorithmen auf Basis der PMD-Sensorik implementiert werden, die ein autonomes Andocken des mobilen Roboters an eine Anhängerdeichsel ermöglichen. Häufig werden fahrerlose Transportsysteme für die Beförderung von Anhängern eingesetzt. Diese werden in der Fertigungshalle befüllt. Die mobile

Roboterplattform holt diese ab und bringt sie zum Zwischenlager, fährt dann zurück und holt den nächsten Anhänger. Hierbei ist es notwendig die Anhänger von der autonomen Zugmaschine an- und abzukoppeln. Dies erfolgt zurzeit noch vollständig manuell. Um diesen Prozess zu automatisieren, wollen wir ebenfalls eine PMD Kamera einsetzen, welche am Heck des Fahrzeugs angebracht wird und eine Lokalisierung der Anhängerdeichsel in Echtzeit ermöglicht. Basierend auf der Erkennung der Anhängerdeichsel ist ein Algorithmus für die Andockregelung zu entwerfen.

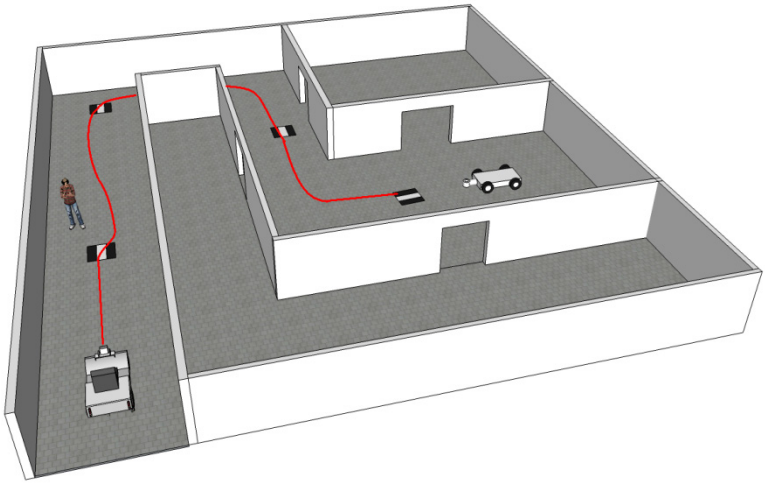


Abbildung 4.5.1: Beispiel einer möglichen Roboterumgebung

Zur Verdeutlichung der beschriebenen Aufgabenstellungen zeigt Abbildung 4.5.1 die Skizze einer vorstellbaren Roboterumgebung. Der Roboter navigiert mittels der PMD-Kamera von einer Start- zu einer vorgegebenen Zielposition und dockt dort an einen Anhänger an. Durch die Erkennung von künstlichen Landmarken, deren Positionen in der Karte bekannt sind, soll eine Positionskorrektur ausgeführt werden. Erkennt das Fahrzeug Hindernisse, wie dies zum Beispiel durch die Person in Abbildung 4.5.1 angedeutet wird, weicht das Fahrzeug dem Hindernis autonom aus, sofern hierzu die Möglichkeit besteht. Ansonsten wartet der Roboter solange bis das detektierte Hindernis die Fahrbahn wieder freigibt. Ein vorstellbarer Pfad, welcher von dem Roboter abgefahren werden könnte, ist durch die angedeutete Trajektorie in obiger Skizze gekennzeichnet.

5.1 Versuchsaufbau

5.1.1 Aufbau der mobilen Roboterplattform

Basis der in dieser Arbeit verwendeten mobilen Roboterplattform ist der Cityliner 412 der Firma Meyra Orthopädia. Abbildung 5.1.1 zeigt den originalen Zustand des Fahrzeugs. Dieses wurde für die Verwendung als fahrerloses Transportsystem (FTS) mit verschiedener Sensorik und Aktorik ausgestattet.

Abbildung 5.1.1: Electroscooter der Firma Meyra Ortopädia GmbH

Die verbaute Sensorik besteht aus zwei in den Hinterachsen verbauten Radencodern, welche zur Ermittlung der Odometriedaten wie Geschwindigkeit, Position und Orientierung des Fahrzeugs dienen. Radencoder, Lenkservomotor sowie Antriebsmotorelektronik sind mit einem Bahnführungsrechner der Firma Götting KG verbunden. Dieser dient dem Auslesen und Vorverarbeiten der Odometriedaten, so dass über diesen Geschwindigkeit und Position des Fahrzeugs, bezogen auf den Startpunkt, von dem Benutzer über eine CAN-Bus Schnittstelle abgefragt werden kann. Des Weiteren übernimmt der Bahnführungsrechner die Lenkwinkel- und Geschwindigkeitsregelung.

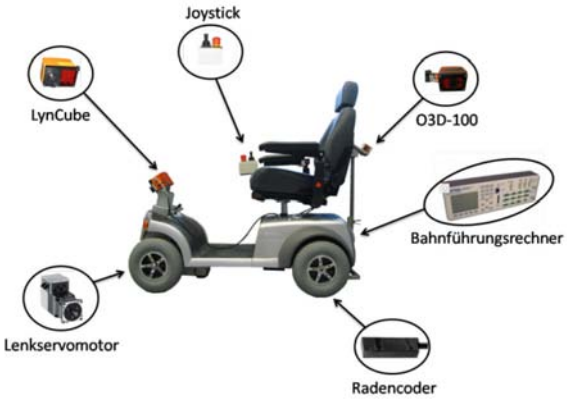


Abbildung 5.1.2: Sensorik der mobilen Roboterplattform

Die mobile Roboterplattform besitzt bei einem Leergewicht von 106kg eine maximale Vorwärtsgeschwindigkeit von $v = 12km/h$. Die Geschwindigkeit in Rückwärtsrichtung ist auf $v = 6\ km/h$ begrenzt. Der maximale Lenkwinkel liegt bei $\varphi = \pm 35°$. Die Abmessungen des Fahrzeugs betragen (l)1200mm x (b)650mm bei einem Radstand von 850mm.

Zur Überwachung der Roboterumgebung wurden, wie auf Abbildung 5.1.2 dargestellt, zwei auf dem PMD-Prinzip basierende Kameras verbaut. Vorne, zur Überwachung des Fahrwegs, die im Lynkeus-Projekt entwickelte LynCube mit einer lateralen Auflösung von 200x200 Pixeln und hinten die O3D-100 der Firma ifm, mit einer lateralen Auflösung von 64x50 Pixeln. Diese dient, wie in der Einleitung dieses Kapitels beschrieben, der Durchführung autonomer Andockmanöver an eine Anhängerdeichsel.

In Kapitel 4.3 wurde die Problematik des Eindeutigkeitsbereichs bei der Verwendung einer auf dem PMD-Prinzip basierenden 3D-Kamera betrachtet. In Anbetracht dessen muss, wie auf Abbildung 5.1.3 verdeutlicht wird, die Kamera zum Fahrweg hin geneigt sein, um Messfehler beziehungsweise Fehlinterpretationen bei der Distanzmessung durch das Überschreiten des Messbereichs zu verhindern.

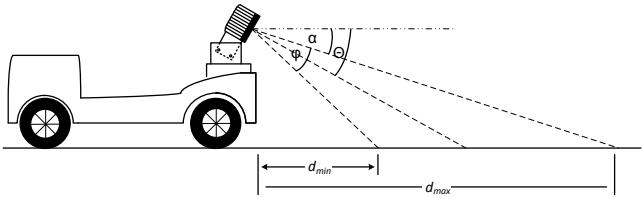


Abbildung 5.1.3: Skizze zur Montage der Kamera an der Roboterplattform

Der minimale Einbauwinkel θ_{min} bei dem der Eindeutigkeitsbereich noch gewahrt wird, berechnet sich nach Abbildung 5.1.3 zu:

$$\alpha = \frac{\pi}{2} - arccos\left(\frac{h}{d_{max}}\right) \tag{5.1}$$

Hieraus folgt für den minimalen Neigungswinkel der Kamera:

$$\theta_{min} = \alpha + \frac{\varphi}{2} \tag{5.2}$$

Bei einer Einbauhöhe $h = 0{,}5m$ und einer Modulationsfrequenz $f_{mod} = 20MHz$, welches einen Eindeutigkeitsbereich von $d_{max} = 7{,}5m$ bewirkt, beträgt der minimale Neigungswinkel:

$$\theta_{min} = \frac{\pi}{2} - \arccos\left(\frac{h}{d_{max}}\right) - \frac{\varphi}{2} = 20{,}38° \tag{5.3}$$

Der maximale Neigungswinkel sollte so gewählt werden, dass die minimale Distanz $d_{min} = 0{,}5m$ nicht unterschritten wird. Der resultierende maximale Neigungswinkel θ_{max} der Kamera beträgt:

$$\theta_{max} = \frac{\pi}{2} - \arccos\left(\frac{h}{d_{max}}\right) - \varphi = 50° \tag{5.4}$$

Der Einbauwinkel θ der Kamera unterliegt nach Gl.(5.3) und Gl.(5.4) einer Begrenzung von $20{,}38° \leq \theta \leq 50°$. Für eine möglichst große Abdeckung des zu beobachtbaren Bereichs vor dem Fahrzeug ist ein Neigungswinkel θ nahe θ_{min} zu wählen. In der Praxis jedoch müssen Faktoren wie Reflektivität des Bodens, eine eventuelle Neigung der Fahrbahn und die Stärke der Beleuchtung berücksichtigt werden. Ist der Neigungswinkel zu gering, empfängt der PMD-Sensor zu wenig Licht und das Messrauschen der Distanzwerte erhöht sich dementsprechend. Beim Einstellen des

Kameraneigungswinkels θ ist folglich ein Kompromiss zwischen maximal beobachtbaren Bereich und dem Entfernungsmessrauschen in den Grenzen $\theta_{min} \leq \theta \leq \theta_{max}$ zu finden.

Die Anbringung der hinteren Kamera, siehe Abbildung 5.1.2, unterliegt denselben Restriktionen. Jedoch muss beim Einstellen des Neigungswinkels in diesem Fall ein Kompromiss zwischen rechtzeitigem Erkennen der Anhängerdeichsel und der Mindestentfernung, in welcher Objekte hinter dem Fahrzeug noch erkannt werden sollen, getroffen werden. Im Kapitel 5.9 „Autonomes Andocken an eine Anhängerdeichsel“ wird detailliert auf diese Problematik eingegangen.

5.1.2 Netzwerkarchitektur

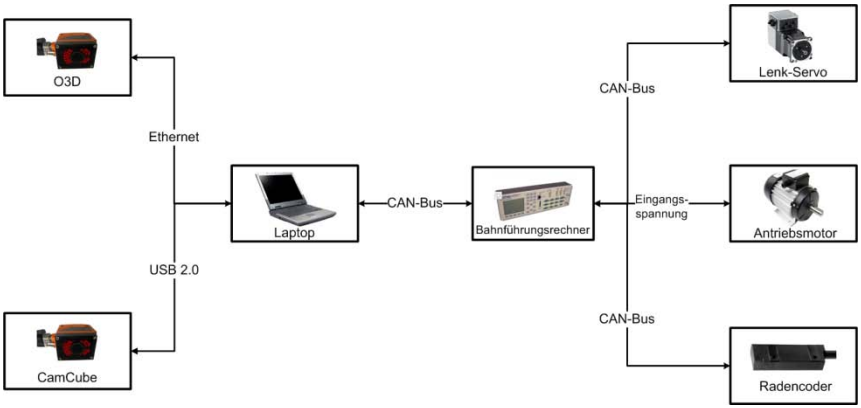


Abbildung 5.1.4: Netzwerkarchitektur

In Abbildung 5.1.4 ist die Netzwerkarchitektur des Robotersystems dargestellt. Zur Steuerung des Roboters dient ein Bahnführungsrechner. Auf diesem ist die Regelung des Lenkservomotors und der Geschwindigkeitsregler implementiert. Die Steuerung des Lenkservomotors ist mit dem Bahnführungsrechner über eine CAN-Bus Schnittstelle verbunden. Hierrüber kann der Lenkwinkel mit einer Zykluszeit von $25\,ms$ abgefragt werden. Zur Steuerung des Antriebsmotors dient eine Ausgangsspannung zwischen $0V$ und $24V$. Ebenfalls über CAN-Bus werden die Radencoder ausgelesen. Über diese erhält der Bahnführungsrechner Informationen bezüglich Geschwindigkeit und relativer Position des Fahrzeugs.

Bildverarbeitung, Bahnplanung, Hinderniserkennung sowie Selbstlokalisierung werden auf einem Computer durchgeführt. Dieser ist zum einen über CAN-Bus mit dem Bahnführungsrechner verbunden und kann so Lenkwinkel und Geschwindigkeit des Roboters vorgeben. Des Weiteren können mit Hilfe des Bahnführungsrechners die Odometriedaten in kartesischen Koordinaten, sowie die Geschwindigkeit ausgelesen werden. Zum anderen ist der Computer über Ethernet mit der O3D-100 und über USB mit der LynCube verbunden. Alle Daten werden von diesem fusioniert und verarbeitet.

5.2 Aufbau der Testumgebung

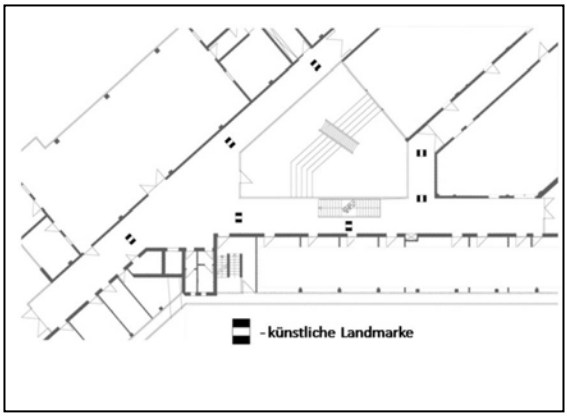


Abbildung 5.2.1: Testumgebung im Bürogebäude

Die verwendete Testumgebung zur Navigation der mobilen Roboterplattform ist in Abbildung 5.2.1 dargestellt. Der Grundriss der Umgebung wird dem Roboter zur Bahnplanung in Form einer CAD-Karte zur Verfügung gestellt. Künstliche Landmarken sind in dem vom Roboter abzufahrenden Areal positioniert, deren exakte Positionen sind in der Umgebungskarte eingetragen. Diese dienen dem mobilen Roboter zur Bestimmung der Startposition sowie zur Verifizierung seiner Pose während der Trajektorienausführung.

5.3 Kalibrierung zwischen Kamera- und Roboterkoordinaten-system

Zur Objekt- und Hinderniserkennung müssen die Kamera- und Roboterdaten fusioniert werden. Hierzu müssen die jeweiligen Daten in einem gemeinsamen Koordinatensystem vorliegen. Dies erfordert die Entwicklung eines geeigneten Kalibrierverfahrens, welches die Überführung der Kameradaten in das Roboter-koordinatensystem als gemeinsames Koordinatensystem ermöglicht.

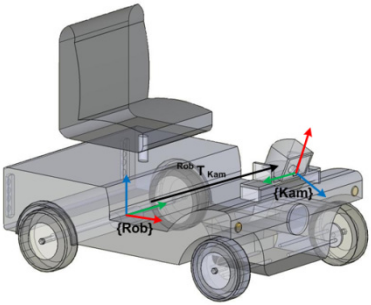


Abbildung 5.3.1: Kameraregistrierung

Die 3D-Kamera wird auf dem Roboter mit einer Halterung montiert, die das Einstellen eines beliebigen Neigungswinkels ermöglicht. Wie bereits in Kapitel 5.1 diskutiert, befindet sich der einstellbare Neigungswinkel θ in den Grenzen

$$24° \leq \theta \leq 50° \tag{5.5}$$

und muss zur Reduzierung des Messrauschens auf die verschiedenen Bodenbeschaffenheiten angepasst werden.

Abbildung 5.3.1 zeigt die Position und Orientierung beider Koordinatensysteme. Der Ursprung des Roboterkoordinatensystems liegt im Mittelpunkt der Hinterachse des Roboters. Die x-Achse zeigt in Fahrtrichtung, die y-Achse nach rechts und die z-Achse senkrecht nach oben. Der Ursprung des Systems liegt im Mittelpunkt des Objektivaustrittsfensters.

Die Transformationsmatrix ${}^{Rob}T_{Kam}$ setzt sich zusammen aus dem Translationsvektor ${}^{Rob}\vec{t}_{Kam}$ und der Rotationsmatrix ${}^{Rob}R_{Kam}$:

$$ {}^{Rob}T_{Kam} = \begin{bmatrix} & {}^{Rob}R_{Kam} & & {}^{Rob}\vec{t}_{Kam} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6} $$

Für die Kalibrierung können beide Anteile getrennt voneinander betrachtet werden.

Der translatorische Anteil ergibt sich aus:

$$ {}^{Rob}\vec{t}_{Kam} = {}^{Rob}\vec{t}_{HK} + {}^{HK}\vec{t}_{Kam} \tag{5.7} $$

wobei ${}^{Rob}\vec{t}_{HK}$ aus dem mechanischen Aufbau des Roboters bestimmt wird und der Translation zwischen Roboterkoordinatensystem und der Drehachse der Kamerahalterung entspricht. ${}^{HK}\vec{t}_{Kam}$ ist abhängig von dem eingestellten Neigungswinkel θ und berechnet sich aus:

$$ {}^{Rob}\vec{t}_{HK} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \cdot \begin{bmatrix} t_x \\ 0 \\ t_z \end{bmatrix} \tag{5.8} $$

Hierbei entsprechen t_x und t_z der Verschiebung von Kamerahalterung bis zum Objektiv für den Fall, dass der Neigungswinkel θ gleich Null ist. Sie werden durch den mechanischen Aufbau bestimmt. Die Verschiebung auf der y-Achse ist, unabhängig von der eingestellten Neigung, gleich Null. Der Neigungswinkel kann aus der Rotationsmatrix berechnet werden.

Die Ermittlung der Rotationsmatrix ${}^{Kam}R_{Rob}$ erfolgt durch ein spezielles Messverfahren. Hierbei wird durch Kameramessungen, zum Teil in Kombination mit Roboterbewegungen, die x- und z-Achse des Roboterkoordinatensystem im Kamerakoordinatensystem bestimmt. Durch die Bildung des Kreuzprodukts:

$$ \vec{y} = \vec{z} \times \vec{x} \tag{5.9} $$

kann anschließend die y-Achse berechnet werden.

$$ {}^{Kam}R_{Rob} = (\vec{x}, \vec{y}, \vec{z}) \tag{5.10} $$

Die ermittelten Vektoren können nach Gl.(5.10) zur Rotationmatrix ${}^{Kam}R_{Rob}$ zwischen Kamera- und Roboterkoordinatensystem zusammengefügt werden. Durch Bildung der Inversen wird die gesuchte Matrix:

$$^{Rob}R_{Kam} = {}^{Kam}R_{Rob}^{-1} \tag{5.11}$$

ermittelt.

5.3.1 Bestimmung des z-Vektors des Roboterkoordinatensystems

Der z-Vektor des Roboterkoordinatensystems besitzt die gleiche Orientierung wie der Normalenvektor des Bodens, auf welchem der Roboter steht, vorausgesetzt dieser weist keine Krümmung auf. Der Normalenvektor des Bodens kann durch Berechnung des Normalenvektors der Ebene, welche von den 3D-Daten der PMD-Kamera aufgespannt wird, im Kamerakoordinatensystem ermittelt werden. Zur präziseren Berechnung der Ebene wird zur Verminderung des Einflusses verrauschter Pixel beziehungsweise zur Verringerung des Einflusses von Messfehlern, für die Berechnung der Ebenengleichung der RANSAC-Algorithmus [Fischler81] in Kombination mit einem Least-Mean-Square (LMS) Verfahren angewandt. Dieser dient der Filterung von Pixeln, welche eine hohe Abweichung von der Ebene aufweisen.

Der RANSAC-Algorithmus

Der Begriff RANSAC steht für „**Ran**dom **Sa**mple **C**onsensus“. Der Algorithmus wurde im Jahr 1981 von [Fischler81] entwickelt und findet Verwendung zur zuverlässigen Berechnung von Modellparametern, auch bei einer höheren Anzahl an Messwertausreißern. Der RANSAC-Algorithmus ist ein iterativer Algorithmus der bei jeder Iteration so viele Punkte zufällig aus einer Punktmenge wählt, wie zur Berechnung des Modells nötig sind. Ein solches Modell kann eine Linien-, Ebenen- oder auch eine Zylindergleichung sein. Abhängig von dem berechneten Modell werden im Folgenden die Abstände aller Punkte zu diesem berechnet. Ist der Abstand eines Punktes geringer als eine gewählte Fehlerschranke, wird dieser in einem sogenannten Concensus-Set abgespeichert. Weil mit zunehmender Größe des Consensus-Sets die Wahrscheinlichkeit steigt, dass die für die Berechnung des Modells berücksichtigte Punktmenge frei von Ausreisern ist, wird das Modell, welches die größte Anzahl an Punkten beinhaltet, anschließend für die Schätzung der Modellparameter mit dem LMS-Algorithmus berücksichtigt. Die Anzahl der hierbei mindestens durchgeführten Iterationen n wird festgelegt auf:

$$n = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \tag{5.12}$$

s entspricht der Anzahl, der für die Berechnung notwendigen Punkte, p der Wahrscheinlichkeit mit der mindestens einmal eine Menge an Punkten ausgewählt

wird die ausreißerfrei ist und ε dem relativen Anteil an Punkten, deren Abstand zum gewählten Modell geringer als die gewählte Fehlerschranke ist.

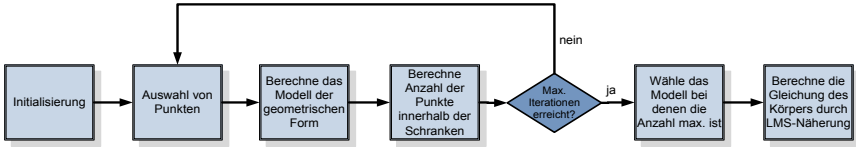


Abbildung 5.3.2: Flussdiagramm zur Implementierung des RANSAC-Algorithmus

Abbildung 5.3.2 zeigt das Flussdiagramm des implementierten RANSAC-Algorithmus. Beim Start des Algorithmus findet zunächst eine Initialisierung statt, welche die Anzahl der maximalen Iterationen, sowie die Größe der Fehlerschranke festlegt. Bei dem für die Berechnung der Ebenengleichung umgesetzten Verfahren wird eine obere Schranke von $f < 0{,}05m$ festgesetzt. Die Wahrscheinlichkeit mindestens eine fehler- und ausreißerfreie Zufallsprobe für die Berechnung der Ebenengleichung zu finden, liegt bei $p = 99\%$ und die maximale Fehlerwahrscheinlichkeit bei $\varepsilon = 50\%$. Somit errechnet sich die minimale Anzahl der Iterationen für den RANSAC-Algorithmus zu:

$$n = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} = \frac{\log(1-0{,}99)}{\log(1-(1-0{,}5)^3)} \tag{5.13}$$

Nach der Initialisierung werden für den dargestellten Fall drei Zufallspunkte ausgewählt, aus denen die Ebenengleichung berechnet wird. Ist diese bestimmt, können die Entfernungen aller Punkte zu der berechneten Ebene ermittelt werden. Die Punkte, die unterhalb der gewählten Fehlerschranke von $f = 0{,}05m$ liegen, werden im Consensus-Set abgespeichert. Die Schritte werden solange wiederholt, bis die Anzahl der maximalen Iterationen erreicht ist. Anschließend wird die Ebenengleichung über einen LMS-Algorithmus aus den Punkten des Consensus-Sets mit der größten Datenmenge bestimmt.

LMS-Algorithmus zur Berechnung einer Ebenengleichung aus 3D Punktmengen

Zur Berechnung der Ebenengleichung nach Anwendung des RANSAC-Algorithmus wird ein Verfahren auf der Basis der Minimierung der Fehlerquadrate angewendet.

In diesem Fall erfolgt die lineare Ebenenapproximation von Punktmenge im $\mathbb{R}^3$. Hierfür stehen in der Literatur zwei verschiedene Ansätze zur Verfügung. Der erste Ansatz ist

die achsenparallele Approximation [Goetze94], bei der die Distanzmessung parallel zu den Koordinatenachsen durchgeführt wird. Der Ansatz führt zur Lösung linearer Gleichungssysteme, deren Koeffizientenmatrix Gramschen (2×2)-Matrizen entsprechen. Die zweite Möglichkeit ist die Lösung über eine orthogonale Approximation [Goetze94], also die Distanzberechnung zum Lot des Punktes auf die Ebene. Diese führt zu einem Eigenwertproblem einer Gramschen (3×3)-Matrix.

Der gewählte Ansatz entspricht einer Berechnung der Ebenengleichung über die achsen-parallele Approximation. Gesucht wird eine Ebenengleichung der Form:

$$ax + by + cz = 0 \tag{5.14}$$

Bedingung für die Anwendung der parallelen Approximation ist die Verschiebung des Schwerpunkts der Punktwolke in den Koordinatenursprung, so dass gilt:

$$\bar{x} = \bar{y} = \bar{z} = 0 \tag{5.15}$$

Der Abstand zur z-Ebene entspricht:

$$d_i = z_i - f_e(x_i, y_i) \tag{5.16}$$

Die Summe dieser Abstände:

$$\Delta = \sum_{i=0}^{n} d_i \tag{5.17}$$

soll minimal werden. Dies entspricht dem Problem der Minimierung des kleinsten Fehlerquadrates. Der Ansatz zur Lösung dieses Problems lautet:

$$z - f(x, y) = ax + by + d \tag{5.18}$$

Aufgrund der Verschiebung der Punktwolke zum Koordinatenursprung gilt $d = 0$ und somit gelangen wir zu folgendem Minimierungsproblem:

$$\Delta = \sum_{i=0}^{n} d_i^2 = \sum_{i=0}^{n} (z_i - (ax_i + by_i))^2 = \|z - (ax + by)\|^2 \tag{5.19}$$

wobei $\|\ldots\|$ der euklidischen Norm entspricht. Nach [Goetze94] hat das formulierte Problem folgende Lösungen:

$$a = \frac{G_x}{G}\,, \qquad b = \frac{G_y}{G}$$

mit

$$G_x = \begin{vmatrix} xz & xy \\ yz & yy \end{vmatrix}; \quad G_y = \begin{vmatrix} xx & xz \\ yx & yz \end{vmatrix} \text{ und } G = \begin{vmatrix} xx & xy \\ yx & yy \end{vmatrix}$$

Dies führt zu der Ebenengleichung:

$$G_x x + G_y y - Gz = 0 \tag{5.20}$$

Gl.(5.20) gilt nicht für $G = 0$. Dies ist der Fall wenn:

$x = 0$ oder $y = 0$ oder $x \neq 0$ und $y \neq 0$, wenn $\alpha \neq 0$ ist, so dass gilt $y = \alpha \cdot x$

Sind obige Bedingungen erfüllt, kann die Aufgabe durch eine parallele Transformation auf eine andere Ebene des Koordinatensystems gelöst werden. Führt die z-Approximation zu einer erfolgreichen Lösung, werden im nächsten Berechnungsschritt zusätzlich x- und y- Approximationen durchgeführt. Zur Berechnung werden in den obigen Ergebnissen die Indizes entsprechend vertauscht. Sind alle Approximationen durchgeführt und führen diese zu einem positiven Ergebnis, werden sie fusioniert. Dies geschieht durch das Gleichrichten der normierten Orthonormalvektoren

$$\vec{e}^{\,i} = (a^i \quad b^i \quad c^i)^T \tag{5.21}$$

wobei $\vec{e}^{\,i}$ der Orthonormalvektor der i-ten Approximation ist. Die anschließende Addition der Orthonormalvektoren führt zu folgender Ebenengleichung in Normalform:

$$(\vec{e}^{\,x} + \vec{e}^{\,y} + \vec{e}^{\,z})^T \cdot (\vec{x} \quad \vec{y} \quad \vec{z})^T = 0 \tag{5.22}$$

Für die Berechnung des z-Vektors des Roboterkoordinatensystem im Kamerakoordinatensystem wird diese Darstellung in die Hessesche Normalform durch die Normierung von $\vec{e}^{\,x} + \vec{e}^{\,y} + \vec{e}^{\,z}$ überführt.

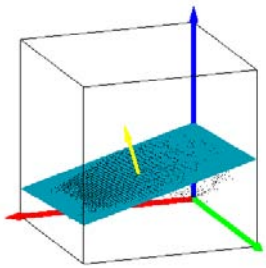


Abbildung 5.3.3: Berechnung des z-Vektors des Roboterkoordinatensystems im Kamerakoordinatensystem

Das Ergebnis, der implementierten Algorithmen zur Bestimmung des z-Vektors des Roboterkoordinatensystems gemessen im Kamerakoordinatensystem, ist in Abbildung 5.3.3 skizziert. Die Punktwolke (schwarz) wird approximiert durch eine Ebene (dunkelblau). Aus der approximierten Normalendarstellung der Ebene kann der Normalenvektor berechnet werden. Der Normalenvektor (gelb) ist gleich dem im Kamerakoordinatensystem berechneten z-Vektor des Roboterkoordinatensystems.

5.3.2 Bestimmung des x-Vektors des Roboterkoordinatensystems

Wie in dem vorherigen Abschnitt erläutert und in Abbildung 5.3.4 zu sehen ist, zeigt der x-Vektor des Roboterkoordinatensystems in Fahrtrichtung des Roboters. Zur Bestimmung des Vektors fährt dieser, mit einer geringen Geschwindigkeit von $v = 0{,}1\ m/s$ und einem eingestellten Lenkwinkel von $\theta = 0\ °$, auf die am Boden positionierte Landmarke zu.

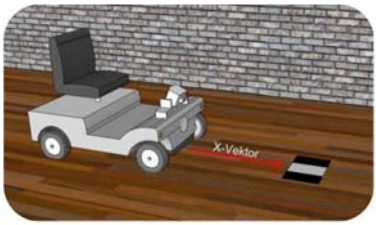


Abbildung 5.3.4: Skizze zur Ermittlung des x-Vektors

Während der Fahrt wird die Position dieser Landmarke mit der PMD-Kamera kontinuierlich ermittelt.

Auf die Algorithmen zur Erkennung und Bestimmung der Landmarkenposition mittels PMD-Kamera soll an dieser Stelle nicht eingegangen werden. Es wird verwiesen auf das Kapitel 5.4.4 „Absolute Positionsbestimmung anhand künstlicher Landmarken“, in dem die Verfahren eingehend behandelt werden.

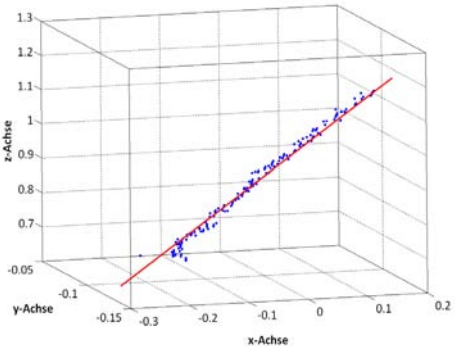


Abbildung 5.3.5: Messwerte zur Ermittlung des x-Vektors in Kamerakoordinaten

Die Bewegung des Roboters führt zu einer linearen Verschiebung der Landmarke im Kamerakoordinatensystem. Dies ist im Graphen auf Abbildung 5.3.5, in dem die gemessenen Landmarkenpositionen eingetragen sind, erkenntlich. Durch das Anpassen einer Geradengleichung, nach der in Kapitel 5.3.1 beschriebenen Methode, kann der Richtungsvektor der Geraden bestimmt werden. Für einen Lenkwinkel $\theta = 0°$ ist der ermittelte Richtungsvektor $\vec{r}$ parallel zum Bewegungsvektor $\vec{b}$ und somit auch parallel zum Vektor $\vec{x}$ des Roboterkoordinatensystem. Es gilt:

$$\vec{r} \,||\, \vec{b} \,||\, \vec{x} \tag{5.23}$$

Die Richtung des x-Vektors im Kamerakoordinatensystem wird berechnet zu:

$$\vec{x}_{Kam} = \begin{cases} +\vec{r}, & r_z \geq 0 \\ -\vec{r}, & r_z < 0 \end{cases} \tag{5.24}$$

5.3.3 Berechnung der Transformationsmatrix

Zur vollständigen Berechnung der Transformationsmatrix, wird durch Bildung des Kreuzprodukts zwischen x- und z-Achse, die Orientierung der y-Achse im Kamerakoordinatensystem berechnet:

$$\vec{y} = \vec{z} \times \vec{x} \tag{5.25}$$

Somit kann die Rotationsmatrix ${}^{Kam}R_{Rob}$ zwischen Kamera- und Roboterkoordinatensystem nach Gl.(5.10) ermittelt werden.

Über die Inverse:

$${}^{Rob}R_{Kam} = {}^{Kam}R_{Rob}^{-1} = {}^{Kam}R_{Rob}^{T} \tag{5.26}$$

lässt sich die gesuchte Rotationsmatrix bestimmen.

Nach Gl.(5.7) setzt sich die Translation zusammen aus einem konstanten, aus dem Aufbau bekannten Anteil und einem Anteil, welcher abhängig vom Neigungswinkel der Kamera berechnet werden kann.

Der Neigungswinkel θ berechnet sich zu:

$$\theta = \beta - 90° \tag{5.27}$$

Hierbei entspricht der Winkel θ der Rotation um die y-Achse des Roboters und kann bei bekannter Rotationmatrix R nach [Krey07] berechnet werden. Die Rotationsmatrix wird wie folgt definiert:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{5.28}$$

Der Drehwinkel um die y-Achse ist:

$$\beta = atan2(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}) \tag{5.29}$$

Für den Fall, dass cos $\beta \neq 0$ ist, berechnen sich die Drehwinkel um die übrigen Achsen nach:

$$a = atan2\left(\frac{r_{21}}{\cos\beta}, \frac{r_{11}}{\cos\beta}\right) \tag{5.30}$$

$$\gamma = atan2\left(\frac{r_{32}}{\cos\beta}, \frac{r_{33}}{\cos\beta}\right) \tag{5.31}$$

Die Betrachtung des Falls $\cos\beta = 0$ kann an dieser Stelle entfallen, dies würde unter Berücksichtigung von Gl.(5.27) nur eintreten, wenn der Kippwinkel θ = 0 wäre. Dies wäre jedoch nur der Fall, wenn die Kamera parallel zum Boden ausgerichtet ist. Somit wäre aber der Betrieb der Kamera nicht mehr möglich, da der Eindeutigkeitsbereich überschritten wird.

Wird Gl.(5.8) in Gl.(5.27) eingesetzt, kann die Translation zwischen Roboter- und Kamerakoordinatensystem berechnet werden.

Zusammenfassend erhalten wir folgendes Ergebnis:

$$ {}^{Rob}T_{Kam} = \begin{bmatrix} & {}^{Kam}R_{Rob}^{T} & & {}^{Rob}\vec{t}_{Kam} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.32} $$

mit:

$$ {}^{Kam}R_{Rob}^{T} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix} \tag{5.33} $$

und

$$ {}^{Rob}\vec{t}_{Kam} = \begin{pmatrix} {}^{Rob}t_{HK,x} + {}^{HK}t_{Kam,x} \cdot cos\theta + {}^{HK}t_{Kam,z} \cdot sin\theta \\ 0 \\ {}^{Rob}t_{HK,z} - {}^{HK}t_{Kam,x} \cdot sin\theta + {}^{HK}t_{Kam,z} \cdot cos\theta \end{pmatrix} \tag{5.34} $$

x, y und z entsprechen den im Kamerakoordinatensystem gemessenen Achsen des Roboterkoordinatensystems, ${}^{Rob}\vec{t}_{HK}$ der konstanten Translation von Roboter-koordinatensystem zur Kamerahalterung und ${}^{HK}\vec{t}_{Kam}$ der Translation von Halterung bis zum Kameraobjektiv. Die y-Koordinaten beider Translationsvektoren sind gleich Null.

5.3.4 Auswertung

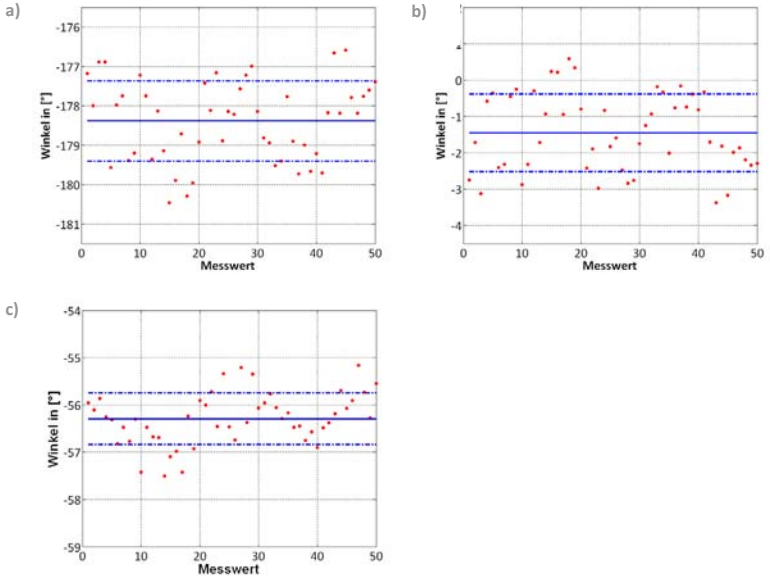


Abbildung 5.3.6: Rotation des Kamerakoordinatensystems zum Roboterkoordinatensystem
a) Rotation um z-Achse b) Rotation um y-Achse c) Rotation um x-Achse

Zur Überprüfung des Kalibrierverfahrens wurden insgesamt 50 Kalibrierungen durchgeführt. Die berechneten Rotationswinkel sind auf den Abbildungen 5.3.6 a bis c für die Winkel α, β, γ dargestellt. Auf der y-Achse der Graphen sind die ermittelten Rotationswinkel sowie deren Mittelwerte und Standardabweichungen eingetragen. Die Aufnahmen wurden mit der im Lynkeus-Projekt entwickelten LynCube durchgeführt. Die Standardabweichungen σ sind:

$$\sigma_\alpha = 1{,}0664°$$

$$\sigma_\beta = 0{,}5449°$$

$$\sigma_\gamma = 1{,}0239°$$

Somit zeigen diese nur eine geringe Abweichung vom Mittelwert des entsprechenden Vektors.

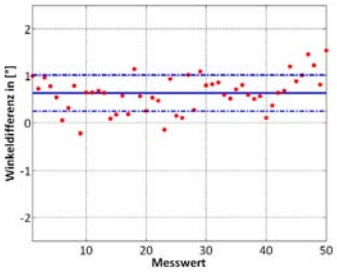


Abbildung 5.3.7: Abweichung des Winkels zwischen x- und z-Achse vom Sollwinkel 90°

Ein weiteres Kriterium für die richtige Funktionsweise der Kalibrierung ist der Winkel zwischen den bei der Kalibrierung berechneten Achsen x und z. Sind die Messungen der einzelnen Vektoren genau, gilt:

$$\arccos\left(\frac{\vec{x}\cdot\vec{z}}{|\vec{x}|\cdot|\vec{z}|}\right)-\frac{\pi}{2}\approx 0 \tag{5.35}$$

Die Winkeldifferenzen für die durchgeführten Kalibrierungen sind für verschiedene Kalibrierungen auf Abbildung 5.3.7 festgehalten. Im Mittel beträgt die Winkeldifferenz $\varphi = 0{,}6314°$, die Standardabweichung ist $\sigma = 0{,}3851°$. Da die Abweichung ungleich Null ist und dies eine unsymmetrische Rotationsmatrix erzeugen würde, wurde der Kalibrieralgorithmus um eine Methode ergänzt, welche die zwei ermittelten Roboterkoordinatenachsen gleichmäßig um die y-Achse soweit zueinander dreht, so dass diese im rechten Winkel zueinander stehen.

Bisher wurden die Ergebnisse der verschiedenen Kalibrierverfahren untereinander verglichen. Somit konnten Aussagen über die Stabilität, also die Zuverlässigkeit der einzelnen Kalibrierungen, getroffen werden. Systematische Fehler im Kalibrierverfahren konnten ausgeschlossen werden. Zur Überprüfung der relativen Genauigkeiten wird der eingestellte Neigungswinkelwinkel der Kamera, bezogen auf den x-Vektor des Roboterkoordinatensystems, aus der Transformationsmatrix berechnet. Dieser entspricht nach Abbildung 5.3.8 dem Winkel zwischen $\vec{x}_{Rob}$ und $\vec{z}_{Kam}$, dem z-Vektor des Kamerakoordinatensystems im Roboterkoordinatensystem.

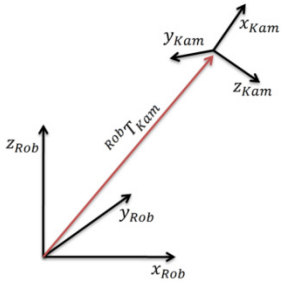


Abbildung 5.3.8: Skizze zu Berechnung der Transformationsmatrix

Mit:

$$ {}^{Rob}R_{Kam} = \left[{}^{Rob}\vec{x}_{Kam} \quad {}^{Rob}\vec{y}_{Kam} \quad {}^{Rob}\vec{z}_{Kam} \right] \tag{5.36} $$

kann der Neigungswinkel berechnet werden zu:

$$ \theta = \arccos\left(\frac{\vec{x}_{Rob} \cdot {}^{Rob}\vec{z}_{Kam}}{|\vec{x}_{Rob}| \cdot |{}^{Rob}\vec{z}_{Kam}|} \right) \tag{5.37} $$

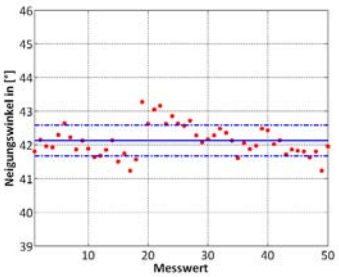


Abbildung 5.3.9: Abweichungen des Kameraneigungswinkel bei verschiedenen Kalibrierungen

Abbildung 5.3.9 zeigt den berechneten Kameraneigungswinkel für mehrere durchgeführte Kalibriervorgänge. Der mittlere Neigungswinkel beträgt:

$$ \bar{\theta} = 42{,}12° $$

Die Standardabweichung hiervon ist:

$$\sigma_\theta = 0{,}46°$$

Durch eine mechanische Messapparatur konnte der bei der Kalibrierung eingestellte Neigungswinkel von:

$$\theta_{mech} = 41{,}5°$$

ermittelt werden. Somit resultiert ein mittlerer Fehler zwischen realem und dem anhand des Kalibrierverfahrens gemessenen Neigungswinkel von:

$$F_\theta = \bar{\theta} - \theta_{mech} = 0{,}62° \tag{5.38}$$

Der bei der Kalibrierung entstandene Fehler entspricht der Summe der Fehler, welche während der Ermittlung der einzelnen Vektoren des Roboterkoordinatensystems auftreten. Kamerabasierte Fehler resultieren aus Messungenauigkeiten während der Ermittlung der Landmarkenpose. Roboterbasierte Fehler begründen sich aus einem fehlerhaften Abfahren einer geraden Bahn in Richtung der Landmarke. Der Lenkwinkel des Roboters ist nicht gleich Null, sondern schwankt aufgrund der Lenkwinkelregelung leicht um den Nullpunkt. In Kombination mit geringfügig unterschiedlichen Radradien, weist die abgefahrene Trajektorie eine leichte Krümmung auf, das zu einer fehlerhaften Bestimmung des x-Vektors führt. Fehler bei der Ermittlung des z-Vektors ergeben sich zum einen durch verrauschte Kameramesswerte, wie zum Beispiel Entfernungsfehler wegen unterschiedlich stark reflektierender Bereiche im Sichtbereich der Kamera, sowie aus leichten Krümmungen der aufgenommenen Ebene an den Bildrändern. Ist die z-Achse des Roboterkoordinatensystems nicht senkrecht zum Boden, können weitere Fehler auftreten. Dies kann der Fall sein, wenn der Fahrweg leichte Unebenheiten aufweist oder wenn die Räder des Roboters leicht von einander unterschiedliche Radien besitzen, welche durch variierende Luftdrücke in den Rädern resultieren können.

5.4 Selbstlokalisierung

Selbstlokalisierung meint die autonome Bestimmung von Position und Orientierung einer mobilen Roboterplattform mittels integrierter Sensoren. Hierbei wird zwischen relativer und absoluter Poseschätzung unterschieden. Relative Poseschätzung entspricht der Bestimmung der Position und Orientierung des fahrerlosen Transportsystems (FTS) beziehungsweise des mobilen Roboters, hinsichtlich eines Bezugspunkts. Üblicherweise ist dies der Startpunkt der Roboterbewegung. Relative Positionsschätzung wird bei der Navigation zur Exploration von unbekannten

Umgebungen durchgeführt oder zur Unterstützung der Navigation in bekannter Umgebung, sollte eine absolute Poseschätzung nicht möglich sein. Relative Poseschätzung kann durch die Ermittlung von Bewegungsvektoren anhand von Radencodern oder aus 3D-Bilddaten einer am Roboter befindlichen optischen 3D-Messeinrichtung erfolgen.

Absolute Poseschätzung befasst sich mit der Berechnung von Position und Orientierung des mobilen Roboters bezogen auf ein Umweltkoordinatensystem. Dies kann zum Beispiel das Koordinatensystem einer Umgebungskarte sein. Absolute Poseschätzung erfolgt üblicherweise über die Ermittlung der Position von natürlichen oder künstlichen Landmarken, deren Positionen in einer Umgebungskarte verzeichnet sind. Natürliche Landmarken sind innerhalb von aufgenommenen Bildern sind leicht zu extrahierende Merkmale, die mit wenig Aufwand in der vorhandenen Karte registriert werden können. Ist die Position dieser Merkmale in der Karte bekannt, kann auf die Position des Roboters zurückgeschlossen werden. Künstliche Landmarken werden nach dem gleichen Prinzip eingesetzt. Es handelt sich um leicht zu detektierende Marker die mit einer festen, bekannten Position innerhalb des zu befahrenden Gebiets positioniert werden.

Für die relative Positionsschätzung stehen der mobilen Plattform zwei Sensortypen zur Verfügung. Zum einen sind dies zwei in den Hinterachsen verbaute Radencoder, zum anderen können mittels ICP-Algorithmus die Bewegungsvektoren, aus hintereinander aufgenommenen 3D-Bilddatensätzen, berechnet werden. Hieraus kann die Bewegung des Roboters geschätzt werden. Eine absolute Positionsbestimmung, welche später zur Ermittlung der Startposition und -Orientierung des Roboters sowie zur Positionskorrektur dienen soll, findet durch die Erkennung und Positionsbestimmung von künstlichen Landmarken, ebenfalls unter Verwendung der verbauten PMD-Kamera, statt.

In den folgenden Abschnitten werden die implementierten Verfahren zur relativen und absoluten Lokalisierung betrachtet. Die gewonnenen Ergebnisse werden aufgeführt und diskutiert. Begonnen wird zunächst mit den Verfahren zur relativen Lokalisierung des Roboters, über die Poseschätzung mit Radencodern und über die Berechnung von Bewegungsvektoren aus 3D-PMD-Bilddaten unter Anwendung des ICP-Algorithmus. Zur Erhöhung der Genauigkeit des Selbstlokalisierungssystems werden die einzelnen Verfahren mittels Kalman-Filter fusioniert. Nachfolgend wird auf die absolute Selbstlokalisierung des Roboters anhand von künstlichen Landmarken eingegangen.

Dies dient zum einen der Ermittlung der Roboterstartposition und zum anderen der Korrektur der durch die relative Selbstlokalisierung ermittelten Roboterpose.

5.4.1 Poseschätzung mit Radencodern

Der Gebrauch von Radencodern zur Bestimmung der Pose eines mobilen Roboters ist ein allgemein übliches Verfahren und kann in der Literatur [Siegwart04] nachgelesen werden. Aus diesem Grund soll hier der Vollständigkeit wegen kurz eingegangen werden.

Zur Ermittlung der Position und Orientierung relativ zu einem Bezugspunkt werden zwei Radencoder benutzt, die an beiden Rädern der hinteren Achse verbaut sind.

Mit den Radencodern kann der zurückgelegte Weg S des linken (L) und des rechten (R) Rads ermittelt werden. Dieser berechnet sich zu

$$S_{L/R} = c_k \cdot N_{L/R} \tag{5.39}$$

Wobei c_k der Umrechnungsfaktor zwischen der Anzahl, der vom Radencoder generierten Impulse N und dem Weg S ist. Dieser wird bestimmt über den Raddurchmesser D_l und der Auflösung des Encoders pro Radumdrehung C_e.

$$c_k = \pi D_l / C_e \tag{5.40}$$

Die lineare Verschiebung des Achsmittelpunkts beträgt somit:

$$\Delta S_i = (S_L + S_R)/2 \tag{5.41}$$

Die Änderung der Orientierung berechnet sich aus:

$$\Delta\theta = (S_L + S_R)/d \tag{5.42}$$

d entspricht dem Abstand der Räder. Aus der Verschiebung des Achsmittelpunkts Gl.(5.41) und der Orientierungsänderung Gl.(5.42) lässt sich die diskrete Zustandsgleichung zur Ermittlung des zurückgelegten Wegs in x/y-Koordinaten herleiten:

$$x(t+1) = x(t) + \Delta S_t \cdot \cos\theta_t \tag{5.43}$$

$$y(t+1) = y(t) + \Delta S_t \cdot \sin\theta_t \tag{5.44}$$

$$\theta(t+1) = \theta(t) + \Delta\theta_t \tag{5.45}$$

Der Einsatz von Radencodern zur Lokalisierung eignet sich nur bedingt zum Einsatz als einziges Lokalisiersystem. Aufgrund von Schlupf, also durch durchdrehende Räder beim Beschleunigen, Drift bei Kurvenfahrten, unebenen Böden oder durch leicht unterschiedliche Raddurchmesser können bei der Messung Fehler entstehen.

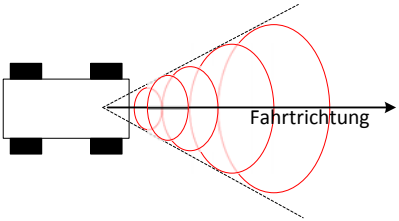


Abbildung 5.4.1: elliptische Fehlerfortpflanzung

Die ermittelte Position und Orientierung wird nach Gl.(5.43) bis Gl.(5.45) über das Aufaddieren der einzelnen Positionen bestimmt. Hierbei zeigt sich, dass der entstehende Fehler im ungünstigsten Fall fortwährend hinzuaddiert wird. Dieser Sachverhalt wird in Abbildung 5.4.1 verdeutlicht. Hier sind die entstehenden Fehlerellipsen für zunehmende Weglänge beispielhaft dargestellt.

5.4.2 Poseschätzung mittels ICP-Algorithmus unter Verwendung von PMD-Daten

Zur Poseschätzung des Roboters anhand der PMD-Daten müssen die Bewegungsvektoren zwischen aufeinanderfolgenden 3D-Datensätzen ermittelt werden. Durch Transformation des Bewegungsvektors in das Roboterkoordinatensystem kann auf die Verschiebung des Roboters geschlossen werden. Für die Berechnung des Bewegungsvektors stehen verschiedene Verfahren zur Verfügung. Verfahren die sich zur Ermittlung des Bewegungsvektors eignen sind zum Beispiel die Berechnung des optischen Flusses [Berthold81], die Berechnung des Bewegungsvektors über die Korrelation der Bilder [Jaehne05] oder der ICP-Algorithmus [Besl92]. Aufgrund der geringen Auflösung des anfangs verwendeten Kameratyps mit nur 64x50 Pixeln, stellte sich die Ermittlung des Bewegungsvektors mit dem ICP-Algorithmus, als geeignetes Verfahren heraus, welches im Folgenden näher betrachtet wird. Dieses

Verfahren wurde später auf die Benutzung des neueren PMD-Sensors mit 200x200 Pixeln angepasst.

Der ICP-Algorithmus

Es seien 2 Punktwolken, P und M, mit geringer Verschiebung zwischen den Aufnahmezeitpunkten gegeben. Der ICP-Algorithmus dient der Berechnung von Translation und Rotation zwischen den Punktwolken, indem von der einen Punktwolke $P = \{p_i\}_{i=1\dots N_p}$ die Rotation zur anderen Punktwolke $M = \{m_i\}_{i=1\dots N_m}$ iterativ berechnet wird. Folgende 3 Schritte sind hierzu notwendig:

1. Zu jedem Punkt p_i wird der nächste Nachbar in M gesucht
2. Durch quadratische Minimierung der Fehler zwischen den korrespondierenden Punkten wird die Transformation von P zu M berechnet
3. Die berechnete Transformationsmatrix wird auf die Punktwolke P angewandt

Algorithmus 5.1: Vereinfachter ICP-Algorithmus

Die drei Schritte werden iterativ solange angewandt bis die Summe der kleinsten Fehlerquadrate eine definierte Grenze unterschreitet oder eine vorher festgelegte Anzahl an maximalen Iterationen erreicht ist. Der ICP-Algorithmus ist ein nicht linearer Suchalgorithmus der immer gegen ein lokales Minimum konvergiert. Dieses lokale Minimum entspricht allerdings nicht immer dem globalen Minimum des quadratischen Fehlers.

Der ursprüngliche ICP-Algorithmus von Besl & McKay [Besl92] trifft die Annahme, dass die Anzahl der Punkte jeder Punktwolke gleich ist. Hier gilt $N_p = N_m$ und jeder Punkt p_i der Punktwolke P besitzt einen korrespondierenden Punkt m_i in der Punktwolke M. Dies ist beim Matching von zwei unterschiedlichen Kameraaufnahmen aufgrund der Kamerabewegung und des Rauschens der PMD-Daten nicht der Fall. Deshalb wurde hierfür ein modifizierter Algorithmus verwendet. Diese implementierte Variante des ICP-Algorithmus wird als Trimmed ICP Algorithmus (TrICP) bezeichnet [Chet02].

Beim TrICP-Algorithmus wird der Faktor ξ eingefügt. Dieser Faktor bezeichnet die minimale Überlappung zwischen den beiden Punktmengen, so dass die Anzahl der Punkte P die einen korrespondierenden Partner in M besitzen, $N_{p0} = \xi * N_p$ ist. Für

die Punkte N_{p0} muss im Folgenden die Rotation und Translation zu der Punktmenge M berechnet werden.

Die zu minimierende Kostenfunktion berechnet sich wie folgt:

$$d(R,t) = \frac{1}{N_{p_0}} \sum_{j=1}^{N_{p_0}} \left\| \vec{m}_j - R \cdot \vec{p}_j - \vec{t} \right\|^2 \tag{5.46}$$

wobei $\vec{m}_j$ und $\vec{p}_j$ die jeweiligen zueinander korrespondierenden Punkte sind. Zur Berechnung des Rotations- und Translationsanteils werden beide Teile getrennt voneinander betrachtet. Zuerst wird die Rotation zwischen den beiden Punktwolken angenähert, indem die Masseschwerpunkte $\vec{c}_m$ und $\vec{c}_p$ beider Punktmengen bestimmt werden.

$$\vec{c}_m = \frac{1}{N_{p_0}} \sum_{i=1}^{N_{p_0}} \vec{m}_i \qquad\qquad \vec{c}_p = \frac{1}{N_{p_0}} \sum_{i=1}^{N_{p_0}} \vec{p}_i$$

Die um den Masseschwerpunkt verschobenen Punktwolken sind:

$$M'\colon\ \vec{m}_i' = \vec{m}_i - \vec{c}_m \qquad \text{und} \qquad P'\colon\ \vec{p}_i' = \vec{p}_i - \vec{c}_p$$

Die Rotation, welche die Kostenfunktion Gl.(5.46) minimiert, kann über die Korrelationsmatrix der korrespondierenden Punkte, der um den Masseschwerpunkt verschobenen Punktwolken, geschätzt werden. Die Korrelationsmatrix H berechnet sich folgendermaßen

$$H = \sum_{i=1}^{N_{p0}} \vec{p}_i' \cdot \vec{m}_i'^T = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{5.47}$$

wobei die Matrixelemente zu

$$S_{xx} = \sum_{i=1}^{N_{po}} m_{ix}' \cdot p_{ix}'$$

$$S_{xy} = \sum_{i=1}^{N_{po}} m_{ix}' \cdot p_{iy}'$$

und so weiter berechnet werden. Über die Berechnung der Eigenwerte in Gl.(5.47) kann die Rotationsmatrix zwischen den Bildern bestimmt werden. In der Literatur stehen dazu verschiedene Verfahren zur Verfügung. Zu erwähnen ist die Anwendung von Quaternionen [Kuipers99] oder die Singularitätswertzerlegung [Deuflhard00] (SVD – Singular Value Decomposition). Für die im Folgenden beschriebenen Algorithmen wurde die Singularitätswertzerlegung verwendet. Hierbei berechnet sich die Rotationsmatrix R aus

$$R = V \cdot U^T \tag{5.48}$$

wobei V und U Komponenten aus der SVD sind und sich aus der Gleichung

$$H = U\Lambda V^T \tag{5.49}$$

bestimmen lassen. Nach der Berechnung der Rotationsmatrix kann die Translation $\vec{t}$ über

$$\vec{t} = \vec{c}_m - R \cdot \vec{c}_p \tag{5.50}$$

ermittelt werden.

Zusammenfassend gestaltet sich der ICP-Algorithmus bisher wie folgt:

ICP-Algorithmus

1. Suche der nächsten Nachbarn von P in M durch die Suche der Punkte mit geringster Entfernung
2. Bestimmung des Überlappungsfaktors ξ durch Betrachtung der Entfernungen
3. Bestimmung der Korrelationsmatrix zwischen beiden Punktwolken
4. Berechnung der Rotationsmatrix durch Anwendung des SVD-Algorithmus
5. Berechnung des Translationsvektors $\vec{t}$
6. Anwendung der Rotation R und Translation $\vec{t}$ auf die Punktmenge P
7. Wiederhole 1-6 solange bis die Summe der kleinsten Fehlerquadrate zwischen den Punktwolken eine definierte Schranke unterschreitet oder die festgelegte Anzahl an maximalen Iterationen erreicht ist

Algorithmus 5.2: Ablauf des ICP-Algorithmus

Zur reproduzierbaren Überprüfung der Funktionsweise des entwickelten ICP-Algorithmus wurden zwei 3D-Punktwolken erstellt und mit einer definierten Transformationsmatrix verschoben. Diese Punktwolken, welche in dem Graphen auf Abbildung 5.4.2a eingetragen sind, werden mittels ICP-Algorithmus abgeglichen.

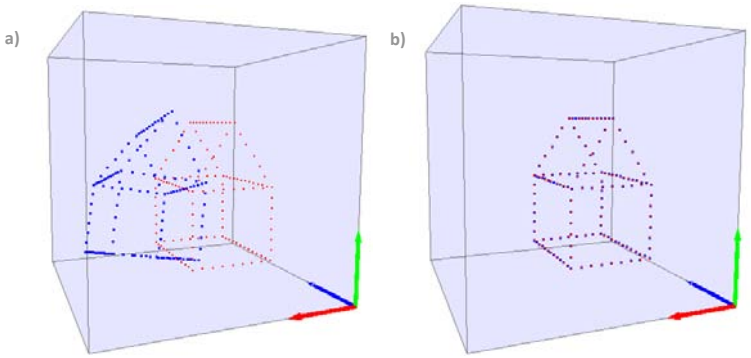


Abbildung 5.4.2: Punktwolken zur Überprüfung der Funktionsweise des ICP a) vor ICP b) nach ICP

Das Ergebnis der Berechnung, auf Abbildung 5.4.2b dargestellt, weist nach der Verschiebung um die mit dem ICP-Algorithmus berechnete Matrix keine Abweichung von der originalen Punktwolke auf. Die richtige Funktionsweise des implementierten Algorithmus konnte somit nachgewiesen werden. Die Punktwolken bestehen aus je 138 Punkten. Hierfür benötigt der Algorithmus eine Berechnungszeit von $t = 175ms$. Aufgrund der Abstandberechnungen, welche zur Suche zueinander gehörender Punkte benötigt werden, weist der ICP-Algorithmus eine starke Zunahme der Berechnungszeit bei zunehmender Anzahl von Punkten auf.

Es ist vorstellbar, dass der oben beschriebene Algorithmus, welcher jeden Punkt der Punktwolke P jeweils mit allen Punkten der Punktwolke M auf den geringsten Abstand überprüft, schon bei einer geringen Auflösung der PMD-Kamera von 64x50 Pixeln eine zu hohe Berechnungszeit erwarten lässt. Der Betrieb der Kamera zur Bewegungs-berechnung erscheint deswegen als nicht sinnvoll.

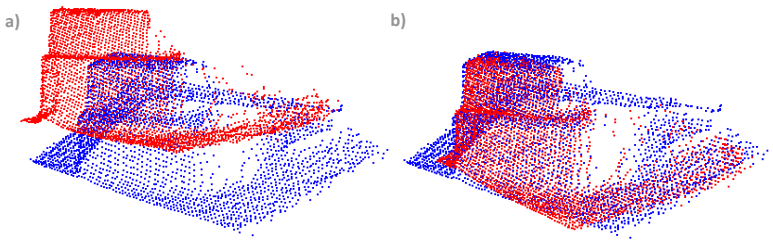


Abbildung 5.4.3: Zwei zueinander verschobene PMD-Aufnahmen a) vor Anwendung des ICP b) nach Anwendung des ICP

Zur Überprüfung dieser Annahme wurden zwei 3D-Datensätze mit einer PMD-Kamera aus zwei verschiedenen Positionen aufgenommen, welche in Abbildung 5.4.3a dargestellt sind. Jede dieser Punktwolken besteht aus 64x48 dreidimensionalen Datenpunkten, welche mittels ICP-Algorithmus abgeglichen werden. Das Ergebnis, erhalten durch die Verschiebung des zweiten 3D-Datenssatzes um die berechnete Transformationsmatrix, ist auf Abbildung 5.4.3b dargestellt. Für die Berechnung der Verschiebung mit dem ICP-Algorithmus wurde eine Dauer von $t = 7{,}6s$ bei $n = 20$ Iterationen benötigt, welches die obige Annahme bestätigt. Um jedoch eine Online-Bewegungsbestimmung aus den PMD-Daten während der Roboterbewegung zu ermöglichen, muss der Rechenaufwand und somit die für die Berechnung verwendeten Daten reduziert werden. Dies kann durch die Anwendung des Sobel- oder des Canny-Filters (siehe Anhang), die sowohl auf die Grauwerte und auch auf die Distanzwerte der einzelnen Aufnahmen angewendet werden, erreicht werden. Hierbei werden alle markanten Punkte, wie Kanten- und Eckpunkte, extrahiert. Die extrahierten 3D-Datensätze ermöglichen eine schnellere Berechnung der Transformationsmatrix zwischen den Punktwolken.

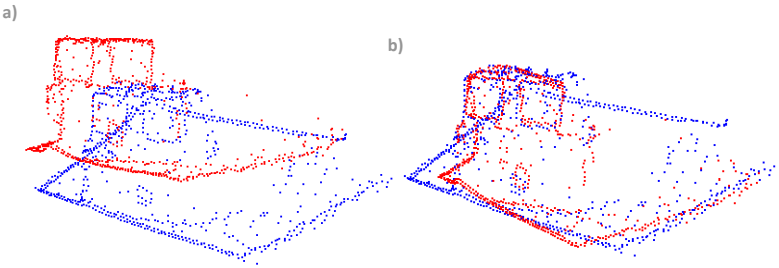


Abbildung 5.4.4: Abgleich von zwei Punktwolken mit Canny-Filter a) vor b) nach Anwendung des ICP

Durch Anwendung der Filter auf die 3D-Datensätze aus Abbildung 5.4.3 können die 3D-Punkte, siehe Abbildung 5.4.4, stark reduziert werden. Die Datensätze bestehen aus 381 beziehungsweise aus 349 Datenpunkten. Dies verringert die Berechnungszeit bei gleichbleibender Anzahl an Iterationen auf $t = 243\,ms$. Somit ist ein deutlicher Zuwachs der Geschwindigkeit bei vorheriger Extraktion der markanten Punkte zu erkennen. Allerdings zeigte sich ebenso eine Zunahme der Ungenauigkeit des Matchingvorgangs, gerade bei nur leicht strukturierten Aufnahmen mit größeren Flächen. Dieses Problem konnte, jedoch durch die Hinzunahme von zufällig ausgewählten Flächenpunkten unter leichter Erhöhung der Berechnungszeit, gelöst werden. Für die Berechnung des ICP-Algorithmus zur bildgestützten Positionsreferenzierung des mobilen Roboters ist dieses Ergebnis jedoch noch nicht zufriedenstellend. Bisher beansprucht die Berechnung aller Abstände der Punkte der Punktwolke M zu denen der Punktwolke P den größten Teil der Berechnungsdauer.

Aus diesem Grund wird der implementierte Standardalgorithmus, zur Suche von korrespondierenden Punktpaaren, durch die Abstandsermittlung mit einem kD-Baum ersetzt. Dies ist eine schnellere Methode zur Abstandsberechnung zwischen Punkten der beiden Datensätze.[1]

Der kD-Baum

Der kD-Baum ist ein binärer Suchbaum der Dimension k, welcher eine Punktemenge der Dimension $\mathbb{R}^k$ repräsentiert. Nach [Bentley75] eignet sich der kD-Baum für effektive Suchanfragen bei „exact"-Match, „partial"-Match und bei Bereichsanfragen zum Beispiel bei der Suche nach dem nächsten Nachbarn oder nach allen Nachbarn in einem bestimmten Radius.

Es wird zwischen dem homogenen und dem heterogenen kD-Baum unterschieden. Während bei einem homogenen Baum die Daten sowohl in den Knoten als auch in den Blättern eingetragen werden, befinden sich die Daten des heterogenen kD-Baums ausnahmslos in den Blättern. In den Knoten befinden sich lediglich Verweise (Schlüssel) auf die Datensätze. Folglich besitzt jeder Knoten je einen Zeiger auf die zwei Child-Knoten und einen auf den parent-Knoten. Jeder Knoten repräsentiert eine Aufteilung der Punktmenge durch Mittelwertbildung der k-ten Dimension in zwei nahezu gleich große Punktemengen.

[1] Bisher wurde das Standardverfahren des ICP-Algorithmus behandelt, welches im folgenden durch eigene Kombinationen bekannter Algorithmen zu einem echtzeitfähigen Verfahren zur Berechnung von Bildvektoren aufeinander folgender PMD-Bilder weiterentwickelt wird.

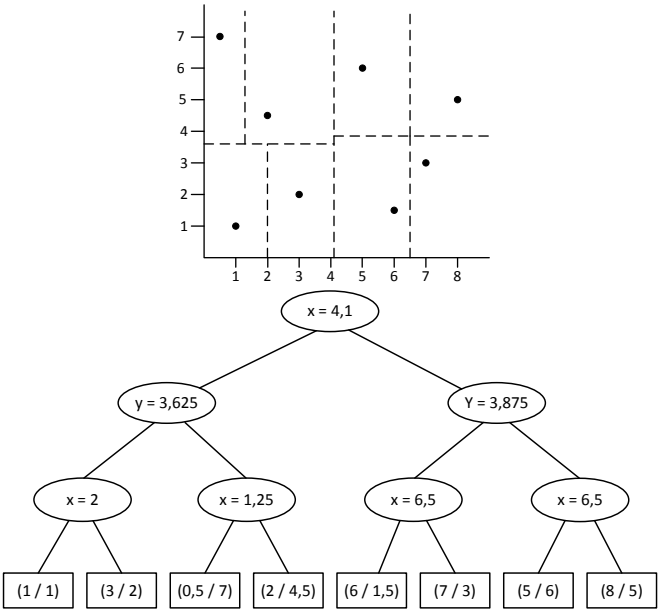


Abbildung 5.4.5: Beispiel für den Aufbau eines kD-Baums

Der implementierte Algorithmus zum Aufbau des kD-Baums gestaltet sich wie folgt:

Aufbau des kD-Baums

1. Initialisierung
2. Berechne Aufteilung durch eine Hyper-Ebene durch Bildung der Mittelwerte der k-ten Dimension, beginne mit der ersten Dimension
3. Schreibe Schlüssel in Knoten
4. Initialisierung der Child-Knoten
5. Gehe zu linkem Child-Knoten
6. Weiter mit 2, solange bis Anzahl der Punkte in Knoten gleich 1 ist
7. Trage Punkt in das Blatt ein
8. Gehe zu parent-Knoten
9. Gehe zu rechtem Child-Knoten
    a. wenn Anzahl der Punkte im Knoten größer 1 ist gehe zu 2.
    b. wenn Anzahl der Punkte im Knoten gleich 1 ist gehe zu 7. solange bis die Wurzel des Baums zweimal besucht wurde

Algorithmus 5.3: Aufbau eines kD-Baums

Der kD-Baum, wie dieser in Abbildung 5.4.5 beispielhaft skizziert ist, wird mit dem beschriebenen Algorithmus erstellt. Bei Verwendung des kD-Baums als Suchbaum zum Auffinden korrespondierender Punkte besitzt der Baum eine 3-dimensionale Struktur. Der hierfür benötigte Speicherbedarf liegt bei $O(n)$, wobei n der Anzahl der Blätter entspricht. Die Zeit für die Erstellung des Baumes beträgt $O(n\ log(n))$.

Nach der Erstellung des Baums gilt es ein geeignetes Matching-Verfahren zur Suche von korrespondierenden Punkten innerhalb der Punktwolke P‘ und der in den KD-Baum eingetragenen, Punktwolke M‘ zu implementieren. Die Suche nach dem nächsten Nachbarn erfolgt wie in Abbildung 5.4 dargestellt ist.

Suche nach nächsten Nachbarn im kD-Baum

1. Punkt p_i bis zum untersten Blatt durchsickern
2. Berechne Distanz zwischen Blatt und Punkt p_i
3. Ball-within-Bound-Test (BwB):
   True → nächster Nachbar ist gefunden
   False → weiter mit Punkt 4
4. Zeiger auf Parent-Knoten
5. BwB: True→ gehe zu Punkt 4
6. Gehe zu rechtem Child-Knoten
7. Gehe zu Punkt 1 solange bis oberster Knoten 2x erreicht wurde.
8. Wähle Knoten mit dem geringsten Abstand

Algorithmus 5.4: Suche nach nächsten Nachbarn im kD-Baum

Bei der Suche nach dem nächsten Nachbarn wird zunächst der Punkt p_i , abhängig von den in den Knoten definierten Grenzen, bis zum untersten Blatt durchgesickert. Im Blatt angekommen wird die Distanz d_{min} zu dem dort gespeicherten Punkt m berechnet und der Ball-within-Bound-Test (BwB) [Görz03] durchgeführt. Dieser Test überprüft ob der Kreis „Ball“, der gebildet wird durch den Mittelpunkt, den Punkt p_i, und dem Radius d_{min}, innerhalb der von allen parent-Knoten bis zur Wurzel definierten Grenzen „Bounds“ liegt. Ist dies der Fall, ist der gefundene Punkt der mit dem geringsten Abstand zu p_i. Ist dies jedoch nicht der Fall, wird der Zeiger auf den parent-Knoten gesetzt. Dies wird solange durchgeführt bis der BwB-Test negativ ist. Anschließend wird der Zeiger auf den rechten Knoten gesetzt und es wird mit dem versickern in dem entsprechenden Teilbaum fortgefahren. Der Abstand des Punktes m_i zum innerhalb des Blattes liegenden Punkt p_i wird berechnet. Ist dieser kleiner als d_{min}, werden die korrespondierenden Punkte gespeichert. Diese Schritte werden

solange ausgeführt bis der Zeiger wieder auf die Wurzel des Baumes zeigt. Der Punkt mit dem geringsten Abstand wird gewählt.

Der ICP-Algorithmus mit KD-Baum

In den folgenden Abbildungen sind je zwei 3D-Datensätze dargestellt, welche jeweils eine Szene aus zwei unterschiedlichen Kamerapositionen wiedergeben. Die Translation und die Orientierungsänderung der Kamera zwischen den beiden Aufnahmen wurden mittels der verschiedenen Implementierungsstufen des ICP-Algorithmus berechnet. In Tabelle 5.1 werden die benötigten Berechnungszeiten gegenübergestellt und verglichen.

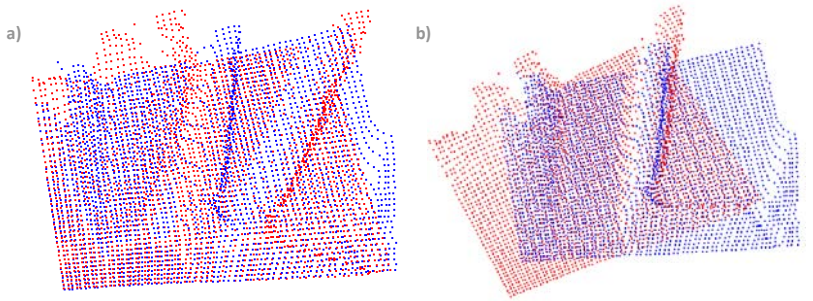


Abbildung 5.4.6: Zwei zueinander verschobene 3D-Datensätze a) vor Anwendung des ICP und b) nach Anwendung des ICP-Algorithmus

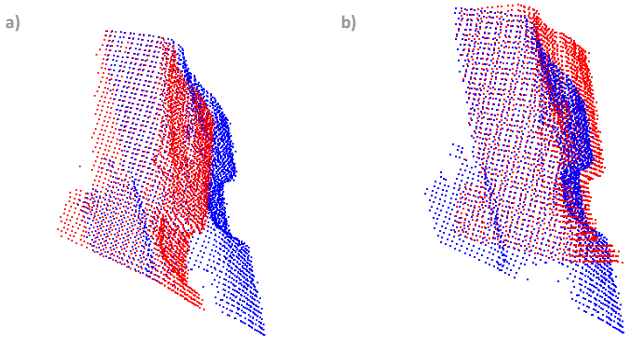


Abbildung 5.4.7: Zueinander verschobene 3D-Datensätze a) vor Anwendung des ICP und b) nach Anwendung des ICP-Algorithmus

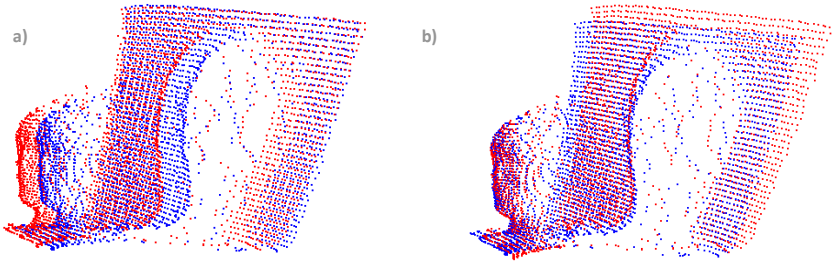


Abbildung 5.4.8: Zueinander verschobene 3D-Datensätze a) vor Anwendung des ICP und b) nach Anwendung des ICP-Algorithmus

	Anzahl 3D-Punkte	Geschwindigkeit ohne kD-Baum	Geschwindigkeit mit kD-Baum
Abbildung 5.4.6	434 / 395 3072 / 3072	231ms 7,68s	**60ms** 468ms
Abbildung 5.4.7	532 / 469 3072 / 3072	260ms 7,82s	**98ms** 537ms
Abbildung 5.4.8	564 / 569 3072 / 3072	312ms 7,79s	**91ms** 514ms

Tabelle 5.1: Vergleich der Implementierungsstufen des ICP-Algorithmus

Die Tabelle 5.1 zeigt die Berechnungszeiten für die in Abbildung 5.4.6 bis Abbildung 5.4.8 dargestellten 3D Punktwolken für die Berechnung mit und ohne kD-Baum. Des Weiteren wurden beide Verfahren, mit und ohne Extraktion der markanten Punkte unter Einbeziehung des Canny-Filters, angewandt. Zu erkennen ist ein deutlicher Geschwindigkeitsgewinn durch Reduzierung der 3D-Daten, wie dies in dem obigen Abschnitt bereits evaluiert wurde. Gleichzeitig ist jedoch zu sehen, dass bei einer Zunahme der für die Berechnung verwendeten Punkte die prozentuale Erhöhung der benötigten Rechenzeit bei Gebrauch des kD-Baums wesentlich geringer ist, als bei der herkömmlichen Berechnungsmethode ohne kD-Baum. Während die Berechnungszeit der Transformationsmatrix zwischen zwei mit der PMDVision S3 aufgenommenen Punktwolken ohne kD-Baum noch zwischen 7 und 8 Sekunden liegt, werden bei Verwendung der kD-Baumsuche lediglich $450ms$ bis $550ms$ benötigt. Bei vorheriger Reduzierung der 3D-Datenpunkte kann die benötigte Rechenzeit auf unter 100ms verringert werden. Die genaue Berechnungszeit ist jedoch abhängig von der Verschiebung sowie der Struktur der Aufnahmen und der hiervon abhängigen Anzahl an benötigten Iterationsschritten.

Es wurde ein Algorithmus entwickelt, welcher die Berechnung von Bewegungsvektoren zwischen aufeinander folgenden PMD-Aufnahmen mit einer Framerate von über 10fps ermöglicht. Auf diese Weise kann eine fortlaufende Positions- und Orientierungsbestimmung der Roboterplattform zur Korrektur der Radencoderdaten erfolgen.

Auswertung des implementierten ICP-Algorithmus

Zur Überprüfung der Funktionsweise des ICP-Algorithmus wurde die LynCube, mit einer lateralen Auflösung von 200x200 Pixeln, an dem Endeffektor eines Industrieroboters (KUKA KR3) befestigt und mit dem vom Deutschen Luft- und Raumfahrtzentrum (DLR) frei erhältlichen Kalibrierprogramm CALDE/CALAB [Stobl06] kalibriert, so dass die Bewegung der Kamera über die Positionsdaten des Roboters ermittelt werden kann. Anschließend wurden insgesamt sechs verschiedene Positionen angefahren und aus jeder Position ein 3D-Datensatz aufgenommen.

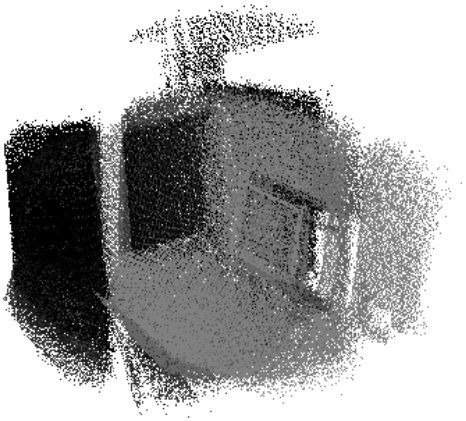


Abbildung 5.4.9: Mit ICP-Algorithmus überlagerte 3D-Punktwolken

Die Bewegungsvektoren zwischen den aufgenommenen Datensätzen wurden mit dem ICP-Algorithmus berechnet und so die Transformationsmatrix zwischen den einzelnen Aufnahmen bestimmt. Abbildung 5.4.9 zeigt das Ergebnis des Matching-Vorgangs. Zur Unterscheidung der einzelnen 3D-Punktwolken sind diese in verschiedenen Graustufen dargestellt. Bei der lateralen Auflösung von 200x200 Pixeln besteht die Punktwolke, unter Vernachlässigung der nicht dargestellten über- oder unterbelichteten Pixel, aus circa 240.000 Pixeln.

	X	Y	Z	α	β	γ
ICP	0,087	-0.339	-0,087	4,19	3,51	4,58
Roboter	0,08	-0,35	-0,10	5	0	7,5
Abweichung	0,007	0,011	0,013	-0,81	3,51	-2,92

Tabelle 5.2: Vergleich der Translation und Rotation von Kamera und Roboter

Zur Überprüfung der Genauigkeit bei der Berechnung des Bildvektors zeigt Tabelle 5.2 die Verschiebung der Kamera zwischen der ersten und letzten Aufnahme, zum einen bestimmt durch die Berechnung der Bildvektoren unter Verwendung des ICP-Algorithmus und zum anderen durch die Robotersteuerung. Aus der Gegenüberstellung der Messwerte resultiert ein mittlerer quadratischer Fehler von:

$$f_{RMS} = \sqrt{f_x^2 + f_y^2 + f_z^2} = 0{,}0184cm$$

Der mittlere quadratische Winkelfehler beträgt:

$$\delta_{RMS} = \sqrt{f_\alpha^2 + f_\beta^2 + f_\gamma^2} = 4{,}49°$$

Das System weist bei obiger Messung folglich eine translatorische Abweichung von 4,9% auf. Somit kann auf die Funktionsfähigkeit des Systems zur Positionsbestimmung des mobilen Roboters über die Bewegungsvektoren in den PMD-Aufnahmen geschlossen werden. Allerdings kann keine konkrete Aussage über die tatsächliche Positionsabweichung getroffen werden, da diese von einer Vielzahl an Faktoren abhängig ist, auf welche im Folgenden näher eingegangen wird.

a) b)

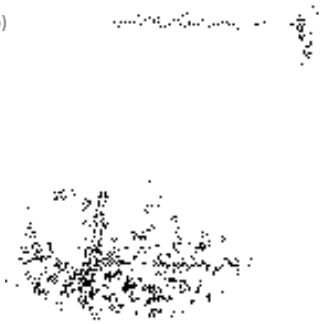
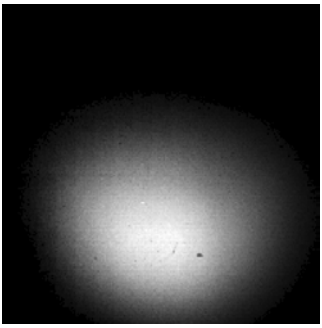


Abbildung 5.4.10: Beispiel zur Filterung von Pixeln vor der Verwendung des ICP-Algorithmus (Die Aufnahme zeigt den Fahrweg des Roboters und eine Wand)

Abbildung 5.4.10a zeigt ein typisches PMD-Bild wie dieses im Betrieb des mobilen Roboters entstehen kann. Hier lassen sich ohne weiteres Punkte für den ICP-Algorithmus extrahieren, die einen guten Abgleich mit dem darauffolgenden Bild ermöglichen. Die extrahierten 3D-Bildpunkte sind in Abbildung 5.4.10b dargestellt.

a) b)



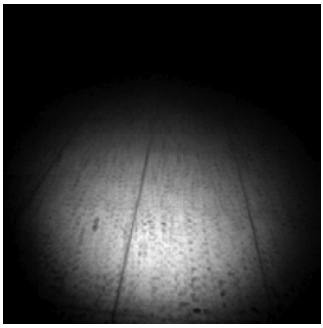


Abbildung 5.4.11: PMD-Aufnahmen während des Betriebs des mobilen Roboters a) keine Struktur b) verwischen der Struktur durch zu hohe Geschwindigkeit

Problematisch gestaltet sich jedoch die Verwendung des ICP-Algorithmus in den in Abbildung 5.4.11a und b dargestellten Szenen. Der Boden in Abbildung 5.4.11a besitzt keinerlei Struktur und aufgrund des Neigungswinkels der PMD-Kamera in Kombination mit der beschränkten Reichweite können keinerlei Strukturen erkannt werden, welche die Grundlage einer erfolgreichen Anwendung des ICP-Algorithmus bilden. In diesem Fall kann dieser zu keinem Ergebnis führen. Gleiches gilt für den in Abbildung 5.4.11b dargestellten Fall. Aufgrund der hohen Geschwindigkeit des mobilen Roboters können keine Strukturen im PMD-Bild erkannt werden. Durch das Vermindern der

Integrationszeit ließe sich zwar ein schärferes Bild erzeugen, dies würde allerdings gleichzeitig das Rauschen der 3D-Werte erhöhen, so dass die Berechnung der Transformationsmatrix ebenfalls fehlerhaft wäre.

Das System unterliegt folglich Fehlern, die aufgrund geringfügig ausgeprägter oder fehlender Struktur im Bildbereich, wie zum Beispiel wenigen oder nicht vorhandenen markanten Kanten und Ecken, resultieren. Des Weiteren führt eine hohe Geschwindigkeit des Roboters zum einen zu verschwommenen Bilddaten, so dass Strukturen innerhalb derer nicht fehlerfrei extrahiert werden können. Zum anderen führt eine hohe Geschwindigkeit zu kleineren Überlappungsbereichen aufeinander folgender Bilder und somit zur Minimierung gemeinsamer Bildpunkte, anhand derer der ICP-Algorithmus die Verschiebung der Bilder zueinander berechnet. Dies führt ebenfalls dazu, dass die Genauigkeit des Verfahrens negativ beeinflusst wird. Zur Vermeidung dessen, können die aufgenommenen Bilddatensätze mit synchron erfassten Radencoderdaten entsprechend initial verschoben werden. Dies führt zur Minimierung der Wahrscheinlichkeit einer Konvergenz des ICP-Algorithmus im Bereich von lokalen Minima und somit zur Minimierung von Fehlberechnungen bei schlecht geeigneten 3D-Datensätzen. Darüber hinaus resultiert die initiale Verschiebung der 3D-Bilddatensätze in einer schnelleren Berechnung des ICP-Algorithmus, da die Anzahl der benötigten Iterationen bis der Algorithmus konvergiert, reduziert wird.

Die Regelung des mobilen Roboters setzt eine zeitäquidistante Generierung von Posedaten in einem Zyklus von mindestens 50ms voraus. Die Betrachtungen zeigten jedoch, dass der ICP-Algorithmus zur Berechnung von Posedaten eine Zeitvarianz aufweist und die Berechnungszeit abhängig von der Anzahl der berücksichtigten 3D-Daten zwischen 100ms und 200ms liegt. Um diese Methode dennoch zur Roboterregelung verwenden zu können, wird diese um das in Abbildung 5.4.12 dargestellte Poseschätzverfahren erweitert.[2]

[2] Bisher wurden die einzelnen Verfahren zur Lokalisierung behandelt. Im Folgenden schließt sich die Entwicklung eines speziell entwickelten geeigneten Verfahrens zur Realisierung einer zeitäquidistanten Generierung der zu fusionierenden Lokalisierungsalgorithmen an.

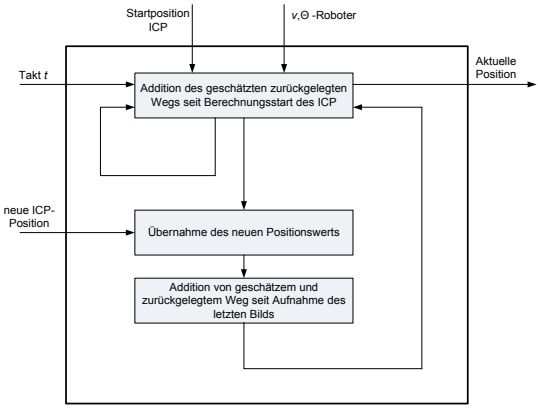


Abbildung 5.4.12: Positionsschätzung zur zeitinvarianten Ermittlung von ICP-Posedaten

Während der Berechnung der Bildvektoren wird unabhängig hiervon die Position und die Orientierung des mobilen Roboters im 50ms Takt anhand der Geschwindigkeit und des Lenkwinkels, welche der Robotersteuerung entnommen werden, geschätzt. Sobald der ICP-Algorithmus eine neue Roboterpose bereitstellt, wird die vorherige Schätzung dementsprechend korrigiert. Dieses Verfahren ermöglicht eine zeitinvariante Erzeugung von Roboterposedaten mit einer Zykluszeit von 50ms.

5.4.3 Sensorfusion mit dem Kalman-Filter

Zur Erzeugung einer robusten Methode zur relativen Selbstlokalisierung der mobilen Roboterplattform, müssen die in den Kapiteln 5.4.1 und 5.4.2 vorgestellten Verfahren zu einer Lokalisiermethode fusioniert werden. Dies kann nicht durch die Bildung von Mittelwerten zwischen beiden Methoden erfolgen. Hierbei hätten einzelne Messfehler eines Systems einen sehr starken Einfluss auf das Ergebnis des gesamten Lokalisiersystems. Vielmehr wäre es von Vorteil die einzelnen Messungen anhand der vorangegangenen Messungen so zu gewichten, das einzelne Messfehler nur einen geringeren Einfluss auf die berechnete Roboterpose haben. Ein in der Literatur oft zu findendes Verfahren zur Sensorfusion, welches diese Eigenschaft besitzt, ist das Kalman-Filter.

Das Kalman-Filter wurde im Jahr 1960 von Rudolph E. Kalman entwickelt. Dieses beschreibt eine rekursive Lösung für das Problem der linearen Filterung von diskreten

Daten [Kalman60] und besteht aus einer Reihe von mathematischen Gleichungen, welche eine Prädiktor-Korrektor-Schätzung beschreiben.

Das Filter wird innerhalb der Robotik häufig zur Bewegungsschätzung, zum Tracking von Objekten aber auch, wie in dieser Arbeit, für die Fusion von zwei oder mehreren Sensoren angewandt. In den letzten Jahren war das Kalman-Filter Gegenstand von vielen Veröffentlichungen. Aufgrund dessen wird im Folgenden nur kurz zum besseren Verständnis der Funktionsweise auf die Grundlagen des Filters eingegangen.

Das diskrete Kalman-Filter

Das diskrete Kalman-Filter versucht eine Vorhersage eines Zustands $\vec{x} \in \mathbb{R}^n$ eines zeitdiskreten Regelprozesses zu treffen, welches von der linearen Differenzengleichung beschrieben wird:

$$\vec{x}_k = A\vec{x}_{k-1} + B\vec{u}_k + \vec{w}_{k-1} \tag{5.51}$$

Mit dem hierzu gehörigen Messvektor $\vec{z} \in \mathbb{R}^m$:

$$\vec{z}_k = H\vec{x}_k + \vec{v}_k \tag{5.52}$$

$\vec{w}_k$ und $\vec{v}_k$sind Zufallsvariablen die das Prozess- und Messrauschen darstellen. Die Matrix A in Gl.(5.51) ist eine $n \times n$ Matrix in der die Beziehung zwischen dem Zustandsvektor $\vec{x}$ zum Zeitpunkt k und dem zum Zeitpunkt $k-1$ hergestellt wird. B ist eine $n \times l$ Matrix und verbindet den Reglereingang $\vec{u} \in \mathbb{R}^l$ mit dem Zustandsvektor $\vec{x}$ zum Zeitpunkt k. H in Gl.(5.52) ist eine $m \times n$ Matrix und verbindet den Messvektor $\vec{z}$ mit dem Zustandsvektor $\vec{x}$ zum Zeitpunkt k. Beide Matrizen A und H werden hier als konstant angenommen. In der Praxis jedoch würden sich diese bei jeder Messung ändern.

Es werden zwei Zustandsschätzungen definiert, eine „a priori"-Schätzung $\hat{x}_k^- \in R^n$ und eine „a-postiori"-Schätzung $\hat{x}_k^+ \in \mathbb{R}^n$ zum Zeitpunkt k. Bezogen auf den Zustandsvektor können die „a-priori" und „a-postiori"-Schätzfehler wie folgt definiert werden,

$$\vec{e}_k^- = \vec{x}_k - \hat{x}_k^- \tag{5.53}$$

$$\vec{e}_k^+ = \vec{x}_k - \hat{x}_k^+ \tag{5.54}$$

so dass sich die „a-priori"-Fehlerkovarianz berechnet zu:

$$P_k^- = E[\vec{e}_k^- \vec{e}_k^{-T}] \tag{5.55}$$

sowie die „a postiori"-Fehlerkovarianz zu:

$$P_k^+ = E\left[\vec{e}_k^+ \vec{e}_k^{+T}\right] \tag{5.56}$$

Nach [Maybeck79] wird die „a postiori"-Schätzung des Zustandsvektors berechnet durch:

$$\hat{x}_k = \hat{x}_k^- + K(\vec{z}_k - H\hat{x}_k^-) \tag{5.57}$$

Dies entspricht einer linearen Kombination aus der „a priori"-Schätzung $\hat{x}_k^-$ und der gewichteten Differenz zwischen dem aktuellen Messvektor $\vec{z}_k$ und der Messvorhersage $H\hat{x}_k^-$. Ist diese Differenz gleich Null stimmen Messung und Vorhersage überein. Der Gewichtungsfaktor K, eine $m \times n$-Matrix, in Gl.(5.57) minimiert die „a postiori"-Fehlerkovarianz in Gl.(5.55) und wird berechnet nach [Welch01] zu:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{5.58}$$

Nach Gl.(5.58) steigt der Gewichtungsfaktor K bei abnehmender Messfehlerkovarianzmatrix R, so dass gilt:

$$\lim_{R \to 0} K_k = H^{-1} \tag{5.59}$$

Geht die „a-priori"-Fehlerkovarianz P_k^-gegen Null gilt für den Verstärkungsfaktor K:

$$\lim_{P_k^- \to 0} K_k = 0 \tag{5.60}$$

Aus den Gl.(5.59) und Gl.(5.60) ist zu entnehmen, dass bei abnehmender Messfehlerkovarianzmatrix R der Messung $\vec{z}_k$ immer stärker vertraut wird, während der vorhergesagten Messung $H\hat{x}_k^-$ weniger vertraut wird. Für abnehmende Fehlerkovarianz P_k^- ist dieser Sachverhalt entgegengesetzt.

Das Kalman-Filter benutzt eine Feedback-Kontrollschleife zur Schätzung des Zustandsvektors. Der Filteralgorithmus besteht aus zwei Schritten der zeitlichen Aktualisierung „Schätzung" und der Aktualisierung der Messung. So können die hierfür hergeleiteten Formeln Gl.(5.51) bis Gl(5.58) aufgeteilt werden in „time-update"-Gleichungen:

$$\vec{x}_k^- = A\hat{x}_{k-1} + B\vec{u}_k \tag{5.61}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{5.62}$$

Und den „measurement update“-Gleichungen:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{5.63}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(\vec{z}_k - H\hat{x}_k^-) \tag{5.64}$$

$$P_k = (I - K_k H)P_k^- \tag{5.65}$$

Zusammenfassend kann der Algorithmus des Kalman-Filters wie folgt dargestellt werden:

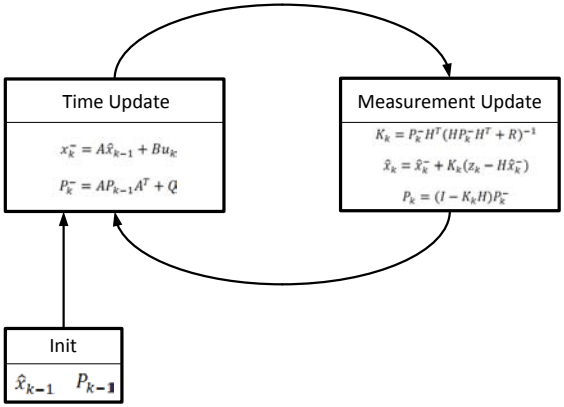


Abbildung 5.4.13: Flussdiagramm zum Kalman-Filter

Implementierung des Kalman-Filters zur Sensorfusion

Die Sensordatenfusion zwischen den Radencoder- und Bildvektordaten erfolgt, wie auf Abbildung 5.4.13 dargestellt, durch Berechnung des Fehlers zwischen selbigen. Dieser Fehler wird durch das Kalman-Filter geschätzt und von den Odometriedaten subtrahiert. Die Verwendung des Kalman-Filters resultiert in der Gewichtung von Ist- und Fehlerschätzwert. Sollte ein Messfehler eines Sensors vorliegen und somit der Fehler überproportional steigen, führt dies zu einer stärkeren Gewichtung des

Schätzwertes, was zur Folge hat, dass der Ist-Fehler einen geringeren Einfluss auf den aktuellen Fehler hat.

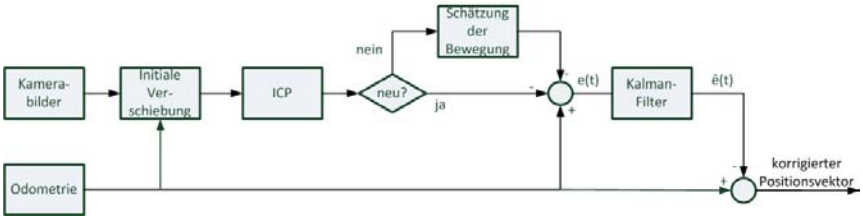


Abbildung 5.4.14: Flussdiagramm Sensorfusion

Zur Initialisierung des Kalman-Filters müssen dessen Elemente definiert werden. Der dem Filter zugrundegelegte Zustandsvektor beträgt:

$$\vec{x}_k = \begin{pmatrix} e_x & e_y & e_\theta & \dot{e}_x & \dot{e}_y & \dot{e}_\theta \end{pmatrix}^T$$

e_x, e_y und e_θ entsprechen den Elementen des Roboterposefehlers zwischen den beiden Berechnungsmethoden und $\dot{e}_x$, $\dot{e}_y$ und $\dot{e}_\theta$ den jeweiligen Änderungsgeschwindigkeiten.

Somit ergibt sich im vorliegenden Fall für die Zustandsübergangsmatrix A nach Gl.(5.51):

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta T & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Der Zustandsvektor $\vec{x}_{k-1}$ beschreibt den angenommenen Fehler zwischen den beiden Sensordaten. Zu Beginn der Roboterbewegung ist dieser:

$$\vec{x}_{k-1} = (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T$$

Das System ist nicht regelbar, somit ist die Matrix $B = 0$. In dem vorliegenden System sind lediglich die Fehlerpositionsvektoren sowie der Orientierungsfehler beobachtbar. Dies führt zu dem Messvektor $\vec{z}_k$:

$$\vec{z}_k = (e_x \quad e_y \quad e_\theta)^T$$

Die Messmatrix H lautet folglich:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Ebenfalls müssen die konstanten Kovarianzmatrizen des Messrauschens R und des Prozessrauschens Q bestimmt werden. Ist die Messkovarianzmatrix R größer als die Prozesskovarianzmatrix Q wird der Einfluss der Prädiktion bei der Berechnung des nächsten Fehlervektors höher gewichtet, als der Einfluss der aktuellen Messung. Verhält es sich andersherum, wird der Messung mehr vertraut und der Einfluss der Prädiktion ist geringer. Während die Messkovarianzmatrix bestimmt werden kann, kann nur schwer eine Aussage über die reale Systemfehlerkovarianz Q getroffen werden. Die Messfehlerkovarianz wird ermittelt, indem die Position des Roboters im Stand über eine längere Zeit mittels PMD-Kamera und Radencoder berechnet wird. Der Fehler zwischen beiden Systemen wird für jede Messung ermittelt. Aus diesen kann anschließend die Fehlerkovarianzmatrix R berechnet werden. Hierbei ergab sich für das verwendete System ein Messfehlerkovarianzmatrix von:

$$R = \begin{bmatrix} 0{,}093 & 0 & 0 \\ 0 & 0{,}051 & 0 \\ 0 & 0 & 1{,}04 \end{bmatrix}$$

Für die Initialisierung der Systemfehlerkovarianzmatrix Q wurde ein sehr geringer Fehler angenommen.

$$Q = 10^{-6} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ein hiervon abweichender Fehler wird zur Laufzeit des iterativen Kalman-Filters in der Berechnung der Fehlerkovarianzmatrix P berücksichtigt.

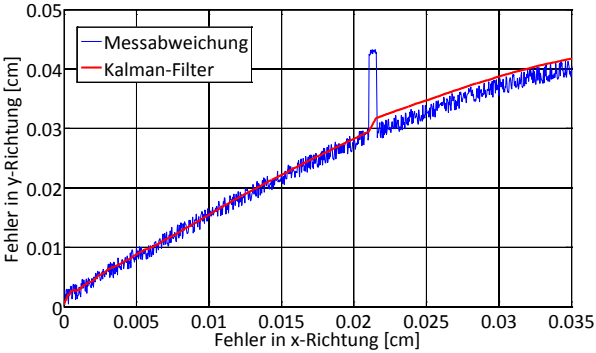


Abbildung 5.4.15: Simulation des Roboterposefehlers unter Benutzung des Kalman-Filters

Abbildung 5.4.15 zeigt die Simulation einer Roboterfahrt und die hierbei entstehende Messdatenabweichung zwischen PMD-Kamera und Radencodermessdaten. Zu erkennen ist, dass der Verlauf des Messfehlers durch Verwendung des Kalman-Filters geglättet wird. Das Messrauschen der Sensoren hat somit einen geringeren Einfluss auf die Qualität der Roboterregelung. Sollte zudem einer der Sensoren eine plötzlich auftretende hohe Abweichung zu den vorherigen Messdaten aufweisen und somit der Posefehler zwischen den Kamera- und Radencodermessdaten von dem mittels Kalman-Filter prädizierten Posefehler stark abweichen, wird dieser anhand der Fehlerkovarianzmatrix P stärker gewichtet als der aktuelle gemessene Posefehler, so dass der Ausgangswert des Kalman-Filters dem Eingangswert nicht oder nur geringfügig folgt. Dies ist in der zweiten Hälfte im Graphen der Abbildung 5.4.15 dargestellt. Hier weichen die Eingangsdaten stark von den vorherigen Messdaten ab.

Folglich wurde ein Verfahren implementiert, welches eine robuste Fusion der Sensordaten ermöglicht. Sollten beispielweise sich bewegende Objekte innerhalb des PMD-Bildbereichs zu Fehlmessungen des Bewegungsvektors führen, wird aufgrund des Kalman-Filters der Schätzwert stärker berücksichtigt. Infolgedessen führt dies nicht zu einer Fehlberechnung der Roboterpose.

5.4.4 Absolute Positionsbestimmung mit künstlichen Landmarken

Die absolute Positionsbestimmung anhand von künstlichen Landmarken dient zum einen der Bestimmung der Startposition, zum anderen zur Korrektur der über die relativen Lokalisierverfahren ermittelten Roboterlage.

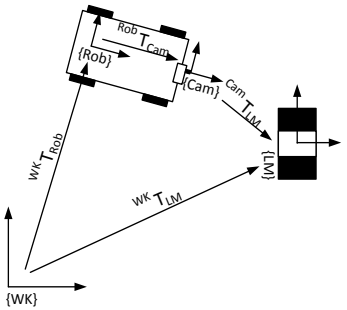


Abbildung 5.4.16: Skizze zur Berechnung der Roboterpose

Zur Ermittlung der Startposition gilt es, die Transformationsmatrix ${}^{WK}T_{Rob}$ für die Anfangsposition zu bestimmen. ${}^{WK}T_{Rob}$ beschreibt, siehe Abbildung 5.4.16, die Transformation zwischen Weltkoordinatensystem *{WK}* und Roboterkoordinatensystem *{Rob}*. Zur Bestimmung der Matrix wird zunächst der Roboter manuell vor einer Landmarke positioniert, so dass sich die Landmarke im Blickfeld der Kamera befindet. Die Translation ${}^{Cam}\vec{t}_{LM}$ und Orientierung der Landmarke zwischen Kamera *{Cam}*- und Landmarkenkoordinatensystem {LM}, kann über die in den nächsten Abschnitten behandelten Verfahren bestimmt werden. Das der Landmarke zugrundeliegende Koordinatensystem wird so gewählt, dass der x-Vektor parallel zu den schwarz-weiß Übergängen der Landmarke liegt. Der z-Vektor verläuft senkrecht zur Ebene auf der sich die Landmarke befindet und kann über die Berechnung des Normalenvektors der Landmarke, analog zu Kapitel 5.3.1, ermittelt werden. Der y-Vektor entspricht folglich dem Kreuzprodukt zwischen $\vec{z}$ und $\vec{x}$:

$$\vec{y} = \vec{z} \times \vec{x} \tag{5.66}$$

Die Orientierung der x-Achse, des mit der Kamera ermittelten Koordinatensystems der Landmarke, muss jedoch nicht zwangsläufig mit dem in der Karte verzeichneten System der Landmarke übereinstimmen, da die Richtung des ersten Koordinatensystems abhängig von der Blickrichtung der Kamera ist. Es gilt:

$$\{LM\}_{Karte} = \{LM\}_{Kamera} \cdot \begin{bmatrix} \cos\vartheta & -sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.67}$$

mit: $\vartheta = \vartheta + \pi k \qquad k = 0,1,2,\ldots$

Der mobile Roboter muss folglich für die Initialisierung der Startposition, in einer vorher definierten Richtung, an die Landmarke heranfahren. Ebenso muss vom Anwender definiert werden, um welche in der Karte befindliche Landmarke es sich handelt. Wird dies berücksichtigt, kann ein Koordinatensystem in die Landmarke gelegt werden, so dass die Transformationsmatrix ${}^{WK}T_{Rob}$ zur Berechnung der Startposition verifiziert werden kann:

$$ {}^{WK}T_{Rob} = {}^{WK}T_{LM} \cdot {}^{Cam}T_{LM}^{-1} \cdot {}^{Rob}T_{Cam}^{-1} \tag{5.68} $$

Die Erkennung von künstlichen Landmarken eignet sich ebenfalls zur Korrektur des Roboterpositionsfehlers, welcher durch das Aufsummieren von Fehlmessungen, beim Gebrauch der relativen Lokalisiermethoden entstanden ist. Zu diesem Zweck werden künstliche Landmarken an diversen Stellen in der abzufahrenden Umgebung platziert. Deren genaue Position und Orientierung wird in die Umgebungskarte eingetragen. Durch Detektion der Landmarken mit der PMD-Kamera kann so auf die tatsächliche Position des Roboters innerhalb der Karte geschlossen werden. Die Position der Landmarke, bezogen auf das Weltkoordinatensystem, entspricht:

$$ {}^{WK}\vec{p}_{LM} = {}^{WK}T_{Rob,rel} \cdot {}^{Rob,rel}T_{Kam} \cdot {}^{Kam}\vec{p}_{LM} \tag{5.69} $$

Durch das Vergleichen der berechneten Landmarkenposition mit denen in der Umgebungskarte, kann auf die entsprechende Landmarke geschlossen werden. Die Position dieser Landmarke entspricht, in Analogie zu Abbildung 5.4.17, ${}^{WK}\vec{p}_{Karte}$ und die Orientierung ${}^{WK}R_{Karte}$.

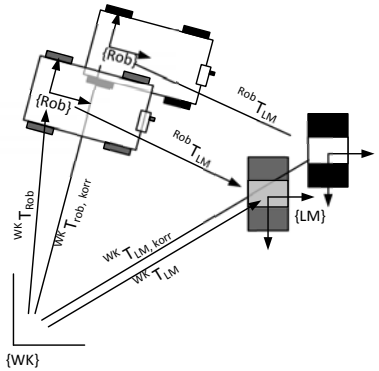


Abbildung 5.4.17: Skizze zur Berechnung der Roboterpose anhand von künstlichen Landmarken

Die bei der Detektion gemessene Orientierung der Landmarke wird berechnet zu:

$$ {}^{WK}R_{LM} = {}^{WK}R_{LM,gemessen} \cdot \begin{bmatrix} \cos\vartheta & -sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.70} $$

mit: $\vartheta = \vartheta + \pi k \qquad k = [0,1]$

${}^{WK}R_{LM,gemessen}$ wird so bestimmt, dass der Vektor $\vec{x}$ in Fahrtrichtung zeigt. Durch die relative Poseschätzung kann der Faktor k so gewählt werden, dass die gemessene Orientierung der Landmarke die gleiche Ausrichtung besitzt, wie die in der Karte verzeichnete Landmarke.

Unter Berücksichtigung der beiden Tranformationsmatrizen ${}^{WK}T_{LM,Karte}$ und ${}^{WK}T_{LM,gemessen}$ der künstlichen Landmarke kann eine Positionskorrektur erfolgen. Nach Abbildung 5.4.17 berechnen sich die Transformationsmatrizen zu:

$$ {}^{WK}T_{LM,gemessen} = {}^{WK}T_{Rob,rel} \cdot {}^{Rob}T_{LM,gemessen} \tag{5.71} $$

$$ {}^{WK}T_{LM,Karte} = {}^{WK}T_{Rob,korr} \cdot {}^{Rob}T_{LM,gemessen} \tag{5.72} $$

Die korrigierte Roboterposition entspricht

$$ {}^{WK}T_{Rob,korr} = {}^{WK}T_{LM,Karte} \cdot {}^{Rob}T^{-1}_{LM,gemessen} \tag{5.73} $$

Mit Gl.(5.73) und Gl.(5.71) lässt sich eine Korrekturmatrix zur Korrektur der relativen Pose berechnen:

$$ {}^{korr}T_{rel} = {}^{WK}T^{-1}_{Rob,korr} \cdot {}^{WK}T_{Rob,rel} \tag{5.74} $$

Die korrigierte Position des Roboters entspricht:

$$ \vec{p}_{korr} = {}^{korr}T_{rel} \cdot \vec{p}_{rel} \tag{5.75} $$

Hierbei ist $\vec{p}_{rel}$ die von den Radencodern und der Kamera durch Sensorfusion bestimmte und $\vec{p}_{korr}$ die korrigierte Roboterposition.

2D Objekterkennung mit Haar-Like-Features

Die Erkennung der künstlichen Landmarken erfolgt, basierend auf den Arbeiten von [Viola01], über die Objektidentifizierung mittels Haar-Like-Features. Hierfür werden Methoden der frei verfügbare Computer Vision Bibliothek OpenCV von Intel© verwendet.

Das Verfahren basiert auf einem 2D-Objekterkennungsalgorithmus. Hierzu werden die von der PMD-Kamera generierten Intensitätswerte verwendet. Die gemessenen Entfernungswerte finden zur Objekterkennung keine Verwendung. Die von [Viola01] eingeführten Haar-Like-Feature dienen der Extraktion von Merkmalen zwischen unterschiedlichen Pixelverbänden. Hierzu werden unterschiedliche Haar-Like-Feature benutzt.

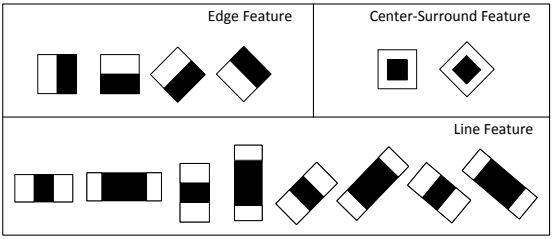


Abbildung 5.4.18: erweiterte Auswahl von Haar-Like-Features

Abbildung 5.4.18 führt einige Beispiele der verwendeten Haar-Like-Feature auf. Zur Berechnung der Merkmale in einem Bild wird ein Feature, standardmäßig mit einer Größe von 24x24 Pixeln, über das Bild geschoben, anhand dem Merkmale extrahiert werden. Anschließend wird das Feature um einen festgelegten Faktor vergrößert und der Extraktionsprozess beginnt erneut, solange bis die Größe des Features gleich der Größe des Bildes ist. Für die Erkennung der künstlichen Landmarken wurde ein Skalierungsfaktor $s = 1{,}5$ gewählt. Es ist gut vorstellbar, dass die Anzahl der extrahierten Merkmale die Anzahl der Pixel weit übersteigt.

Ein Merkmal wird berechnet durch die Differenz zwischen der Summe aller Pixelwerte innerhalb des schwarzen Bereichs und der Summe aller Pixelwerte innerhalb des weißen Bereichs. Aufgrund der häufigen Summenbildung beim Gebrauch der Haar-Like-Feature werden nach [Viola01] Integral-Bilder zur Beschleunigung der Berechnung erstellt.

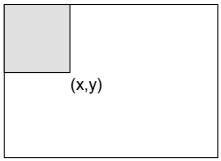


Abbildung 5.4.19: Skizze zur Berechnung von Integralbildern

Der Wert eines Integral-Bilds an der Stelle (x, y) entspricht der Summe aller Pixelwerte links oberhalb des entsprechenden Pixels.

$$ii(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y') \tag{5.76}$$

$ii(x,y)$ entspricht dem Integral-Bild, berechnet aus dem originalen Grauwertbild $i(x,y)$. Unter der Benutzung der folgenden Gleichungen:

$$s(x,y) = s(x,y-1) + i(x,y) \tag{5.77}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{5.78}$$

wobei $s(x,y)$ der Summe aller Pixel innerhalb einer Reihe entspricht. Wenn $s(x,-1) = 0$ und $ii(-1,y) = 0$ ist, kann das Integral-Bild in nur einem Schleifendurchgang berechnet werden. Die Verwendung der Integral-Bilder ermöglicht die Berechnung von Rechtecksummen durch die Verwendung von nur vier Pixelwerten. Die Summe der Werte innerhalb der Fläche D auf Abbildung 5.4.20 entspricht folglich:

$$ii(D) = ii(4) - ii(2) - ii(3) + ii(1) \tag{5.79}$$

Der Wert eines Two-Rectangle-Feature kann auf diese Weise mit nur 6 Werten, eines Three-Rectangle-Features mit 8 Werten und eines Four-Rectangle-Features mit 9 Werten aus dem Integral-Bild bestimmt werden.

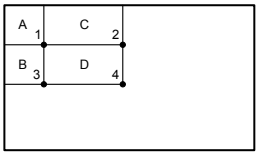


Abbildung 5.4.20: Skizze zur Berechnung eines Rectangle-Feature

Zur Extraktion von Merkmalen, anhand derer das Objekt innerhalb eines Bilds erkannt werden kann, müssen möglichst viele Beispielbilder aufgenommen werden. Hierzu gehören Positivbilder, Bilder die das Objekt enthalten und Negativbilder die es nicht enthalten. Üblicherweise liegt die benötigte Anzahl an Bildern im Bereich von tausend Positiv- und doppelt so vielen Negativbildern. Mit einer Variante des AdaBoost-Algorithmus, einem adaptiven Lernalgorithmus, werden unter Berücksichtigung der aufgenommenen Trainingsbilder durch die Haar-Like-Feature Merkmale zur Objektidentifizierung berechnet. Auf die Funktionsweise des AdaBoost-Algorithmus soll hier nicht näher eingegangen werden. Es wird verwiesen auf die Arbeiten von [Freund99].

Das Ergebnis des AdaBoost-Algorithmus ist eine Reihe an schwachen Klassifizierern. Durch Kaskadierung dieser schwachen Klassifizierer, kann ein sogenannter starker Klassifizierer mit sehr hoher Erkennungswahrscheinlichkeit ermittelt werden, welcher zur Identifizierung des eingelernten Objekts verwendet wird.

Erkennung der künstlichen Landmarken mit Haar-Like-Features

Die zur Lokalisierung verwendete Landmarke besteht aus zwei schwarzen und einem in der Mitte befindlichen weißen Streifen, so wie sie im PMD-Grauwertbild auf Abbildung 5.4.21 dargestellt ist und auch zur Roboter-Kamera-Kalibrierung eingesetzt wird. Aufgrund des Designs der Landmarke kann neben einer Positionsbestimmung ebenfalls die Orientierung der Landmarke auf einfache Weise ermittelt werden.

Abbildung 5.4.21: PMD-Grauwertbild der verwendeten künstlichen Landmarke

Die Ermittlung der Landmarkenposition und -Orientierung erfolgt, wie in dem Ablaufdiagramm auf Abbildung 5.4.22 skizziert ist.

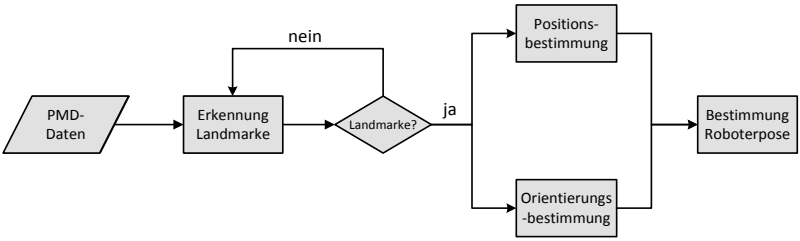


Abbildung 5.4.22: Programmablaufdiagramm zur Ermittlung der Landmarkenpose

Ist eine Landmarke innerhalb der PMD-Aufnahme detektiert, erfolgt die Bestimmung ihrer Position und Orientierung. Hierzu werden zunächst alle Kanten im Bildbereich der detektierten Landmarke berechnet. Die Ermittlung der Kanten erfolgt durch Anwendung eines modifizierten Canny-Filters. Prinzipiell basiert dieser auf der Kantendetektion nach Sobel [Jähne05]. Hierbei werden zwei Differentialfilter verwendet, welche die Ableitung des Bildes in horizontale und vertikale Richtung bilden. Bei einem herkömmlichen Canny-Filter werden die Ableitungen überlagert und anschließend hieraus die Kanten durch Berechnung der lokalen Maxima der Ableitung ermittelt. Durch Modifikation des Filters, so dass das Sobelfilter nur die Ableitung in vertikale oder horizontale Richtung bildet, kann zwischen vertikalen und horizontalen Kanten unterschieden werden.

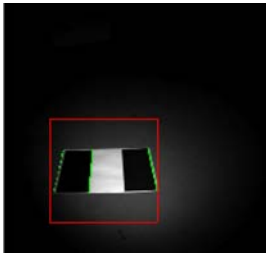


Abbildung 5.4.23: künstliche Landmarke

Abbildung 5.4.23 zeigt umrahmt die erkannte Landmarke mit den markierten Kanten in vertikaler Richtung. Ähnlich wie bei der Roboter-Kamera-Kalibrierung in Kapitel 5.3.2, werden Geradengleichungen unter Anwendung des RANSAC-Algorithmus an die vertikalen Kanten angepasst. Die Orientierung der Landmarke θ_{LM}bezogen auf das Roboterkoordinatensystem berechnet sich folglich zu:

$$\theta_{LM} = arccos\left(\frac{\vec{x}_{Rob} \cdot \vec{l}_{vert}}{|\vec{x}_{Rob}| \cdot |\vec{l}_{vert}|}\right) \tag{5.80}$$

wobei $\vec{l}_{vert}$ der auf die x/y-Ebene des Roboterkoordinatensystems projizierte und über alle vertikalen Kanten gemittelte Richtungsvektor ist. Die Position der künstlichen Landmarke wird durch Ermittlung des Schwerpunkts des weißen Bereichs der Landmarke bestimmt. Dazu wird zunächst das mittlere Pixel des Bereichs in der sich die Landmarke befindet, berechnet. Ausgehend hiervon werden mit einem Bereichssuchverfahren alle Pixel des weißen Bereichs ermittelt. Die Position der Landmarke $\vec{p}_{LM}$ entspricht:

$$\vec{p}_{LM} = \frac{1}{N}\sum_{i=1}^{N}\begin{pmatrix} x \\ y \\ z \end{pmatrix}_i \tag{5.81}$$

Durch Anwendung des beschriebenen Verfahrens zur Erkennung von künstlichen Landmarken lässt sich die Roboterposition korrigieren.

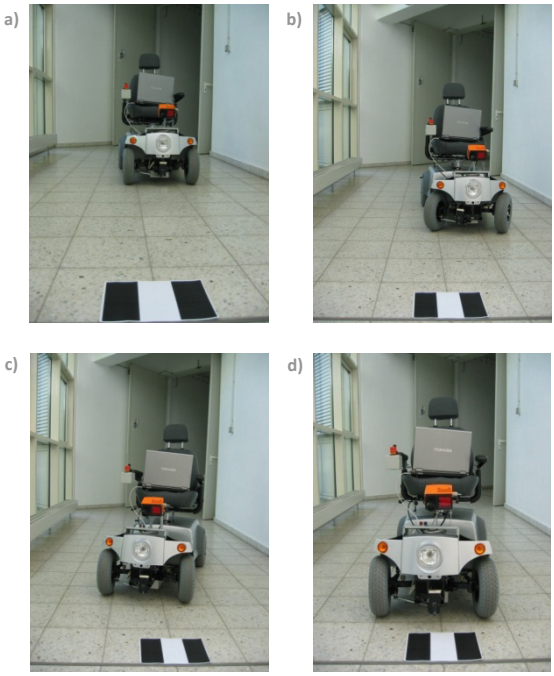


Abbildung 5.4.24: Korrektur der Roboterposition mit künstlichen Landmarken

Nachdem die Position der Landmarke bezüglich der mobilen Roboterplattform bestimmt wurde, kann unter Berücksichtigung der Landmarkenposition, innerhalb der zugrundeliegenden Umgebungskarte, auf die Position und Orientierung des Roboters innerhalb des Weltkoordinatensystems geschlossen werden. Der implementierte Positionsregler des Roboters führt anschließend zur Minimierung des Positionsfehlers, wie dies auf Abbildung 5.4.24a-d zu sehen ist. Im dargestellten Fall soll der Roboter eine Bahn senkrecht zur Landmarke abfahren. Nach Erkennen der Landmarke in Abbildung 5.4.24b erfolgt die Bahnkorrektur.

5.4.5 Fusion der absoluten und relativen Lokalisierung[3]

In den vorherigen Abschnitten wurden verschiedene Methoden zur PMD basierten Positionsbestimmung eingehend behandelt. Für eine präzisen Selbstlokalisierung müssen die verschiedenen Verfahren fusioniert werden. Die relativen Verfahren über Radencoder und Kameradaten werden, wie bereits erwähnt, über die Schätzung des Fehlers mit dem Kalman-Filter fusioniert. Dies stellt eine fortlaufende Posebestimmung dar, deren Fehler durch die Sensorfusion zwar minimiert, jedoch keinesfalls ausgeschlossen werden kann. Um den zwangsläufig entstehenden Fehler auszugleichen, erfolgt nach Abbildung 5.4.25 die Korrektur der Roboterposition und -orientierung durch die absolute Lokalisierung mit künstlichen Landmarken. Sobald eine Landmarke erkannt wird, kann die aktuelle Roboterposition durch Multiplikation mit der Korrekturmatrix korrigiert werden.

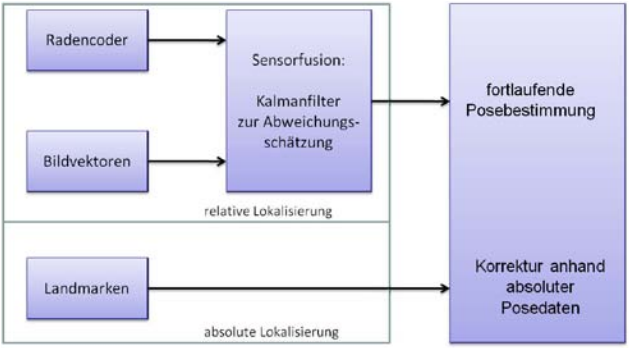


Abbildung 5.4.25: Sensorfusion zur Realisierung des Lokalisiersystems

[33] Entwicklung eines neuen Sensormodells zur PMD basierten Poseerkennung beruhend auf den Odometrie- und PMD-Kameradaten.

5.5 Bahnplanungsverfahren

Das automatisierte Bahnplanungsverfahren, als zentrales Element in diesem Projekt, stellt Anforderungen an die Optimalität hinsichtlich der Weglänge der geplanten Trajektorie und vor allem an die Dauer der Bahnplanung zwischen zwei beliebigen Punkten. Wie in der Einführung zu diesem Kapitel bereits erwähnt, soll das FTS anhand einer vorgegebenen Karte selbstständig zwischen Start- und Zielpunkt navigieren. Hierbei soll dieses die Autonomie besitzen Hindernisse, welche mit der PMD-Kamera erfasst werden, zu umfahren und alternative Trajektorien zum Zielpunkt zu finden. Sollte die Berechnung alternativer Routen nicht möglich sein, muss der Roboter anhalten und warten bis der ursprüngliche Weg wieder passierbar ist.

Der Roboter kann sich hierbei mit einer maximalen Geschwindigkeit von $v_{max} = 8\ km/h$ bewegen. Mit der PMD-Kamera sind, laut der beschriebenen Konfiguration bezüglich Beleuchtungsmodule und Einbauwinkel, Hindernisse erst bei einer Entfernung von $d_{max} = 4\ m$ zu erkennen. In einer Sekunde legt das Fahrzeug einen Weg von $s = 2{,}22\ m$ zurück. Das Fahrzeug besitzt keine aktive Bremse und stoppt nur aufgrund des Widerstands des Elektromotors. Versuche haben gezeigt, dass zum Anhalten mindestens eine Weglänge von $d_{stopp} \approx 1{,}0\ m$ nötig ist. Die Zeit die das Planungsverfahren, also maximal zum Auffinden einer alternativen Strecke benötigen darf, entspricht:

$$t_{max} = \frac{d_{max} - d_{stopp}}{v_{max}} = \frac{4\ m - 1\ m}{2{,}22\ {}^{m}/_{s}} = 1{,}35\ s \tag{5.82}$$

Die maximale Planungsdauer ist die Zeit die dem Roboter nach Erkennen des Hindernisses bleibt, bis der Bremsvorgang gestartet werden muss. Sollte der Roboter eine Kurve durchfahren, ist es vorstellbar, dass das Hindernis nicht von vorne in das Bild kommt, sondern seitlich. In diesem Fall wäre die zuerst detektierte Entfernung des Objekts wesentlich geringer als die angenommene Entfernung, welche zur Berechnung von Gl.(5.82) angenommen wurde. Ziel ist es einen Bahnplaner zu entwerfen, der unabhängig von der Länge des noch zurückzulegenden Weges weniger als $t_{max} = 0{,}7s$ benötigt, um einen alternativen Weg zum Ziel zu finden oder die Entscheidung zu treffen den Roboter anzuhalten.

5.5.1 Ansätze für Bahnplanungsverfahren in der mobilen Robotik

Bei Bahnplanungsverfahren wird zwischen lokaler und globaler Bahnplanung unterschieden. Globale Bahnplanung meint die Planung einer Trajektorie von einem Startpunkt zu einem Zielpunkt. Hierbei wird die Kenntnis der Umgebung, welche durch eine Karte oder durch Sensorinformationen vor der Planung gewonnen wurde, vorausgesetzt. Für die lokale Bahnplanung wird keine Umgebungskarte benötigt. Die Planung in die gewünschte Richtung basiert lediglich auf den aktuellen Informationen im Sichtbereich der Robotersensoren. Eine Umgebungskarte kann hier jedoch auch unterstützend zu den Sensorinformationen verwendet werden.

In der Literatur gibt es diverse globale Bahnplanungsverfahren. Diese Verfahren basieren auf metrischen oder auch topologischen Karten. In neuerer Zeit finden sich jedoch auch einige Planungsverfahren, welche auf Kombinationen beider Karten beruhen.

In [Murray97] wird, basierend auf einer metrischen Darstellung der Roboterumgebungskarte, eine Bahnplanung durchgeführt. Mittels eines „occupancy-grid“-Verfahrens [Thrun05] kann unter Verwendung einer metrischen Karte eine Trajektorie geplant werden. Bei diesen Algorithmen wird die Umgebungskarte üblicherweise in zwei- oder dreidimensionale Zellen aufgeteilt. Durch die Verwendung einer CAD-Karte oder durch Sensorinformationen werden diese Zellen entweder als frei oder besetzt definiert. Bei der Aufnahme der Karte mittels Sensoren, dies können zum Beispiel Sonarsensoren oder auch Laserscanner sein, werden Vektormapping-Verfahren eingesetzt.

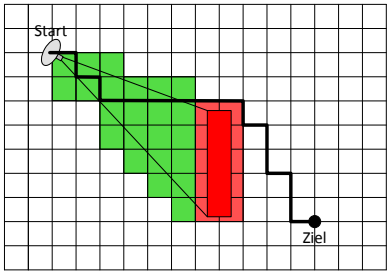


Abbildung 5.5.1: Planung mittels Occupancy-Grid

Hierbei wird eine Zelle als „besetzt“ markiert für die der Sensor die Koordinaten zurück gibt. Alle Zellen zwischen Sensor und der besetzten Zelle werden als „frei“ markiert. Bei

der Planung der Trajektorie wird ein Weg zwischen Start- und Zielpunkt über die als „frei“ markierten Zellen gesucht. Eine metrische Karte ist einfach zu erstellen. Hindernisse, welche nachträglich mit Sensoren erfasst werden, können leicht in die Karte aufgenommen werden. Eine Selbstlokalisierung des mobilen Roboters ist mit Bildsensoren und der Karte durch Anwendung von Mapping-Verfahren möglich. Zudem erlauben metrische Karten die Bestimmung des kürzesten Wegs zwischen zwei beliebigen Punkten. Jedoch benötigt dieses Verfahren aufgrund der Detailgenauigkeit der Karte viel Speicher. Für eine echtzeitfähige Online-Bahnplanung ist die Trajektorienplanung mittels metrischer Karte nur bedingt geeignet. Topologische Karten eignen sich besser.

Topologische Karten innerhalb der Robotik enthalten nur die Informationen über erreichbare Positionen, welche in sogenannten Knoten abgespeichert werden. Die Knoten, die von anderen aus kollisionsfrei erreichbar sind, werden in einem Graphen verbunden. Häufig angewandte Algorithmen zur Berechnung von topologischen Karten verwenden die Methode der Sichtbarkeitsgraphen oder die Berechnung von Voronoidiagrammen.

Ein Sichtbarkeitsgraph G ermöglicht das Suchen der Verbindung von zwei beliebigen Knoten q_{init} und q_{goal}. Hierzu werden in dem Sichbarkeitsgraphen die Knoten q_i, sowie deren Verbindungen E gespeichert. Die Knoten q_i repräsentieren die Positionen aller Hinderniskanten. Verbunden sind zwei Knoten q_1 und q_2, wenn q_2 von q_1 aus sichtbar ist. Dies meint, dass deren Verbindungsgerade kein Hindernis schneidet. Werden dem Sichtbarkeitsgraphen die Knoten q_{init} und q_{goal} hinzugefügt, kann mittels eines beliebigen Graphensuchverfahrens [Latombe91] eine Trajektorie von Start- zu Zielpunkt gefunden werden.

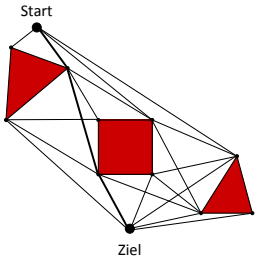


Abbildung 5.5.2: Skizze zur Erstellung eines Sichtbarkeitsgraphen

Nach [Latombe91] entspricht der gefundene Pfad der euklidisch kürzesten Verbindung zwischen Start und Ziel. Die Bahnplanung mittels Sichtbarkeitsgraphen ist eine sehr schnelle und effiziente Methode zur Planung des kürzesten Pfads. Jedoch nur solange die Hindernisumgebung nicht zu komplex ist. Mit zunehmender Hinderniskomplexität erhöht sich die Anzahl der Knoten und damit die Anzahl der möglichen Verbindungen. Aus diesem Grund eignet sich dieser für eine reaktive Bahnplanung nur bis zu einem gewissen Grad an Komplexität und Größe.

Eine weitere Methode eine topologische Karte aufzubauen bietet das Voronoi-diagramm [Ayra02]. Zur Erstellung eines Voronoidiagramms wird der Hindernisraum in verschiedene Teilebenen aufgeteilt. Jedem Hindernis wird eine Teilebene zugewiesen. Die Punkte des Hindernisraums mit dem geringstem Abstand zu einem Hindernis werden der jeweiligen Teilebene zugewiesen. Es ist vorstellbar, dass Punkte zu zwei oder mehreren Hindernissen den gleichen Abstand aufweisen. Diese Punkte bilden die Voronoilinien.

Abbildung 5.5.3: Skizze zur Bahnplanung mit Voronoi-Diagrammen

Besitzt der Hindernisraum eine hohe Komplexität, ist es sehr zeitaufwendig die Voronoilinien vollständig zu berechnen. Demzufolge werden nur markante Voronoipunkte ermittelt. Zusammenhängende Punkte innerhalb des Diagramms werden als Nachbarn gekennzeichnet. Ebenso wie bei der Methode des Sichtbarkeitsgraphen werden Graphensuchverfahren eingesetzt, um den bestmöglichsten Pfad von einer beliebigen Start- zu einer beliebigen Zielkonfiguration zu berechnen. Im Gegensatz zu der Methode des Sichtbarkeitsgraphen führt das Voronoidiagramm, bezogen auf die Länge des geplanten Wegs, nicht zu einer optimalen Lösung. Die Bahnplanung mittels Voronoidiagramm resultiert hingegen in der Maximierung des Abstandes zwischen geplanter Route und den jeweiligen Hindernissen.

5.5.2 Graphensuchverfahren

Wichtige Bestandteil der Bahnplanung anhand von topologischen Karten sind Graphensuchverfahren. Es gibt es eine große Anzahl verschiedenster Suchalgorithmen. Eine Zusammenstellung dieser Verfahren ist in [Laumond98] zu finden. Im Folgenden wird kurz auf die am häufigsten verwendeten Graphensuchverfahren eingegangen. Es wird unterschieden zwischen uninformierten und heuristischen Suchverfahren. Während bei uninformierten Suchverfahren die Wahl des nächsten zu expandierenden Knoten rein zufällig ist, werden bei den heuristischen Verfahren Informationen bezüglich deren Position zum Ziel und/oder zur Startposition berücksichtigt.

Tiefensuche	**Breitensuche**
1. Schreibe den Startknoten q_{init} in den Stack 2. Nehme den ersten Knoten aus dem Stack 3. Ist dieser Knoten q_{akt} gleich dem Zielknoten q_{goal} terminiere den Algorithmus 4. Trage alle Folgeknoten in den Stack ein 5. Gehe zu Schritt 2	1. Schreibe den Startknoten q_{init} in den Stack 2. Nehme den letzten Knoten aus dem Stack 3. Ist dieser Knoten q_{akt} gleich dem Zielknoten q_{goal} terminiere den Algorithmus 4. Trage alle Folgeknoten in den Stack ein 5. Gehe zu Schritt 2

Algorithmus 5.5: Methoden zur Graphensuche

Die wichtigsten uninformierten Suchverfahren sind die Tiefen- und Breitensuche. Die Expansion des jeweiligen Folgeknoten erfolgt ohne Information über dessen Position bezüglich Start- und/oder Zielknoten. Bei Verwendung des **Breitensuchverfahrens** werden die Knoten, wie in Abbildung 5.5.4a dargestellt, expandiert, indem jeweils erst alle Nachbarknoten betrachtet werden, bevor mit den Kindknoten fortgefahren wird.

Im Gegensatz zu dem Breitensuchverfahren expandiert das **Tiefensuchverfahren** die Knoten nach dem LIFO (Last In First Out) - Prinzip. Es werden zunächst die jeweiligen Kindknoten, also die Knoten die zuletzt in das Knotenregister gespeichert wurden, untersucht.

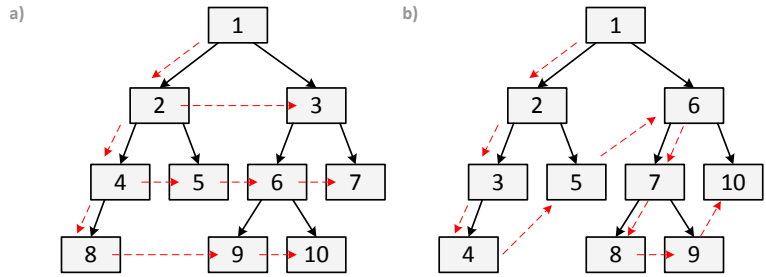


Abbildung 5.5.4: Suchschema: a) Breitensuchverfahren b) Tiefensuchverfahren

Beide vorgestellten Verfahren besitzen einen Speicherplatzverbrauch von $O(n)$, wobei $n = |V| + |E|$ ist, also der Summe aller Ecken V und Pfadsegmente E entspricht. Im schlechtesten Fall müssen alle Knoten des Graphen untersucht werden. Dies resultiert in einer maximalen Laufzeit von $O(n)$. Des Weiteren sind beide Algorithmen nicht optimal. Der gefundene Pfad zwischen Start- und Zielknoten entspricht zwar der kürzesten Verbindung zwischen den Knoten, jedoch entspricht dies nicht in jedem Fall dem minimalen euklidischen Abstand zwischen Start und Ziel. Aufgrund des hohen Speicherverbrauchs und der ebenfalls hohen maximalen Laufzeit sind diese Verfahren nur für kleine Graphen mit geringer Komplexität geeignet.

Besser geeignet sind hier Suchalgorithmen aus der Klasse der heuristischen Verfahren. Zu nennen ist hier der Algorithmus von Dijkstra [Dijkstra59] sowie der hieraus weiterentwickelte A*-Algorithmus [Hart68]. Diese suchen mit der Hilfe einer Kostenfunktion einen Weg zwischen Start- und Zielknoten.

Im Folgenden soll der A*-Algorithmus näher betrachtet werden. Zur Erläuterung der Funktionsweise des Dijkstra-Algorithmus sei auf [Dijkstra59] verwiesen.

Ein Suchgraph G wird definiert durch eine Menge von Knoten $\{n_i\}$ und deren Verbindungen $\{\vec{e}_{ij}\}$ zueinander. Die Verbindung $\vec{e}_{pq}$ zum Beispiel verbindet den Knoten n_p und dessen Nachfolgeknoten n_q miteinander. Ein Knoten enthält Informationen über dessen Position im Graphen G sowie einen Zeiger auf den darüberligenden "parent"- Knoten und je einen Zeiger auf dessen „child"-Knoten. Allen Verbindungen $\{\vec{e}_{ij}\}$ werden durch eine Kostenfunktion die Kosten c_{ij} zugewiesen. Die Kosten eines Knoten ergeben sich aus der Summation der einzelnen Kosten c_i , c_{i+1}, … aller bis dahin zurückgelegten Pfadsegmente. Der optimale Weg zwischen den Knoten n_i und n_j enspricht folglich dem Pfad, welcher die geringsten Kosten verursacht. $h(n_i, n_j)$ entspricht der Kostenfunktion. Zur Berechnung des optimalen Pfads wird die

Evaluierungsfunktion $f(n)$ eingeführt. Der als nächstes zu expandierende Knoten ist immer der Knoten, dessen Kosten den geringsten Wert f aufweist.

A*-Algorithmus

1. Markiere den Knoten s als „offen“ und berechne $f(s)$
2. Wähle aus der Liste der „offenen“-Knoten denjenigen Knoten n, welcher den geringsten Wert f aufweist.
3. Wenn $n \epsilon T$ ist, wobei T dem Zielknoten entspricht, markiere n als “geschlossen“ und beende den Algorithmus.
4. Andernfalls markiere n als „geschlossen“ und generiere dessen Folgeknoten. Berechne den Wert f der neuen Knoten und markieren diese als „offen“, wenn der Knoten noch nicht in der „geschlossen“-Liste vorhanden ist. Sollte der Knoten in der „child“-Liste vorhanden sein, wird der Knoten mit geringeren Kosten übernommen.
5. Gehe zu Schritt 2.

Algorithmus 5.6: Ablauf des A*-Algorithmus

Die Evaluierungsfunktion $f(n)$ besteht aus zwei Anteilen und berechnet sich zu:

$$f(n) = g(n) + h(n) \tag{5.83}$$

Hierbei entspricht $g(n)$ der Summe der Kosten aller Pfadsegmente des optimalen Pfads vom Startknoten s zum aktuellen Knoten n.

$h(n)$ entspricht einer heuristischen Funktion zur Schätzung der Kosten zwischen dem Knoten n und dem Zielknoten z. Die zur Berechnung der Zielkosten herangezogene heuristische Funktion soll einen Kostenwert berechnen, welcher möglichst gering ist. Zudem darf dieser die tatsächlich anfallenden Kosten keinesfalls überschreiten. Sollte dies der Fall sein, ist nach [Hart68] die Optimalität des Algorithmus nicht mehr gewährleistet.

5.5.3 Umsetzung des Bahnplanungsverfahren[4]

Für den im Lynkeus-Projekt implementierten Bahnplaner wurde ein zweistufiges hybrides Planungsverfahren entwickelt. Die erste Stufe basiert auf einer groben

[4] Die bisherigen aufgeführten Methoden zur Bahnplanung werden im folgenden zu einem eigenen angepassten Verfahren verschmolzen.

Pfadplanung, welche auf Basis einer topologischen Karte durchgeführt wird. Der zweite Planungsschritt dient der detaillierten Planung unter Berücksichtigung der Fahrzeuggeometrie und basiert auf der Berechnung des Pfads mittels metrischer Umgebungskarte.

Erstellung der topologischen Umgebungskarte

In Kapitel 5.5.1 wurden zwei Verfahren zur Berechnung einer topologischen Karte vorgestellt. Während der Sichtbarkeitsgraph sich durch eine schnelle Berechnung aller Knoten und Pfadsegment auszeichnet, hat dieser jedoch den Nachteil, dass die hiermit geplante Route sehr nah an den Hindernissen vorbeiführt. In der Praxis kann dies zu Kollisionen führen, sollte die Position des Roboters vorübergehend falsch bestimmt worden sein und dies zu Abweichungen von der geplanten Bahn führen sollte. Das Voronoi-Diagramm benötigt jedoch einen sehr hohen Berechnungsaufwand, eignet sich jedoch bei der Positionsbestimmung mit PMD-Kamera besser, da die Abstände des Roboters zu den Hindernissen maximiert wird und eventuelle sensorbedingte Fehlberechnungen der Roboterposition nicht zwangsläufig zu Kollisionen führen. Durch ausreichenden Sicherheitsabstand bleibt genügend Zeit für eine Aktualisierung der PMD-Informationen und somit zu einer Korrektur der Roboterposition.

Um die Vorteile beider Verfahren, schnelle Berechnung und Optimalität bezüglich des Abstands zu Hindernissen zu gewährleisten, werden zur Berechnung der topologischen Karte beide Verfahren kombiniert. Zunächst erfolgt die Berechnung der topologischen Karte. Der erste Schritt hierzu ist die Bestimmung eines Sichtbarkeitsgraphen.

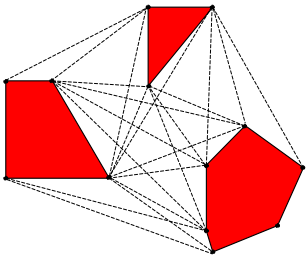


Abbildung 5.5.5: Beispiel zur Erstellung eines Sichtbarkeitsgraphen

Ist dieser bestimmt, werden im nächsten Schritt von jedem Hindernisknoten die kürzesten Verbindungen zu den Knoten anderer Hindernisse ermittelt. Von diesen Verbindungen werden die Mittelpunkte berechnet, welche als Knoten im topologischen Graphen gespeichert werden.

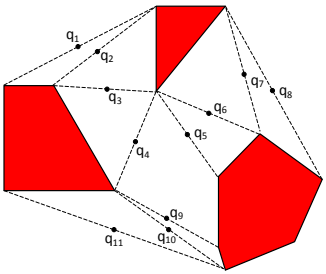


Abbildung 5.5.6: Extraktion der Voronoipunkte / Stützstellen

Auf diese Art können die Knoten der topologischen Karte ermittelt werden. Die berechneten Knoten sind eine Teilmenge des Voronoidiagramms. Als nächstes erfolgt die Definition von Verbindungen zwischen den ermittelten Knoten. Als verbunden gelten diese, wenn die Gerade zwischen ihnen kein Hindernis schneidet. q_7 in Abbildung 5.5.6 ist folglich verbunden mit q_4, q_5, q_6, q_9, q_{10} und q_{11}. Hierdurch entsteht ein ungerichteter Graph der zur Bahnplanung geeignet ist.

Erstellung der metrischen Umgebungskarte

Für die Durchführung des zweiten Bahnplanungsschritts ist die Erstellung einer geeigneten Umgebungskarte notwendig. Die Karte muss eine schnelle und effektive Abfrage der relevanten Hindernisse in der Roboterumgebung ermöglichen.

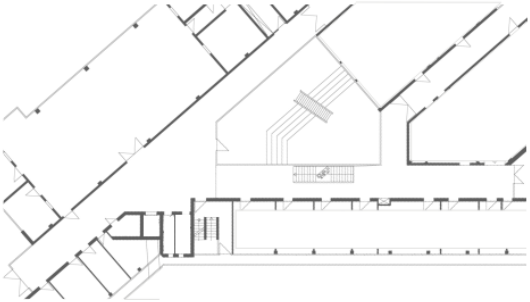


Abbildung 5.5.7: Karte der realen Testumgebung

Als gute Lösung erwiesen sich das Einlesen und das anschließende Konvertieren einer CAD-Umgebungskarte in eine geeignete Quadtree-Repräsentation. Abbildung 5.5.7

zeigt die originale CAD-Umgebungskarte. Die zur Bahnplanung verwendete Karte ist in Abbildung 5.5.8 dargestellt. Rot eingezeichnet sind die berechneten Quadtrees.

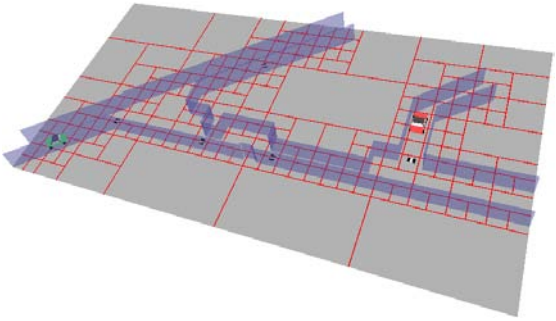


Abbildung 5.5.8: modellierte Karte mit Quadtree-Einteilung

Die Einordnung der Hindernisse in Quadtrees gewährleistet eine effiziente Überprüfung von Roboterpositionen auf Kollision. Hierbei werden nur die Hindernisse berücksichtigt, die sich im gleichen Quadtree befinden wie der Roboter.

5.5.4 Bahnplanungsalgorithmus

Wie bereits erwähnt, handelt es sich bei dem implementierten Verfahren um ein hybrides, zweistufiges Planungsverfahren. Der erste Planungsschritt, wie auch im Flussdiagramm auf Abbildung 5.5.9 verdeutlicht wird, ist eine Vorplanung anhand der topologischen Karte. Diese dient der Unterstützung der eigentlichen metrischen Pfadplanung.

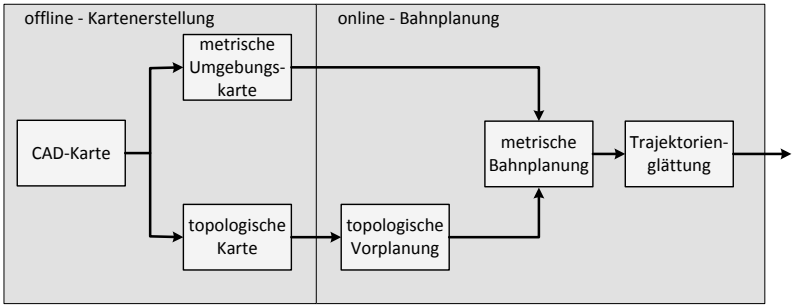


Abbildung 5.5.9: Flussdiagramm zur Bahnplanung

Für die topologische Vorplanung der Trajektorie wird der Dijkstra-Algorithmus [Dijkstra59] verwendet. Dieser funktioniert ähnlich wie der in Kapitel 5.5.2 vorgestellte A*-Algorithmus. Der Unterschied liegt in der Berechnung der Kostenfunktion $f(q)$, die zur Auswahl des nächsten Knotenpunktes im topologischen Graphen dient. Während die Kostenfunktion des A*-Algorithmus, wie bereits erwähnt, sich berechnet zu

$$f(q) = g(q) + h(q) \tag{5.84}$$

wobei $h(q)$ den Zielkosten und $g(q)$ den Kosten des bereits zurückgelegten Weges entspricht, berechnen sich die Kosten des Dijkstra-Algorithmus lediglich aus der Entfernung zwischen Ist- und Zielposition, also dem Weg der noch zurückgelegt werden muss. Es gilt:

$$f(q) = h(q) \tag{5.85}$$

mit:

$$h(q) = \|q_{goal} - q\| \tag{5.86}$$

Aufgrund der Vorplanung bei gegebenen Start- und Zielpunkt wird nach Gl.(5.85) der kürzeste Pfad gefunden. In Abbildung 5.5.10 ist das Ergebnis der Vorplanung, bei Verwendung der gegebenen Karte, dargestellt. Rot eingezeichnet ist die Startposition und grün die Zielposition des Fahrzeugs. Die berechnete Verbindung (rote Linie) verbindet Start und Ziel über die bei der Vorplanung berechneten Verbindungsknoten (schwarze Punkte).

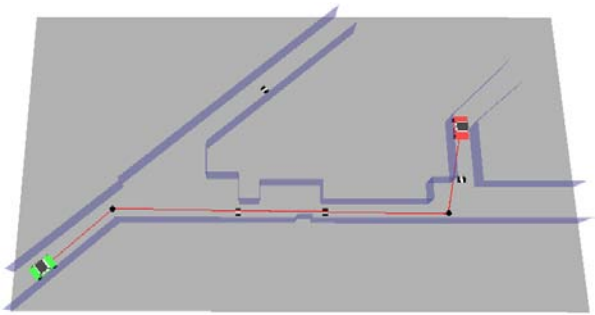


Abbildung 5.5.10: Beispiel zur Berechnung des ersten Bahnplanungsschritts

Ist die Vorplanung durchgeführt, schließt sich hieran die Bahnplanung basierend auf einer metrischen Umgebungskarte unter Berücksichtigung der berechneten

Stützpunkte an. Die Vorplanung wird ohne Berücksichtigung der Fahrzeuggeometrie und der Fahrzeugbeschränkungen, getätigt. Die metrische Bahnplanung jedoch setzt genaue Kenntnisse von Größe und Eigenschaften des mobilen Roboters voraus.

Durch die vorangegangene grobe Vorplanung der Robotertrajektorie ist es möglich, die metrische Bahnplanung in verschiedene Abschnitte zu unterteilen. Die metrische Planung erfolgt von einem Stützpunkt zum anderen. Die einzeln berechneten Pfadsegmente werden nach Abschluss der Trajektorienplanung zusammengefügt. Dies führt zur Minimierung der bei der metrischen Bahnplanung auszuführenden Planungsschritte, da durch die Vorplanung die Richtung des zu planenden Pfads um die vorhandenen Hindernisse vorgegeben ist. Wäre dies nicht der Fall, würde der Bahnplaner eine Strecke auf dem direkten Weg in Richtung Ziel favorisieren. Dies würde beim Vorhandensein von Hindernissen eine Expansion unnötig vieler Roboterpositionen mit sich bringen und somit den Zeit- und Speicheraufwand für die Pfadplanung immens erhöhen.

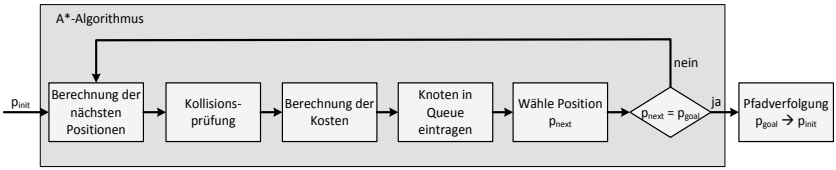


Abbildung 5.5.11: Flussdiagramm zur metrischen Bahnplanung

Abbildung 5.5.11 zeigt das Flussdiagramm des implementierten metrischen Bahnplanungsverfahrens. Der Algorithmus bedient sich dem Djikstra-Algorithmus, welcher in Kapitel 5.5.2 vorgestellt wurde. Der Unterschied hier ist jedoch, dass dem A*-Algorithmus kein topologischer Graph zugrunde gelegt wird. Ausgehend von der Startposition werden zunächst mögliche Folgepositionen berechnet. Auf die Berechnungsweise wird im Verlauf dieses Abschnitts noch näher eingegangen, genauso wie auf andere im Flussdiagramm dargestellte Berechnungsschritte.

Die ermittelten Folgepositionen werden zunächst auf Kollision mit den in der Karte verzeichneten Hindernissen überprüft. Für alle kollisionsfreien Folgepositionen werden die Kosten nach Gl.(5.84) berechnet. Die implementierte Kostenfunktion lautet mit eingesetzten Gewichtungen:

$$f(q) = 0{,}75 \cdot h(q) + 0{,}25 \cdot g(q)$$

mit:

$$h(q) = N(q) \cdot S \qquad g(q) = \|q - q_{goal}\|$$

$N(q)$ ist die Anzahl aller Vorgängerpositionen, die bis zu der aktuellen Position q durchlaufen wurden. S entspricht der Länge des Weges zwischen zwei Positionen. Sind die Kosten für die kollisionsfreien Roboterpositionen berechnet, werden diese in eine Warteschlange „Queue“ inklusive eines Verweises auf den Vorgängerknoten eingetragen. Die nächste Roboterposition wird bestimmt, indem die Queue unter Verwendung des Heapsort-Algorithmus [Williams64] aufsteigend sortiert und die Position mit den geringsten Kosten gewählt wird. Sollte die neu gewählte Roboterposition der Zielposition entsprechen, wird die Pfadsuche beendet und der Pfad wird rekursiv von der Zielposition aus, mit der Hilfe der abgespeicherten Verweise auf die Vorgängerpositionen, ermittelt. Ist die aktuelle Roboterposition ungleich der Zielposition, wird mit der Expansion der Roboterposition mittels A*-Algorithmen fortgefahren.

Berechnung der möglichen Roboterfolgepositionen

Zur Berechnung der möglichen Folgepositionen der aktuellen Roboterposition muss zunächst festgelegt werden, für welche Lenkwinkel die nächsten Positionen berechnet werden sollen. Für den implementierten Bahnplaner wurde eine Diskretisierung des Lenkwinkels in 5° Schritten vorgenommen. Bei einem maximalen Lenkeinschlag von $\varphi = \pm 30°$ werden insgesamt für jeden Planungschritt 13 verschiedene Folgepositionen berechnet. Ein weiterer wichtiger Parameter ist die Länge des bei einem Planungsschritt zurückzulegenden Wegs s. Ebenso wie die Wahl der Schrittweite des Lenkwinkels, bestimmt dieser Parameter während der Bahnplanung die Wendigkeit des Fahrzeugs. Umso geringer die gewählten Schrittweiten sind, umso leichter ist es, eine Bahn durch engere Passagen zu planen. Ein weiterer begrenzender Faktor für die maximale Schrittweite liegt begründet im minimal erlaubten Abstand zwischen mobilem Roboter und den Hindernissen. Umso geringer die Schrittweite ist, umso näher kann der Roboter an die Hindernisse heranfahren. Für die Schrittweite wurde bei der vorliegenden Applikation $s = 0{,}1m$ gewählt.

Bei gewählter Lenkwinkeldiskretisierung, Schrittweite und einer gegebenen Ausgangsposition und -Orientierung kann für jeden beliebigen Lenkwinkel die Folgeposition wie folgt berechnet werden.

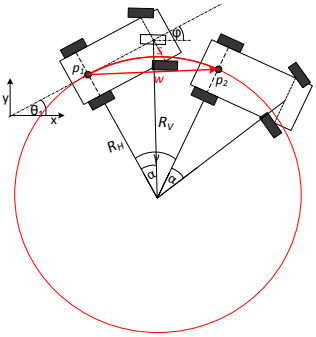


Abbildung 5.5.12: Skizze zur Berechnung der folgenden Roboterposition

Für einen Lenkwinkel $\varphi \neq 0$ bewegt sich der Mittelpunkt der hinteren Achse des mobilen Roboters auf einer Kreisbahn mit dem Radius:

$$R_H = l \cdot \tan\left(\frac{\pi}{2} - |\varphi|\right) \tag{5.87}$$

Bei definierter Schrittweite s beträgt der Winkel γ des hierbei befahrenen Kreissegments:

$$\gamma = \frac{s}{R_H} \tag{5.88}$$

Die Kreissehne w dieses Kreissegments entspricht:

$$w = 2 \cdot R_H \, sin\left(\frac{\gamma}{2}\right) \tag{5.89}$$

Hieraus lässt sich die neue Position $\vec{p}_2$ des Fahrzeugs bei gegebener Position und Orientierung ermitteln.

$$p_{x_2} = p_{x_1} + w \cdot cos\left(\theta_1 - \frac{\gamma}{2}\right) \tag{5.90}$$

$$p_{y_2} = \begin{cases} p_{y_1} + w \cdot sin\left(\theta_1 - \frac{\gamma}{2}\right), & \varphi < 0 \\ p_{y_1} - w \cdot sin\left(\theta_1 - \frac{\gamma}{2}\right), & \varphi > 0 \end{cases} \tag{5.91}$$

Für die neue Orientierung θ_2 gilt:

$$\theta_2 = \begin{cases} \theta_1 + \gamma, & \varphi < 0 \\ \theta_1 - \gamma, & \varphi > 0 \end{cases} \tag{5.92}$$

Für einen Lenkwinkel $\varphi = 0$ gilt:

$$p_{x_2} = p_{x_1} + s \cdot cos(\theta_1) \tag{5.93}$$

$$p_{y_2} = p_{y_1} - s \cdot sin(\theta_1) \tag{5.94}$$

$$\theta_2 = \theta_1 \tag{5.95}$$

Überprüfung der Folgepositionen auf Kollision

Nachdem die möglichen Folgepositionen berechnet sind, müssen diese auf Kollision mit den in der Umgebungskarte befindlichen Hindernissen überprüft werden. Positionen, an denen der Roboter kollidieren würde, werden gelöscht und für die weitere Bahnplanung nicht mehr berücksichtigt.

Da die Kollisionsberechnung für jede berechnete Roboterpose erneut durchgeführt wird, veranschlagt diese einen Großteil der benötigten Berechnungsdauer des Bahnplaners. Folglich ist es von enormer Bedeutung, eine sehr schnelle und effektive Methode zur Kollisionsberechnung zu implementieren. Zu diesem Zweck wird eine angepasste Variante des „Separating-Axis-Theorem“ [Laumond98] angewandt. Dieses besagt, dass wenn sich zwei konvexe Körper nicht schneiden sollten, ein Gerade existiert, auf welche beide Körper projiziert werden können, so dass deren Projektionen nicht überlappen.

Vorteilhaft beim Gebrauch des Theorems ist, dass hier keine Abstände zwischen den einzelnen auf Kollision zu überprüfenden Objekten berechnet werden müssen. Durch einfache Projektionen der Robotereckpunkte sowie der Eckpunkte des Hindernisses

kann eine Kollision erkannt werden. Dies führt nach [Latombe91] zu einer wesentlich kürzeren Berechnungszeit.

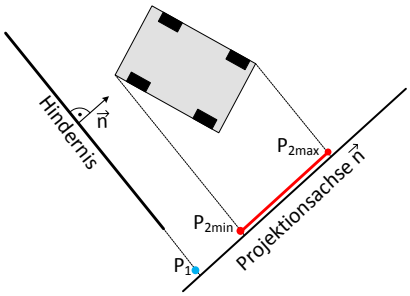


Abbildung 5.5.13: Separating Axes Theorem

Zur Kollisionsüberprüfung werden die Eckpunkte der Hindernisse auf den Normalenvektor einer beliebigen Objektkante projiziert. Auf Abbildung 5.5.13 ist dies für die Projektion auf den Normalenvektor des Hindernisses dargestellt. Sollte für die berechneten Projektionspunkte gelten:

$$\left(P_{1,max} < P_{2,min} \bigwedge P_{1,min} < P_{2,max}\right) \bigvee \left(P_{2,max} < P_{1,min} \bigwedge P_{2,min} < P_{1,max}\right) = true \qquad (5.96)$$

dann ist eine "Separating-Axis" gefunden und der mobile Roboter kollidiert nicht mit dem Hindernis. Ist die Bedingung aus Gl.(5.97) jedoch nicht erfüllt, müssen die Eckpunkte der Objekte auf die übrigen Normalenvektoren der Objektkanten projiziert werden. In unserem Fall entspräche dies der Projektion auf die Normalenvektoren beider Seiten des mobilen Roboters. Sollte hier ebenfalls keine separierende Achse gefunden werden können, kollidiert der Roboter auf der angenommenen Position mit seiner Umgebung und die berechnete Folgeposition wird nicht in die Queue eingetragen.

5.5.5 Auswertung

Unter Verwendung einer CAD-Karte kann eine Umgebungsrepräsentation erstellt werden, die zum einen aus einer in Quadtrees aufgeteilten 3D-Umgebungskarte und zum anderen aus einer topologischen Karte, welche die nach Abschnitt 5.5.1 berechneten Stützpunkte enthält, besteht. Diese Umgebungsrepräsentation ermöglicht

die Verwendung des implementierten 2-stufigen Planungsverfahrens. Die erste Phase berechnet anhand der topologischen Karte die kürzeste Verbindung über die berechneten Stützstellen zwischen Start und Zielkonfiguration. Hierbei werden keine Fahrzeugparameter berücksichtigt. Erst der zweite Planungsschritt, welcher eine Bahn entlang der gefundenen Stützpunkte berechnet, berücksichtigt die Parameter Fahrzeugabmessung, maximaler Lenkwinkel und Geschwindigkeit. Auf diese Weise konnte ein Bahnplaner entwickelt werden, dem es möglich ist, auch bei komplizierten Umgebungen, schnell und effektiv eine Trajektorie zu finden, sollte diese für die gewählte Fahrzeugkonfiguration existieren.

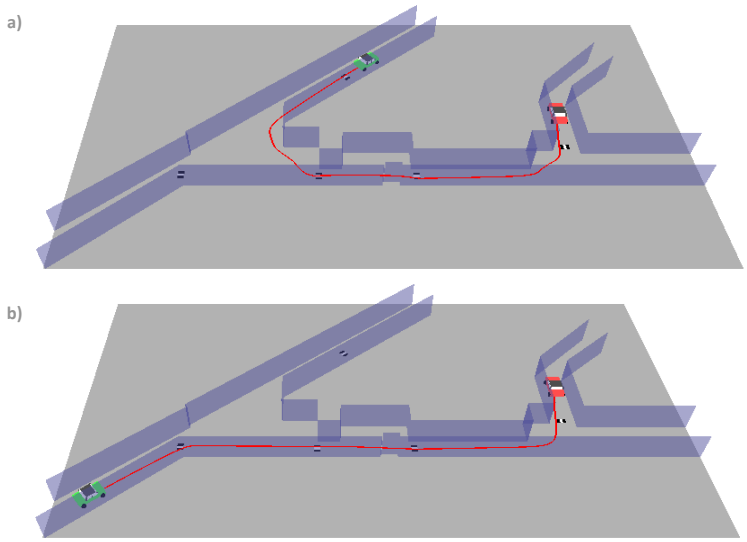


Abbildung 5.5.14: Geplante Trajektorien (Startposition: rot, Zielposition: grün)

Die Abbildung 5.5.14a und b zeigen die Ergebnisse der Bahnplanung für verschiedene Start- und Zielkonfigurationen innerhalb der definierten Umgebung. Unter anderem durch die Verwendung eines schnellen Kollisionstest mit dem „Separating-Axis-Theorem“ konnte ein sehr schneller Algorithmus entwickelt werden. Die Zeit, welche zur Bahnplanung benötigt wird, ist abhängig von der Komplexität und Länge der zu planenden Trajektorie und der Anzahl der hierfür benötigten Kollisionstests. Bei Versuchen mit verschiedensten Start- und Zielkonfigurationen in obiger Umgebung lag die Planungszeit einer Trajektorie zwischen $t_{min} = 174ms$ und $t_{max} = 741ms$. Die

Planungszeit liegt somit innerhalb der geforderten Grenzen für ein Online-Planungsverfahren.

5.6 Bahnregelungsalgorithmus

Drifts und Schlupf beim Beschleunigen und bei Kurvenfahrten oder Verzögerungen beim Einstellen des vorgegebenen Lenkwinkels, aufgrund einer begrenzten Geschwindigkeit des Lenkservomotors, führen zu Abweichungen der mobilen Roboterplattform von der geplanten Trajektorie. Aus diesem Grund wird ein Positionsregler zur Minimierung des Positionsfehlers $e(t)$ und des Orientierungsfehlers $\theta(t)$, während des Abfahrens der geplanten Bahn, benötigt.

Die verwendete mobile Roboterplattform basiert auf einem nach Ackermann spezifiziertem Typ [Haken08], wie dies in Kapitel 5.5 bereits beschrieben wurde.

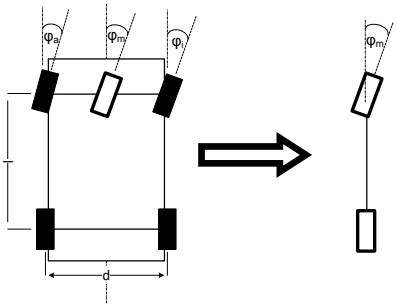


Abbildung 5.6.1: Vereinfachung des Ackermann-Antriebs

Dieser Fahrzeugtyp, in Abbildung 5.6.1 dargestellt, wird an seiner Hinterachse angetrieben. Die Vorderachse dient zur Lenkung.

Die Problematik bei der Verwendung von diesem Fahrzeugtyp besteht aus dem Sachverhalt, dass für einen Lenkwinkel φ ungleich Null die Radien der Kreisbahnen auf denen sich die inneren und äußeren Räder bewegen und somit der innere Lenkwinkel φ_i und äußere Lenkwinkel φ_a unterschiedlich groß sein müssen.

Nach [Haken08] kann jedoch ein mittlerer Lenkwinkel φ_m berechnet werden.

$$\varphi_m = \tan\left(\frac{d}{2l} + cot(\varphi_i)\right) = \tan\left(cot(\varphi_a) - \frac{d}{2l}\right) \tag{5.97}$$

Unter Zuhilfenahme von Gl.(5.97) kann der mobile Roboter als zweirädriges Modell, siehe Abbildung 5.6.2, dargestellt werden.

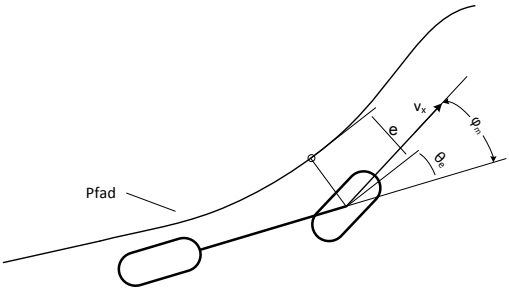


Abbildung 5.6.2: Skizze zur Roboterregelung

Die Aufgabe des implementierten Bahnregelalgorithmus ist die Minimierung des Positions- und Orientierungsfehler, indem abhängig von der Abweichung des Fahrzeugs der Lenkwinkel φ_m so eingestellt wird, dass der Positionsfehler $e(t)$ und der Orientierungsfehler $\theta_e(t)$, bezogen auf den geplanten Pfad, für $t \to \infty$ gegen Null geht. Die zur Robotersteuerung zur Verfügung stehenden Parameter sind der Lenkwinkel φ_m und die Geschwindigkeit v. Die Geschwindigkeit soll als konstant angenommen und nicht von dem Regler beeinflusst werden. Der einzige von der Regelung beeinflussbare Parameter ist somit der Lenkwinkel φ_m. Als Eingangsgrößen können, die Geschwindigkeit v, die Fahrzeugposition in (x,y)-Koordinaten, die Fahrzeugorientierung θ sowie deren Sollwerte dienen.

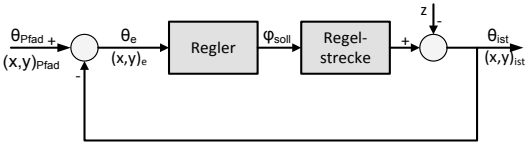


Abbildung 5.6.3: Blockschaltbild des Roboterreglers

Abbildung 5.6.3 zeigt das Blockschaltbild der Regelstrecke. Die Störgröße z entsteht zum Beispiel durch Unebenheiten des Bodens. Die Regelstrecke entspricht den Fahrzeugeigenschaften.

Der implementierte Regler nach [Thrun06] basiert auf einem nichtlinearen „feedforward"- Regelalgorithmus. Die Reglergleichung lautet:

$$\varphi_m(t) = \theta_e(t) + \arctan\left(\frac{k \cdot e(t)}{v_x(t)}\right) \tag{5.98}$$

Hierbei entspricht k dem Verstärkungsfaktor und $v_x(t)$ der Geschwindigkeit zum Zeitpunkt t in Richtung des Lenkwinkels und berechnet sich zu:

$$v_x(t) = v(t) \cdot \cos\varphi_m(t) \tag{5.99}$$

Gl.(5.98) besteht aus zwei Komponenten. Der erste Teil entspricht dem Lenkwinkelfehler:

$$\theta_e(t) = \varphi_m(t) - \theta_{path}(t) \tag{5.100}$$

Erhöht sich der Lenkwinkelfehler $\theta_e(t)$, erhöht sich auch der Lenkwinkel $\varphi_m(t)$ des Roboters, um einer Erhöhung von $\theta_e(t)$ entgegenzuwirken. Die zweite Komponente berechnet sich aus dem Arkustangens des Positionsfehlers $e(t)$ zu der Geschwindigkeit $v_x(t)$ in Lenkwinkelrichtung. Bei einer Zunahme des Positionsfehlers steigt dessen Einfluss auf den Lenkwinkel, während der Einfluss des Orientierungsfehlers abnimmt. Durch die Berücksichtigung der Geschwindigkeit in Lenkwinkelrichtung wird dieser so eingestellt, dass sich das Fahrzeug in Richtung des geplanten Pfads bewegt. In Folge der Abnahme des Positionsfehlers steigt der Einfluss des Orientierungsfehlers auf den einzustellenden Lenkwinkel in der Art, dass der Lenkwinkelfehler, der beim Minimieren des Positionsfehlers entstand, verringert wird und die Orientierung des Fahrzeugs sich der Soll-Fahrtrichtung annähert.

Nach [Thrun06] beträgt die Änderung des Positionsfehlers:

$$\dot{e}(t) = v(t) \cdot \sin\left(\arctan\left(\frac{ke(t)}{v(t)}\right)\right) \tag{5.101}$$

$$\dot{e}(t) = \frac{-ke(t)}{\sqrt{1 + \frac{ke(t)}{v(t)}}} \tag{5.102}$$

Für kleine Positionsfehler gilt:

$$e(t) = e(0) \cdot \mathrm{e}^{-kt} \tag{5.103}$$

Aus Gl.(5.103) folgt:

$$\lim_{t \to \infty} e(t) = 0 \tag{5.104}$$

Der Fehler $e(t)$ konvergiert nach Gl.(5.104) gegen Null. Hierbei entspricht der Faktor k nach Gl.(5.103) der Geschwindigkeit, mit welcher der Positionsfehler $e(t) \to 0$ konvergiert.

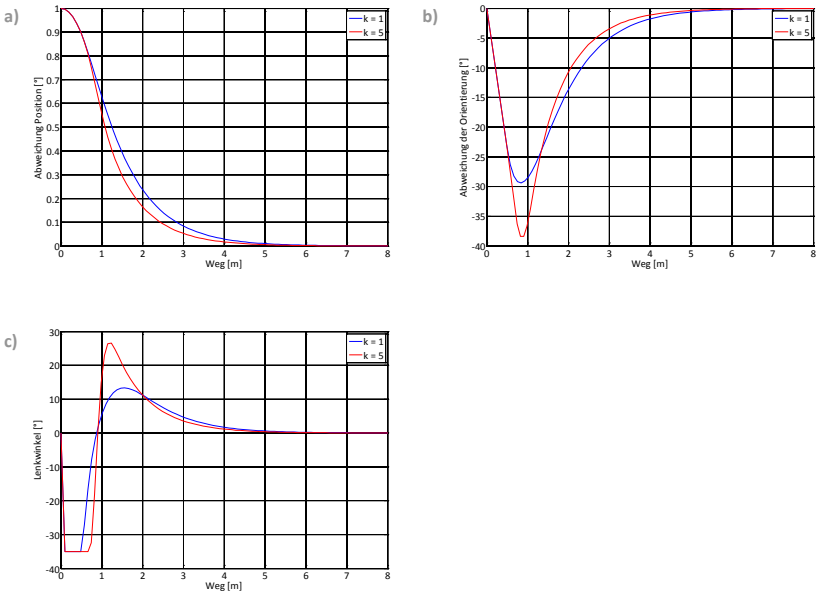


Abbildung 5.6.4: Reglersimulation a) Abweichung der Position
b) Abweichung der Orientierung und c) der eingestellte Lenkwinkel

Zur Überprüfung der Funktionsweise des Positionsreglers, zeigen die Abbildungen 5.6.4a-c die Simulation des implementierten „feedback"-Reglers mit verschiedenen Verstärkungsfaktoren, anhand eines vereinfachten Robotermodells. Der simulierte Roboter soll eine gerade Bahn in x-Richtung des Weltkoordinatensystems abfahren. Die Startposition des Roboters bei der Simulation ist $P_{start} = (0{,}0m; 1{,}0m)$. Dies entspricht, wie in Abbildung 5.6.4a gezeigt, einer Abweichung von der vorgegebenen Geraden um $e(0) = 1{,}0m$ in y-Richtung. Die Ausrichtung des Roboters, in Abbildung

5.6.4b dargestellt, entspricht der gewünschten Fahrtrichtung. Somit ist die Orientierungsabweichung $\theta_e(0) = 0$. Im Graphen von Abbildung 5.6.4c ist der Lenkwinkel φ_m des Fahrzeugs für verschiedene Verstärkungsfaktoren bezogen auf den zurückgelegten Weg eingetragen.

Für zunehmenden Verstärkungsfaktor k ist eine deutliche Zunahme der Geschwindigkeit, mit welcher der Positionsfehler ausgeglichen wird, erkennbar. Dies stimmt mit dem in Gl.(5.101) hergeleiteten Sachverhalt überein. Die maximale Geschwindigkeit der Positionsfehlerabnahme wird jedoch beschränkt durch den maximalen Lenkwinkel $\varphi_{m_{max}} = 35°$. Für eine Geschwindigkeitszunahme, bei der Verringerung des Positionsfehlers für hohe Verstärkungsfaktoren, spricht ebenfalls die Zunahme des maximalen Orientierungsfehlers während des Positionsfehlerausgleichs.

Nachdem das Funktionsprinzip des implementierten Positions- und Orientierungsreglers durch obige Simulation bewiesen wurde, wird der Regler im Folgenden an der realen mobilen Roboterplattform getestet. Die Startparameter entsprechen denen der durchgeführten Simulation. Die Startposition des Roboters beträgt $P_{start} = (0{,}0m; 1{,}0m)$ im Weltkoordinatensystem. Die Orientierungs-abweichung ist $\theta_e(0) = 0$. Dies beschreibt eine geplante Trajektorie entlang der x-Achse des Weltkoordinatensystems. In den Abbildungen 5.6.5a-c sind die Ergebnisse der geregelten Fahrt des realen Roboters zu sehen. Die Geschwindigkeit des mobilen Roboters beträgt $v = 2\,{}^{m}/_{s}$. Bei Betrachtung von Abbildung 5.6.5 ist zu erkennen, dass das System zunächst durch das Einstellen eines negativen Lenkwinkels reagiert. Dies entspricht einem nach rechts gerichteten Lenkeinschlag. Dies führt zu einer Verringerung des Positionsfehlers bei gleichzeitiger Erhöhung des Orientierungsfehlers. Ist der Positionsfehler unter eine von k abhängige Grenze gefallen, erhöht sich der Einfluss des Orientierungsfehlers auf den Lenkwinkel. Der implementierte Regler versucht durch entgegengesetztes Lenken den Orientierungsfehler zu minimieren. Die Messung am realen Roboter zeigt ein ähnliches Verhalten wie in der Simulation.

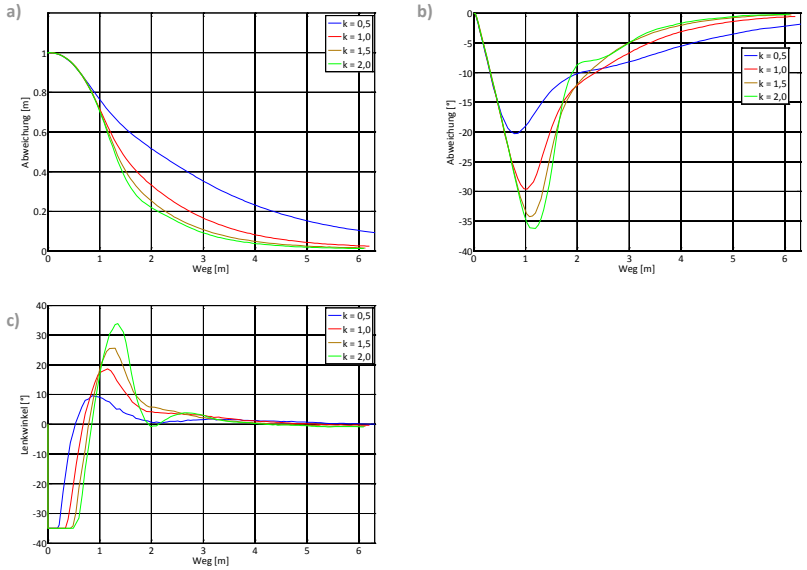


Abbildung 5.6.5: Roboterregelung für verschiedene Verstärkungsfaktoren k a) Abweichung der Position b) Abweichung der Orientierung und c) der eingestellte Lenkwinkel

Analog zu der Simulation führt auch im realen Messaufbau eine Erhöhung des Verstärkungsfaktors zu einem schnelleren Ausgleich des initialen Positionsfehlers. Jedoch ist beim Vergleich von Abbildung 5.6.4a und Abbildung 5.6.5a feststellbar, dass der Ausgleich des Positionsfehlers in der realen Messung wesentlich langsamer erfolgt. Dies liegt in der Systemträgheit begründet, welche zu einem verzögerten Einstellen des Lenkwinkels führt.

Auf den Abbildungen 5.6.5a-c sind die Messwerte von Positions- und Orientierungsabweichung sowie der Lenkwinkel für einen Verstärkungsfaktor $k = 0{,}1$ dargestellt. Zu erkennen ist lediglich eine geringe Abnahme des Positionsfehlers bei gleichzeitiger geringer Erhöhung des Orientierungsfehlers. In der Beschleunigungs- und Bremsphase zeigen sich die Einflüsse der aktuellen Geschwindigkeit auf den Lenkwinkel deutlich. Bei einer Geschwindigkeit $v \to 0$ kommt es nach Gl.(5.98) zu einer deutlichen Zunahme des Lenkwinkels, welches in Abbildung 5.6.6c erkennbar ist. Im Startmoment für eine Geschwindigkeit von $v = 0$ ist die Gleichung nicht definiert, beziehungsweise geht der Lenkwinkel $\varphi_m \to \infty$, welches jedoch durch die Robotersteuerung auf den maximalen Lenkeinschlag von $\varphi_m = \pm 35°$ begrenzt wird.

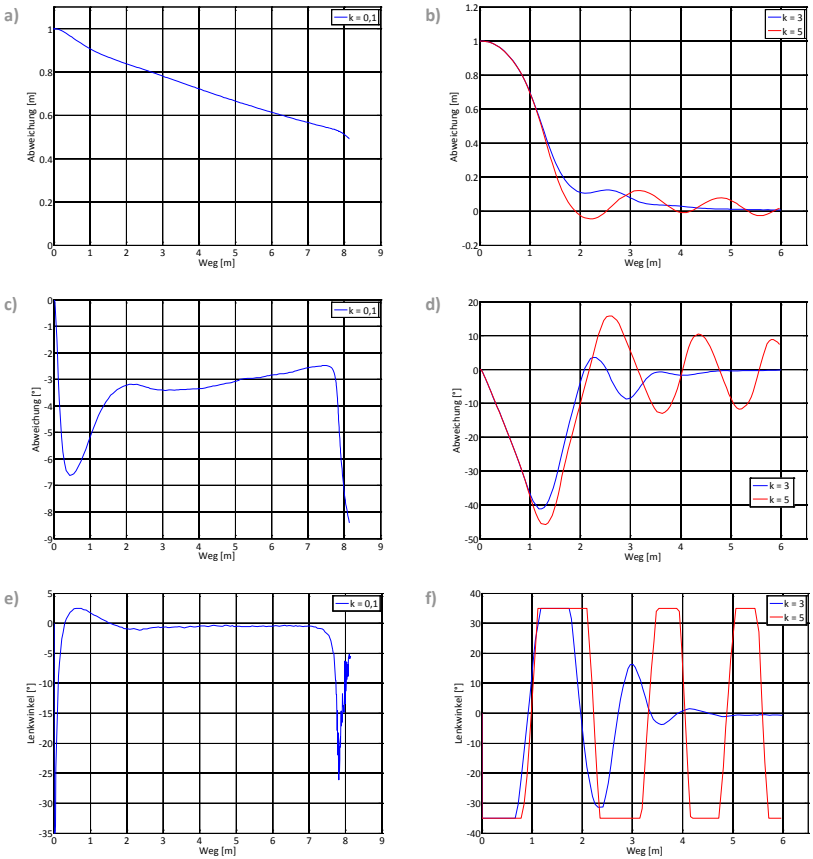


Abbildung 5.6.6:
Roboterregelung für zu niedrigen Verstärkungsfaktor:
a) Positionsfehler b) Orientierungsfehler c) Lenkwinkel und für zu hohen Verstärkungsfaktor: d) Positionsfehler e) Orientierungsfehler f) Lenkwinkel

Die Abbildungen 5.6.6 d-f zeigen den Einfluss eines zu hohen Verstärkungsfaktor k auf die Parameter Position, Orientierung und Lenkwinkel. Bereits für kleine Positionsabweichungen reagiert die Regelung mit einem betragsmäßig höheren Lenkwinkel um der Positionsabweichung entgegen zu steuern. Dies resultiert in einer schnellen Abnahme des Positionsfehlers, jedoch führt dies gleichzeitig zu einer starken Zunahme des Orientierungsfehlers, so dass diesem aufgrund der Systemträgheit nicht

rechtzeitig entgegengewirkt werden kann. Es kommt zu einem Überschwingen des Positions- und Orientierungsfehlers.

Solange sich der Verstärkungsfaktor k in definierten Grenzen bewegt, kann davon ausgegangen werden, dass das Regelsystem ein stabiles Verhalten aufweist. Für den obigen Fall empfiehlt es sich einen Verstärkungsfaktor in den Grenzen:

$$0{,}5 \leq k \leq 2$$

zu wählen. Für den Reglertest wurde die Sprungantwort auf einen Positionsfehler von einem Meter untersucht. Beim Abfahren einer vorgegebenen Bahn ist solch eine hohe Abweichung jedoch unwahrscheinlich und tritt gegebenenfalls nur nach der Lokalisierung der Startposition auf. Für ein möglichst exaktes Abfahren der vorgegebenen Trajektorie empfiehlt es sich daher, einen höheren Verstärkungsfaktor zu wählen, welcher unter Umständen dynamisch verändert werden kann.

5.7 Erkennung von Hindernissen

Die Implementierung einer schnellen und robusten Hinderniserkennung aus dreidimensionalen PMD-Bilddaten ist ein elementarer Bestandteil dieser Arbeit. Alle Hindernisse müssen in Echtzeit aus den Bilddaten extrahiert und auf eine eventuelle Kollision überprüft werden. Sollte eine mögliche Kollision erkannt werden, muss die Hinderniserkennung die Planung einer neuen kollisionsfreien Trajektorie auslösen. Das hierzu entwickelte Verfahren kann in die Abschnitte Datenvorverarbeitung beziehungsweise Datenaufbereitung, Extraktion möglicher Hindernisse aus den 3D-Bilddaten, Berechnung von Hindernispositionen und Kollisionsprüfung eingeteilt werden.

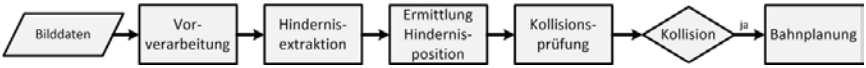


Abbildung 5.7.1:Datenflussdiagramm zur Erkennung von Hindernissen

Abbildung 5.7.1 zeigt das Datenflussdiagramm des entworfenen Algorithmus. Für jedes PMD-Bild wird dieses erneut durchlaufen.

5.7.1 Vorverarbeitung der 3D-Bilddaten

Aufgrund des in Kapitel 5.1 beschriebenen Versuchsaufbaus, wurde die PMD-Kamera mit einem Neigungswinkel ungleich Null am Roboter angebracht. Auf diese Weise wird der Fahrweg innerhalb der PMD-Aufnahme wiedergegeben. Folglich ist die Erkennung von eventuell auftauchenden Objekten im PMD-Bild ohne Weiteres nicht möglich. Eine Unterscheidung zwischen Fahrbahn und Objekt kann nur schwer getroffen werden.

Dies führt dazu, dass vor der eigentlichen Hinderniserkennung zunächst die Position des Fahrwegs innerhalb des PMD-Bilds ermittelt werden muss. Hierzu werden, mittels der in Kapitel 5.3 hergeleiteten Transformationsmatrix ${}^{RKS}T_{Kam}$ zwischen Kamera und Roboter, alle PMD-Bildpunkte in das Roboterkoordinatensystem transformiert. Es gilt:

$$ {}^{RKS}\vec{p} = {}^{RKS}T_{Kam} \cdot {}^{Kam}\vec{p} \tag{5.105}$$

Die Transformation der Daten nach Gl.(5.105) ermöglicht die Filterung des Fahrwegs aus den PMD-Bildern, indem die Grenzen $z_{off,-}$ und $z_{off,+}$ in z-Richtung des Roboterkoordinatensystems definiert werden. Alle Bildpunkte $\vec{p}_i$ für die gilt:

$$ z_{off,-} < |\vec{p}_i| < z_{off,+} \tag{5.106}$$

repräsentieren den Fahrweg und werden in einer Befahrbarkeitsmatrix A als „frei“ und somit als befahrbar definiert. Alle Pixel, deren z-Wert nicht innerhalb der Schranken liegt, werden als besetzt markiert und können mögliche Hindernisse darstellen. Pixel unterhalb der Grenze $z_{off,-}$ sind negative Hindernisse, wie zum Beispiel Treppen oder tiefe Löcher. Positive Hindernisse sind Hindernisse, deren z-Wert größer als die definierte Grenze $z_{off,+}$ ist. Dies können zum Beispiel Wände oder Personen sein die im Bildbereich der Kamera auftauchen. Problematisch sind jedoch Pixel, die aufgrund von Fehlmessungen der Kamera, zum Beispiel durch einen schlecht reflektierenden Boden, Messwerte liefern, die nach Gl.(5.106) nicht innerhalb dieser Grenzen liegen. Damit diese nicht zur Objektdetektion verwendet werden und somit die Objekterkennung verfälschen, wird zusätzlich, um einzelne als besetzt angenommene Pixel zu eliminieren, auf die bisher berechnete Befahrbarkeitsmatrix A das Median-Filter angewandt.

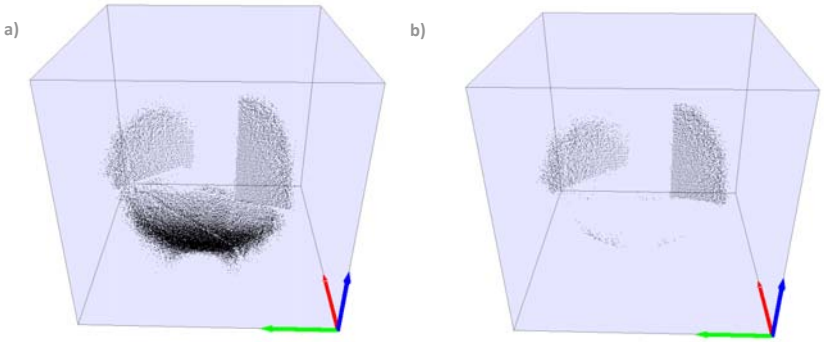


Abbildung 5.7.2: Datenvorverarbeitung zur Detektion von Hindernissen

Ist das Fahrzeug nahe an einem Objekt, kann es im schlechtesten Fall vorkommen, dass alle Pixel der Kamera das Objekt repräsentieren. Bei der Verwendung der LynCube mit einer lateralen Auflösung von 200x200 Pixeln würde dies bedeuten, dass insgesamt 40.000 Pixel zur Objekterkennung berücksichtigt werden. Dies würde zu einer Zunahme der Berechnungsdauer führen. Damit diese aber weitestgehend unabhängig von der Größe des Objekts im Bild ist, werden der Hindernisberechnung lediglich alle Randpixel des Objekts zugeführt. Als Randpixel werden die Pixel definiert, welche nur einen direkten Nachbarn, der als besetzt markiert wurde, in vertikaler Bildrichtung haben oder als besetzt markierte Pixel die maximal 5 Pixel von dem Rand entfernt liegen.

5.7.2 Hinderniserkennung

Basierend auf den im vorherigen Kapitel 5.7.1 extrahierten Pixeln, können Objekte dreidimensional aufgenommen und deren Position bezüglich des Roboterkoordinaten-systems ermittelt werden.

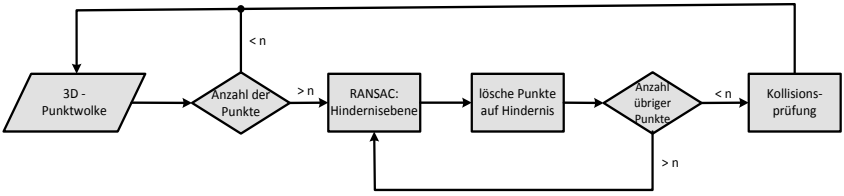


Abbildung 5.7.3: Datenflussdiagramm zur Berechnung der Hindernisposition

Das Datenflussdiagramm in Abbildung 5.7.3 zeigt den Verlauf der Objekt- beziehungsweise Hinderniserkennung. Ausgehend von der ermittelten 3D-Punktwolke, die die Randpunkte möglicher Objekte enthält, wird versucht eine oder mehrere Ebenen in die Punktwolke zu legen. Die Berechnung der entsprechenden Ebenengleichungen basiert auf dem Ransac-Algorithmus [Fischler81] in Verbindung mit einer Least-Mean-Square (LMS)-Näherung von Ebenengleichungen, wie dies bereits in Kapitel 5.3.1 für die Berechnung der Roboter-Kamera Kalibrierung vorgestellt wurde. Ist die Ebene gefunden, welche durch die meisten Punkte der 3D-Aufnahme genähert werden kann, werden alle zur Berechnung verwendeten 3D-Daten gelöscht. Anschließend wird versucht eine weitere Ebene in die verbleibenden Punkte zu legen. Mit der Berechnung von weiteren Ebenen wird solange fortgefahren, bis die Anzahl der übrigen Punkte, die nicht zur Ebenenberechnung berücksichtigt wurden, eine vorher definierte Anzahl n unterschreitet. Hindernisse werden definiert durch den Normalenvektor der jeweiligen Hindernisebene und deren Begrenzungspunkte in x, y und z-Richtung.

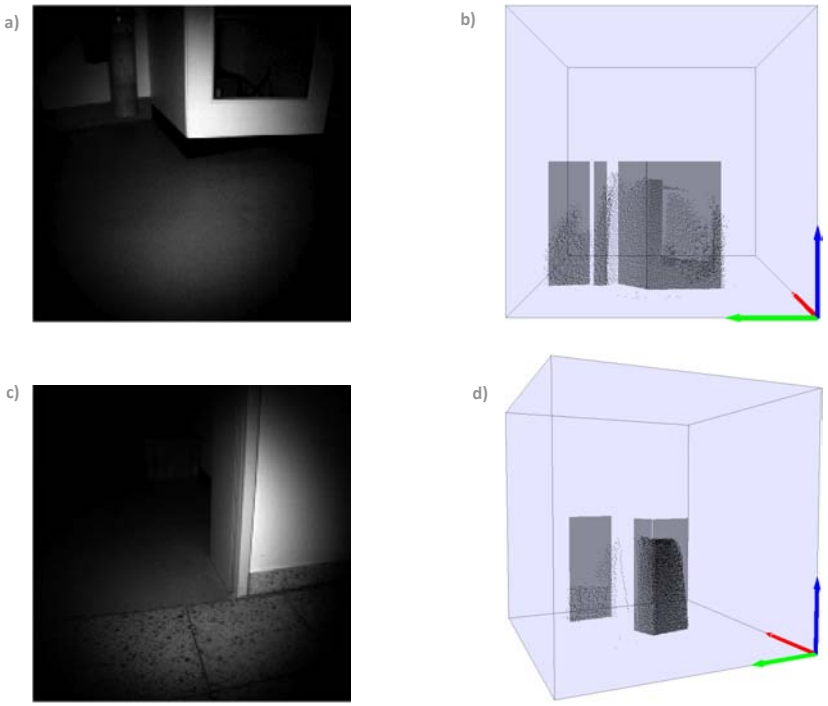


Abbildung 5.7.4: Beispiele für erkannte Hindernisse

Die Abbildungen 5.7.4 a-d zeigen zwei verschiedene Szenen, für welche die Hindernisse berechnet wurden. Die Abbildungen 5.7.4 a und c zeigen das jeweilige Intensitätsbild der aufgenommenen Szenen, die Abbildungen 5.7.4 b und d die entsprechenden dreidimensionalen Punktwolken. Es ist zu erkennen, dass alle in den Bildern befindliche Hindernisse richtig detektiert wurden.

Aufgrund der Filterung verrauschter Bildpunkte, sowie der Filterung des Fahrwegs konnte ein sehr präziser und robuster Algorithmus entworfen werden, welcher die Detektion von Hindernissen mit einer PMD-Kamera in Echtzeit gewährleistet. Als problematisch stellt sich die Erkennung schlecht reflektierender Objekte dar. Hier kann es vorkommen, dass die PMD-Kamera keine Entfernungswerte für das betreffende Objekt liefert.

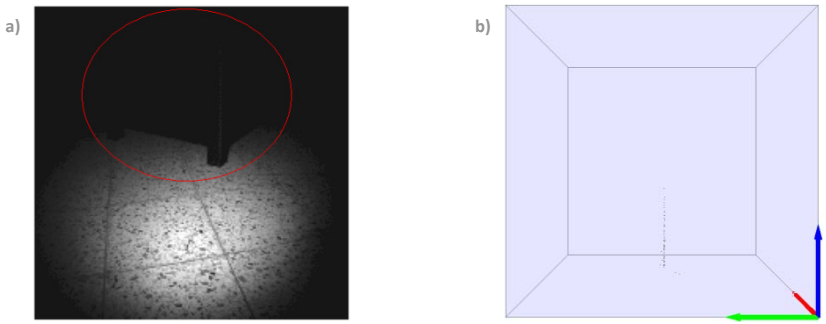


Abbildung 5.7.5: Fehlerkennung bei schlecht reflektierenden Objekten
a) Originalbild b) 3D-Daten nach Filterung

Abbildung 5.7.5a zeigt das Intensitätsbild eines Sessels, welcher im Sichtbereich der PMD-Kamera auftaucht. Bei Betrachtung der Abbildung 5.7.5b, welche die 3D-Punktwolke der gleichen Szene zeigt, ist zu erkennen, dass die Kamera aufgrund des schlecht reflektierenden Sessels keine brauchbaren Entfernungswerte für die Interpretation der Szene liefert. Die Erkennung und Positionsbestimmung des Hindernisses kann in dieser Situation nicht erfolgen. Erst bei sehr geringem Abstand zum Hindernis trifft genug moduliertes Licht auf den PMD-Sensor, so dass das Hindernis erkannt wird. In der Praxis führt dies dazu, dass die Berechnung einer kollisionsfreien Trajektorie zeitlich nicht möglich ist. Der Roboter bleibt vor dem Hindernis stehen, bevor eine neue Trajektorie gefunden werden kann. Eine Verbesserung des Kameramoduls, speziell eine Verbesserung der Beleuchtungs-intensität, ist hier die einzige Möglichkeit, auch schlecht reflektierende Objekte früh genug zu erkennen. Um Kollisionen dennoch zu vermeiden, wurde eine softwareseitige Aktivierung des Notausschalters implementiert,

welche zum sofortigen Blockieren der Antriebsräder führt und so den mobilen Roboter in nur wenigen Zentimetern kollisionsfrei zum stehen bringt.

5.7.3 Überprüfung auf Kollision

Wurden ein oder mehrere Hindernisse in dem PMD-Bild erkannt, müssen diese durch Abgleich mit der Umgebungskarte und der geplanten Trajektorie auf eine mögliche Kollision überprüft werden. Diesbezüglich wurde ein mehrstufiges Verfahren entwickelt, welches im Datenflussdiagramm in Abbildung 5.7.6 skizziert ist.

Zuerst erfolgt die Überprüfung, ob das detektierte Objekt in der Umgebungskarte des mobilen Roboters vorhanden ist. Dies könnte zum Beispiel eine bereits eingetragene Wand oder ein sonstiges bekanntes Hindernis sein.

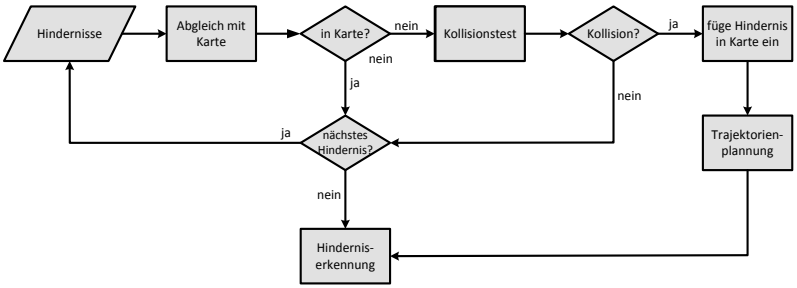


Abbildung 5.7.6: Datenflussdiagramm zur Kollisionsüberprüfung und –Vermeidung

Die Überprüfung, ob ein Objekt bereits in der Karte verzeichnet ist, geschieht durch die Bestimmung des Differenzwinkels $\Delta\vartheta$ zwischen den Normalenvektoren und des Abstands d_{max} zwischen dem gefundenen Objekt und den Hindernissen im relevanten Quadtree der Umgebungskarte. Da der Normalenvektor der Hindernisebenen immer in die Richtung des befahrbaren Raumes zeigt, gilt für den Differenzwinkel $\Delta\vartheta$:

$$\Delta\vartheta = \arccos\left(\frac{\vec{n}_{Obs}\cdot\vec{n}_{Obj}}{|\vec{n}_{Obs}|\cdot|\vec{n}_{Obj}|}\right) \tag{5.107}$$

wobei $\vec{n}_{Obs}$ der Normalenvektor eines Hindernisses in der Umgebungskarte und dementsprechend $\vec{n}_{Obj}$ der Normalenvektor des detektierten Objekts ist.

Der maximale Abstand zwischen erkanntem Objekt und dem jeweiligen Hindernis, projiziert in die xy-Ebene des Weltkoordinatensystems, entspricht:

$$d_{max} = \max\left(\left|\vec{n}_{Obs} \cdot \vec{p}_{Obj_i} - \vec{n}_{Obs} \cdot \vec{p}_{Obs}\right|\right) \ , \qquad i = 1 \dots 4 \qquad (5.108)$$

Gilt:

$$|\Delta\vartheta| < \Delta\vartheta_{limit} \qquad \text{und} \qquad d_{max} < d_{limit}$$

ist das detektierte Objekt bereits Bestandteil der in der Karte eingetragenen Hindernisse. Dies ist der Fall, wenn der maximale Abstand und der Winkel zwischen den Normalenvektoren innerhalb definierter Grenzen liegen. In der Praxis wird:

$$\Delta\vartheta_{limit} = 5°$$

$$d_{limit} = 0{,}10m$$

gewählt. Dieser Vorgang wird wiederholt, bis alle Objekte des aktuellen Bildes überprüft sind. Sollte ein detektiertes Hindernis nicht in der Karte eingetragen sein, wird dieses auf Kollision mit der aktuell geplanten Trajektorie geprüft. Dies geschieht in dem alle Roboterpositionen der Trajektorie mit dem Separating-Axis/Plane-Theorem auf Kollision getestet werden. Wird eine Kollision festgestellt, wird das gefundene Objekt in die Umgebungskarte eingetragen und anschließend eine neue Route zum Zielpunkt geplant. Die Hinderniserkennung bricht an dieser Stelle ab und wird erneut ausgeführt, sobald eine neue PMD-Aufnahme zur Verfügung steht.

5.8 Reaktive Bahnplanung

In den vorangegangenen Kapiteln, über die Entwicklung des Bahnplaners, sowie der Erkennung von Hindernissen mit Hilfe der PMD-Kamera, wurde die Basis für die Entwicklung eines neuartigen, auf PMD-Daten basierenden, reaktiven Bahnplaners gelegt. In diesem Kapitel werden im Folgenden die behandelten Bausteine zusammengefügt, so dass es dem mobilen Roboter ermöglicht wird, autonom auf Hindernisse zu reagieren und alternative Trajektorien nahezu in Echtzeit zu berechnen.

5.8.1 Erweiterung der Umgebungskarte

Für die Realisierung einer auf PMD-Daten basierenden reaktiven Bahnplanung, muss die Umgebungsrepräsentation in einer Art erweitert werden, dass es möglich ist Objekte, welche durch die Hinderniserkennung detektiert werden, in die Umgebungskarte aufzunehmen und dem Bahnplanungsalgorithmus zur Verfügung zu

stellen. Allerdings gestaltet es sich als problematisch, die mit der Kamera aufgenommenen und noch nicht vorhandenen Hindernisse ohne Weiteres in die Umgebungskarte aufzunehmen. Bewegliche Objekte werden als feste Hindernisse in die Karte aufgenommen und beeinflussen langfristig die Bahnplanung, obwohl diese nicht mehr vorhanden sind, sollte der Roboter ein weiteres Mal an der gleichen Stelle vorbeifahren. Zur Lösung dieser Problematik sind mehrere Ansätze vorstellbar.

Die Berechnung des optischen Flusses innerhalb der PMD-Aufnahmen stellt ein adäquates Mittel dar. Ist der optische Fluss für das erkannte Hindernis stark abweichend von denen der restlichen Pixel, könnte auf ein sich bewegendes Objekt geschlossen werden. Die Berechnung des optischen Flusses nach [Berthold81] gestaltet sich allerdings, unter Verwendung der verbauten Kamera, als schwierig. Hierzu wird die Erkennung von Merkmalen, die den Pixeln in zwei aufeinander folgenden Bildern zugeordnet werden können, benötigt. Aufgrund des relativ geringen Sichtfeldes der PMD-Kamera und der hohen zu fahrenden Geschwindigkeit, welches in Kombination zu unscharfen Grauwertbildern führen kann, ist die Berechnung des optischen Flusses stark fehlerhaft, da nur wenige korrespondierende Punkte in den Bildern ermittelt werden können.

Selbst das langfristige Speichern von statischen Hindernissen würde bei mehrmaligem Abfahren der Strecke zwischen Start und Ziel dazu führen, dass keine Überprüfung auf noch Vorhandensein, von während der Laufzeit in die Karte eingetragenen Hindernissen, stattfinden kann, da diese aufgrund einer veränderten Trajektorie nicht mehr im Sichtbereich der Kamera erscheinen. Wenn weitere Hindernisse auf der alternativen Bahn detektiert werden, erhöht sich die Wahrscheinlichkeit, dass keine Trajektorie zum Ziel gefunden werden kann, obwohl die ursprüngliche wieder passierbar ist.

Aus den genannten Gründen wird auf die Speicherung von erkannten Hindernissen in der Umgebungskarte verzichtet. Stattdessen wird eine zusätzliche Hinderniskarte erstellt, in der alle erkannten Objekte, sollten diese nicht in der Umgebungskarte verzeichnet sein, eingetragen werden. Dies bietet den Vorteil, dass alle Hindernisse nach Erreichen der Zielposition wieder gelöscht werden können, ohne auch die Objekte in der Umgebungskarte zu löschen. Hindernisse beeinflussen dann, unabhängig davon ob sie beweglich sind oder nicht, nur die Trajektorie zu dem Zeitpunkt in dem sie sich im Sichtbereich der Kamera befinden.

Eine langfristige Speicherung kann nur zum Erstellen einer Umgebungskarte erfolgen, sollte keine CAD-Karte vorhanden sein. Hierbei ist jedoch auszuschließen, dass bewegte Objekte die Fahrbahn kreuzen.

5.8.2 Erstellung einer Hinderniskarte

Die Vorbetrachtung in Abschnitt 5.8.1 haben gezeigt, dass die Erstellung einer geeigneten Hinderniskarte nötig ist. Ebenso wie die aus einer CAD-Karte der Roboterumgebung erstellte Umgebungskarte, muss die Hinderniskarte sowohl aus einer metrischen Repräsentation der Roboterumgebung als auch aus einer topologischen Karte bestehen.

Im ersten Planungsschritt findet, siehe Kapitel 5.5.3, eine topologische Bahnplanung statt, welche eine Verbindung zwischen Start- bzw. Ist-Position und der Zielposition über die bestmögliche Verbindung von Stützpunkten sucht. Zur Erstellung der topologischen Hinderniskarte werden für jedes Hindernis Stützpunkte berechnet. Diese befinden sich in einem definierten Abstand auf beiden Seiten neben dem berechneten Hindernis. Der Abstand ergibt sich aus einem geforderten Mindestabstand des Roboters zu den jeweiligen Hindernissen. Dieser setzt sich aus den Abmessungen des mobilen Roboters und einem Sicherheitsabstand zusammmen. Der Sicherheitsabstand wird aus empirischen Werten ermittelt. Sind die zu einem Hindernis gehörenden Stützpunkte berechnet, werden diese mit den Stützpunkten der topologischen Umgebungskarte verknüpft. Alle möglichen Verbindungen zu diesen, die keine Hindernisse beider Karten schneiden, werden in der topologischen Hinderniskarte gespeichert.

Bei der Erstellung der metrischen Hinderniskarte wird diese, zur Beschleunigung der bei der Bahnplanung nötigen Kollisionsberechnungen, in die gleiche Quadtree-Struktur aufgeteilt wie die Umgebungskarte. Neue Hindernisse werden in die entsprechenden Quadtrees, mit Informationen über deren Normalenvektoren und Begrenzungspunkte, eingetragen.

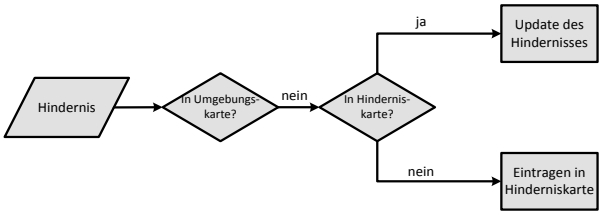


Abbildung 5.8.1: Datenflussdiagram zum Hinzufügen von Hindernissen in die Karte

Werden Objekte innerhalb des Sichtbereichs der Kamera detektiert, erfolgt das Hinzufügen von möglichen Hindernissen in die Karte entsprechend dem Flussdiagramm in Abbildung 5.8.1. Zunächst wird nach dem in Abschnitt 5.7.2 beschriebenen Verfahren überprüft, ob sich das Objekt in der Umgebungskarte befindet. Ist dies der Fall wird mit dem nächsten Objekt fortgefahren. Sollte das Objekt jedoch nicht in der Karte verzeichnet sein, erfolgt die Überprüfung, ob das Objekt oder Teile des Objekts in der Hinderniskarte eingetragen sind. Dies geschieht durch Überprüfung des Winkels ϑ zwischen den Normalenvektoren der Objektflächen und den bekannten Hindernissen. Gilt für den Winkel

$$|\vartheta| < \vartheta_{limit} = 5°$$

wird im nächsten Schritt der Abstand zwischen diesen bestimmt. Schneidet sich das mögliche Hindernis mit einem der Hindernisse in der Karte oder sollte der minimale Abstand im Bereich von nur wenigen Zentimetern liegen, werden beide durch Mittelung des Normalenvektors sowie der Erweiterung des Hindernisses auf die äußersten Randpunkte zusammengefasst.

5.8.3 Neuplanung der Trajektorie

Ist die Hinderniskarte aktualisiert, gilt es im nächsten Schritt unter Einbeziehung beider Karten eine neue Trajektorie zum Ziel zu planen. Den hierbei eingesetzten Bahnplaner gilt es hinsichtlich der für die Planung benötigten Rechenzeit zu optimieren, so dass ein flüssiges Fahren des mobilen Roboters, trotz auftauchender Hindernisse gewährleistet ist. Ein Anhalten des Roboters während der Neuplanungsphase sollte vermieden werden. Des Weiteren muss der mobile Roboter die Möglichkeit haben, nach erfolgloser Bahnplanung, rechtzeitig vor einer Kollision zum Stehen zu kommen. Versuche haben gezeigt, dass mit den entwickelten Bahnplanungsverfahren in Kapitel 5.5.5, abhängig von der Umgebung, Planungszeiten von $t < 800ms$ möglich sind. Hier besteht zur Realisierung eines reaktiven Bahnplaners Optimierungsbedarf. Sollte keine

Bahn existieren, müssen Abbruchkriterien für die Trajektorienplanung gefunden werden. Ohne diese würde mit der Expansion möglicher Folgepositionen fortgefahren werden, welches zum Überlauf des Arbeitsspeichers des verwendeten Computers führen würde. Dies gilt es in jedem Fall zu vermeiden. Unter Einbeziehung dieser Vorüberlegung wurde die Limitierung des Arbeitsspeichers, also die Anzahl der maximal zu explorierenden Positionen, sowie die maximale Berechnungsdauer des Bahnplaners als Abbruchkriterien festgelegt. Diese werden während der Planungsphase überwacht. Beim Überschreiten dieser Parameter wird die Bahnplanung sofort abgebrochen, der Bremsvorgang wird eingeleitet und der Roboter fährt erst weiter, wenn das Hindernis entfernt wurde und der Anwender den mobilen Roboter neu startet.

Wurde von der Kamera ein Hindernis detektiert und in die Hinderniskarte eingetragen, muss vor dem Start der eigentlichen Trajektorienplanung zunächst die Ausweichrichtung des mobilen Roboters bestimmt werden.

Diesbezüglich werden im Roboterkoordinatensystem die Ausdehnung des Hindernisses in y-Richtung, welches der Ausdehnung quer zur Fahrzeugrichtung entspricht, sowie Winkel und Abstand des mobilen Roboters zum Hindernis bestimmt. Abhängig von diesen Parametern wird die Wahl der Ausweichrichtung getroffen. Gilt für den Abstand d zum Hindernis:

$$d > d_{max}$$

wobei d_{max} ein auf Tests basierender Wert ist, welcher abhängig von Fahrzeuggröße und maximalem Lenkwinkel ist, wird die Ausweichrichtung in Richtung der geringsten Ausdehnung des Hindernisses gewählt. Bei sehr geringem Abstand zum Hindernis wird zusätzlich der berechnete Winkel α zwischen Hindernis und Fahrzeug berücksichtigt.

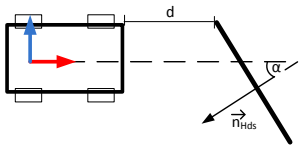


Abbildung 5.8.2: Skizze zur Bestimmung der Ausweichrichtung

Gilt für den Winkel α zwischen Fahrtrichtung $\vec{x}$ und dem Normalenvektor des Hindernisses $\vec{n}_{Hds}$:

$$\propto > 45°$$

Entspricht die Ausweichrichtung der Richtung des auf die y-Achse des Roboterkoordinatensystem projizierten Normalenvektors des Hindernisses $\vec{n}_{Hds}$.

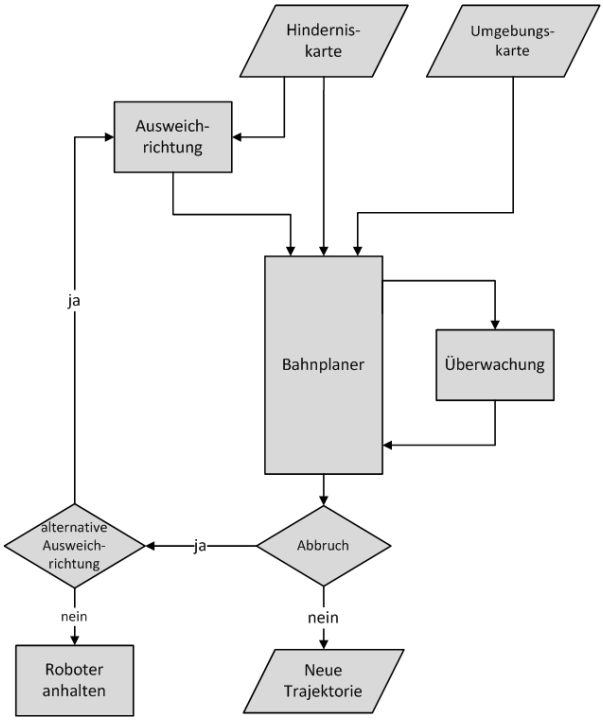


Abbildung 5.8.3: Datenflussdiagramm des reaktiven Bahnplaners

Zusammenfassend gestaltet sich die reaktive Bahnplanung wie in Abbildung 5.8.3 dargestellt. Wird ein Hindernis detektiert und in die Hinderniskarte eingetragen, wird mit der Ermittlung der Ausweichrichtung begonnen. Diese dient zur Initialisierung des Bahnplanungsalgorithmus. Unter der Berücksichtigung von Umgebungs- und Hinderniskarte wird eine kollisionsfreie Bahn berechnet. Während der Berechnung werden die bereits definierten Abbruchkriterien überwacht. Sind diese erfüllt, führt dies zum Abbruch der Bahnplanung. Sollte die alternative Ausweichrichtung noch nicht untersucht worden sein, wird der Bahnplaner erneut gestartet. Wird hierbei ebenfalls kein Weg gefunden, wird der Roboter gestoppt.

Das verwendete Planungsverfahren basiert auf modifizierten Ergebnissen des in Kapitel 5.5 entwickelten zweistufigen Bahnplaners. Während bei der Trajektorienplanung lediglich Start- und Zielpunkt der Trajektorie vorgeben waren, wird hier zusätzlich der erste Zwischenknoten für die Berechnung der kürzesten Verbindung über die in der Karte verzeichneten Stützpunkte vorgeschrieben. Erster Zwischenknoten ist der in die Hinderniskarte eingetragene Stützpunkt des detektierten Hindernisses in Ausweichrichtung. Die Verbindungsgerade zwischen Start- und Stützpunkt wird vor der Ausführung des Bahnplaners auf Kollision überprüft. Sollte diese in beiden Karten keine Objekte schneiden wird analog der Bahnplanung aus Kapitel 5.5 fortgefahren.

Bei der Planung von langen Strecken und der hiermit höheren Anzahl an Wegpunkten kommt es zu einer Zunahme der Planungsdauer. Dies führt im schlechtesten Fall zur Kollision mit einem Hindernis, beziehungsweise zu einem Abbruch der Bahnplanung durch die Überwachung der maximalen Planungsdauer, obwohl ein Pfad zum Ziel existiert.

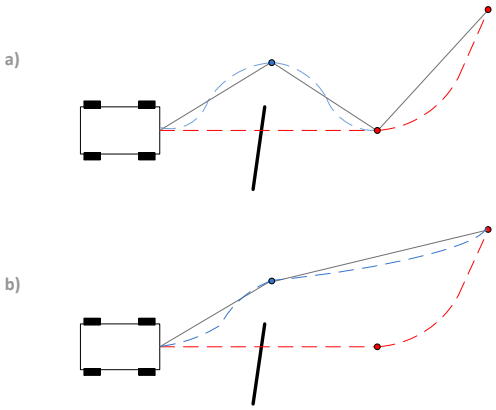


Abbildung 5.8.4: Methoden zur Beschleunigung der reaktiven Bahnplanung: a) Hindernisumfahren b) finden eines optimalen Pfads

Zum Planen der neuen Trajektorie stehen zwei Lösungsansätze zur Verfügung. Der erste Ansatz ist die Planung eines Weges um das Hindernis herum zum nächsten erreichbaren Wegpunkt der ursprünglich geplanten Bahn, wie es in Abbildung 5.8.4a dargestellt ist. Diese Methode dient der Verkürzung der Planungsdauer, da nur ein Teilstück der Bahn neu berechnet werden muss, und führt zur Lösung der angesprochenen Probleme. Nachteilig hierbei ist, dass die geplante Bahn keine Optimalität bezüglich der Weglänge aufweist. Dies bietet allerdings eine vollständige

Neuberechnung der Trajektorie, wie es in Abbildung 5.8.4b dargestellt ist. Eine Optimierung der Planungsdauer kann durch Aufteilung der Planung in zwei Abschnitte erreicht werden. Zuerst wird ein Weg in der topologischen Karte geplant. Ist eine Route gefunden, wird die Planung von der aktuellen Roboterposition, durch Einbeziehung der metrischen Umgebungsdarstellung unter Berücksichtigung der Robotermaße, bis zum ersten bei der topologischen Planung ermittelten Stützpunkt berechnet. Dieser Wegabschnitt wird unmittelbar an die Steuerung des mobilen Roboters übertragen. Während der Planung des nächsten Abschnitts, kann der Roboter den bis dahin berechneten Teilabschnitt abfahren. Neu geplante Abschnitte werden der Trajektorie hinzugefügt. Problematisch hierbei ist jedoch, dass zwar ein Weg in der topologischen Karte vorhanden sein kann, dies aber nicht bedeutet, dass der Weg auch unter Berücksichtigung der Robotergeometrie in der metrischen Karte gefunden werden kann. Dies kann dazu führen, dass das autonome Fahrzeug in eine Sackgasse gelangt. Trotzdem wurde dieses Verfahren zur reaktiven Bahnplanung implementiert, denn der Nachteil aufgrund der Fälle in denen ein Weg in der topologischen Karte gefunden wird, jedoch nicht in der metrischen Umgebungskarte, ist hinsichtlich der Beschleunigung des Bahnplanungsverfahrens zu vernachlässigen. Dies kann sich allerdings ändern, sobald der Roboter in einer komplex strukturierten Umgebung fährt.

5.8.4 Ergebnisse der reaktiven Bahnplanung

Durch die Kombination der PMD-Kamera unterstützten Hinderniserkennung und des entwickelten Bahnplanungsverfahren, wird es dem fahrerlosen Transportsystem ermöglicht, plötzlich auftauchenden Hindernissen autonom auszuweichen. Abbildung 5.8.5 und Abbildung 5.8.6 zeigen, zur Verdeutlichung der Funktionsweise der entwickelten Methoden, das Abfahren einer geplanten Trajektorie von Start- zur Zielposition.

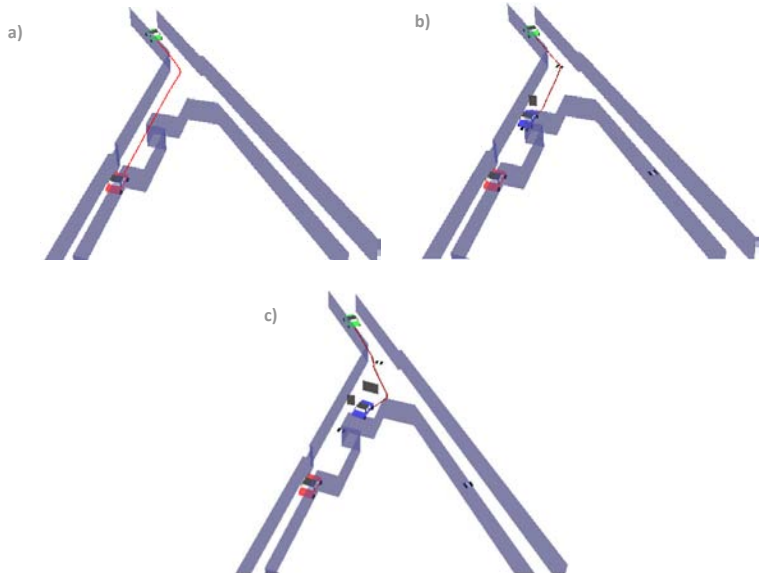


Abbildung 5.8.5: Grafische Ausgabe des Bahnplaners a) zum Start b) nach Erkennen des ersten Hindernisses c) nach Erkennen des zweiten Hindernisses

Abbildung 5.8.5a zeigt die anfangs geplante Trajektorie. Das rote Fahrzeug entspricht hier der Startposition und das grüne der Zielposition. Die geplante Trajektorie ist rot eingezeichnet. In Abbildung 5.8.5b wurde ein Hindernis detektiert und eine neue Trajektorie von der aktuellen Roboterposition (blau) zum Ziel berechnet. Abbildung 5.8.5c skizziert die Ausgabe des Bahnplaners nach der Detektion eines weiteren Hindernisses. Das Hindernis wurde in die Karte eingetragen und die Trajektorie erneut berechnet.

Entsprechend den Abbildungen 5.8.5a-c zeigen die Abbildungen 5.8.6a-h Bilder des realen mobilen Roboters während des Abfahrens der Trajektorie. Zum Zeitpunkt der Aufnahme von Abbildung 5.8.6a wird von dem Robotersystem das erste Hindernis erkannt. Die Trajektorie wird erneut berechnet. Die Abbildungen 5.8.5b-e zeigen das Ausweichmanöver des Roboters, bis zu dem Zeitpunkt, zu welchem das zweite Hindernis mit der PMD-Kamera detektiert wurde und die Robotertrajektorie vom Bahnplanungsverfahren korrigiert wird. Der mobile Roboter fährt entsprechend Abbildung 5.8.5c um das zweite Hindernis herum. Basierend auf den Ergebnissen der Kapitel 5.1 bis 5.7 konnte ein Robotersystem aufgebaut werden, welches unter Verwendung einer PMD-Kamera Hindernisse in Echtzeit erkennt und diese mit der

Umgebungskarte abgleicht. Werden Kollisionen des Roboters mit dem detektierten Hindernissen erkannt, so ist es dem Roboter möglich den Hindernissen autonom auszuweichen, sofern eine alternative Trajektorie zum Zielpunkt existiert. Ist dies nicht der Fall, ist es durch schnelles Anhalten möglich Kollisionen zu vermeiden.

Abbildung 5.8.6: Roboter weicht autonom unbekannten Hindernissen aus

5.9 Autonomes Andocken

In den vorherigen Abschnitten wurden Algorithmen entwickelt, die es einem fahrerlosen Transportsystem ermöglichen, selbstständig von einem Startpunkt zu einem beliebigen Zielpunkt innerhalb einer nur teilweise bekannten Umgebung zu navigieren. In der Praxis ist es jedoch auch häufig der Fall, dass fahrerlose Transportsysteme Anhänger von einem Ort zum nächsten bringen, wo diese dann entweder be- oder entladen werden. Da die Be- und Entladevorgänge einige Zeit beanspruchen, ist eine Entkopplung des Anhängers notwendig, damit das fahrerlose Transportsystem in dieser Zeit weitere Aufgaben durchführen kann. Zurzeit wird dies in der Industrie manuell ausgeführt. Eine zufriedenstellende autonome Lösung hierfür gibt es zurzeit nicht. Aus dieser Motivation heraus wird im Folgenden ein auf PMD-Daten basierender Lösungsansatz für das autonome Ankoppeln von Anhängern vorgestellt. Zielsetzung dieses Kapitels ist die Realisierung eines autonomen Andockvorgangs des aufgebauten Versuchsträgers an eine Anhängerdeichsel.

5.9.1 Versuchsaufbau

Für das autonome Andocken des fahrerlosen Transportsystems wurde die Kamera, eine O3D-100 der Firma ifm, am Heck des Fahrzeugs angebracht. Diese besitzt eine laterale Auflösung von 64x50 Pixeln und ist zum Boden geneigt. Dies gewährleistet zum einen die Einhaltung des Eindeutigkeitsbereich von $d_{max} = 7{,}5m$ und zum anderen eine gute Erkennung der Deichsel.

Abbildung 5.9.1: Versuchsaufbau zum fahrerlosen Andocken

Bei der Wahl des Neigungswinkels muss ein Kompromiss gefunden werden zwischen der maximalen Entfernung von Roboter zur Deichsel, bei der die Deichsel erkannt werden soll, und der minimalen Entfernung, bei der die Deichsel noch erkannt werden muss. Soll die maximale Entfernung groß sein, bedeutet dies, dass das Fahrzeug

aufgrund des beschränkten Objektivöffnungswinkel das letzte Stück blind fahren und sich auf die vorherigen Messdaten verlassen muss. Eine frühe Deichselerkennung jedoch gewährleistet, dass dem Roboter genug Zeit bleibt den Andockvorgang durchzuführen. Um die Kameradaten mit dem Roboter zu verarbeiten, wird die Transformationsmatrix zwischen PMD-Kamera und Roboter, mit dem in Abschnitt 5.3 beschriebenen Kalibrierverfahren, ermittelt. In Abbildung 5.9.1 ist das Deichselmodell, welches zum Test des Andockvorgangs verwendet wurde, dargestellt. Die Deichsel ist schwarz lackiert und somit kommt es aufgrund des PMD-Funktionsprinzips zu einer starken Beeinträchtigung der entsprechenden Distanzwerte. Um diese zu vermeiden und damit die Anhängerdeichsel in dem PMD-Bild leichter erkennbar ist, wurden an diese drei weiße Markierungen angebracht. Prinzipiell ist das Aussehen der Deichsel für die implementierten Algorithmen von zweitrangiger Bedeutung und kann auf diverse Deichseltypen angepasst werden.

5.9.2 Berechnung der Deichselpose

Für die Planung des Andockvorgangs muss die Position und Orientierung der Deichsel zum Roboter bestimmt werden. Der zu diesem Zweck entwickelte Algorithmus ist in Abbildung 5.9.2 skizziert. Der Algorithmus zur Detektion der Deichselposition besteht aus den Schritten: Erkennung der Deichsel mit den Haar-Like-Feature [Viola01], Segmentierung der relevanten Pixel und dem Berechnen der Verschiebung mittels ICP-Algorithmus zwischen aktuellem PMD-Bild und der Referenzaufnahme.

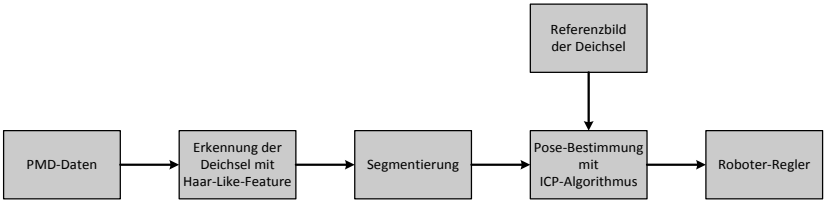


Abbildung 5.9.2: Flussdiagramm zum Andocken an Deichsel

Analog zur Erkennung künstlicher Landmarken, wird die Erkennung der Deichsel durch die Verwendung der Methode der Haar-Like-Feature ausgeführt. Zum Einlernen des Algorithmus werden mit der PMD-Kamera Grauwertbilder aufgenommen, siehe Abbildung 5.9.3, aus denen nach Kapitel 5.4.4 ein Klassifizier berechnet wird.

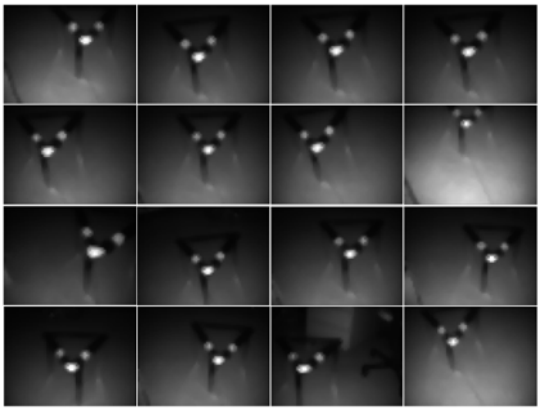


Abbildung 5.9.3: Beispielbilder zum Einlernen der Deichsel

Zur Gewährleistung einer robusten Erkennung der Deichsel wurden mit der am Roboter befestigten PMD-Kamera möglichst viele verschiedene Bilder der Deichsel, mit verschiedenen Hintergründen, aus verschiedenen Positionen aufgenommen.

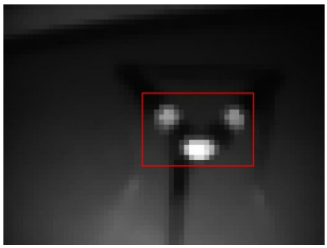


Abbildung 5.9.4: Deichselerkennung mit Haar-Like-Featuren

Abbildung 5.9.4 zeigt ein Beispiel für eine mit Haar-Like-Featuren erkannte Deichsel. Die Bestimmung der Position und Orientierung allerdings ist unter ausschließlicher Verwendung der Haar-Like-Feature nicht möglich. Tests haben gezeigt, dass die Position der Deichsel zwar durch Bestimmung des Schwerpunkts der vordersten weißen Deichselmarkierung hinreichend genau berechnet werden kann, allerdings kann die Orientierung der Deichsel bezogen auf das Roboterkoordinatensystem nur unzureichend bestimmt werden.

Aus diesem Grund muss das Verfahren, wie im Flussdiagramm in Abbildung 5.9.2 dargestellt ist, erweitert werden. Durch die Aufnahme eines Referenzbildes dessen

Position bezüglich der Anhängerkupplung des fahrerlosen Transportsystems manuell vermessen wird, kann die Pose der Deichsel durch Verwendung des ICP-Algorithmus [Besl92] bestimmt werden.

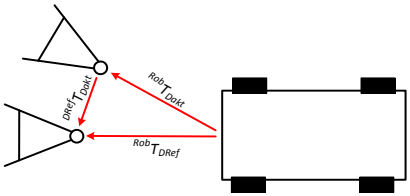


Abbildung 5.9.5: Skizze zur Berechnung der Verschiebung zwischen Roboter und Deichsel

Die aktuelle Transformationsmatrix ${}^{Rob}T_{D_{akt}}$ zwischen Roboter und Deichsel kann somit unter Einbeziehung von Abbildung 5.9.5 berechnet werden.

$$ {}^{Rob}T_{D_{akt}} = {}^{Rob}T_{D_{Ref}} \cdot {}^{D_{Ref}}T_{D_{akt}} \tag{5.109} $$

${}^{Rob}T_{D_{Ref}}$ entspricht der manuell vermessenen Position und Orientierung des Roboters bezüglich der Deichselpose. Die Transformationsmatrix ${}^{D_{Ref}}T_{D_{akt}}$ entspricht der Verschiebung zwischen der aktuellen Deichselposition und der Position der Deichsel zum Zeitpunkt der Referenzbildaufnahme. Diese ist gleich der Verschiebung der Deichsel innerhalb der PMD-Bilder.

Zur Berechnung der Transformationsmatrix müssen in den Aufnahmen die Pixel, welche die Deichsel repräsentieren, extrahiert werden. Aufgrund der geringen Kameraauflösung von nur 64x50 Pixeln, können Pixel, welche irrtümlich für die Berechnung mit dem ICP-Algorithmus berücksichtigt werden, zu einer starken Verfälschung der berechneten Transformationsmatrix führen.

Durch die Erkennung der Deichsel mit der Methode der Haar-Like-Feature kann ein Bildausschnitt, indem sich die Deichsel befindet, vorsegmentiert werden. Die entsprechenden Pixel werden in das Roboterkoordinatensystem überführt. Dies ermöglicht das Filtern aller nicht relevanten Pixel. Alle Pixel in dem vorsegmentierten Bildbereich für die gilt:

$$ 0{,}2m \leq z \leq 0{,}3m $$

zeigen mit hoher Wahrscheinlichkeit die Dechsel. Nur diese werden zur Berechnung der Verschiebung mittels ICP-Algorithmus verwendet.

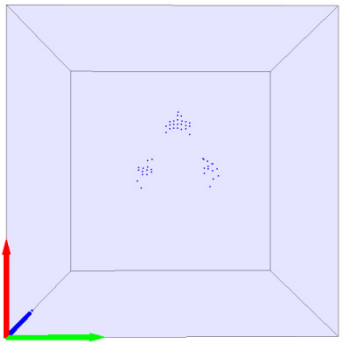


Abbildung 5.9.6: extrahierte 3D-Punkte der Deichsel

Das Ergebnis der Segmentierung ist in Abbildung 5.9.6 für das aufgenommene Referenzbild dargestellt. Diese Punkte, welche lediglich den Markierungen auf der Deichsel entsprechen, werden in die x,y-Ebene projiziert und dem vereinfachten ICP-Algorithmus zur Verfügung gestellt.

Nach der Erkennung der Deichsel und der anschließenden Segmentierung der relevanten Pixel, wird der ICP-Algorithmus zur Berechnung der Verschiebung zwischen dem aufgenommenen Referenzbild, mit bekannter Verschiebung zur Anhängerkupplung, und dem aktuellen Bild verwendet. Der verwendete ICP-Algorithmus basiert auf den Ergebnissen von Abschnitt 5.4.3, wurde jedoch auf die speziellen Anforderungen angepasst. Für die Berechnung der Deichselverschiebung und dem hieraus geplanten Andockmanöver ist prinzipbedingt eine Translation in Richtung der z-Achse des Roboterkoordinatensystems unerheblich, ebenso wie die Rotation um die x- und y-Achse. Aus diesem Grund wurde der eigentliche 3D-ICP-Algorithmus auf die Verwendung von 2D-Daten angepasst. Dies führte zu einer robusteren Berechnung der Transformationsmatrix und zur Minimierung der Berechnungsdauer.

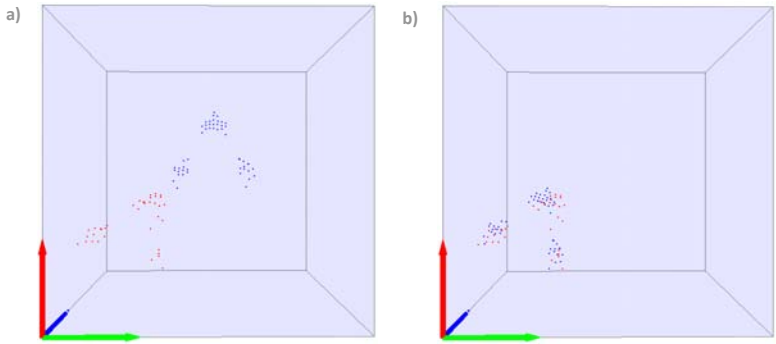


Abbildung 5.9.7: 3D-PMD-Daten a) vor der Anwendung des ICP b) nach der Anwendung des ICP-Algorithmus

Zur Berechnung der Transformationsmatrix zwischen beiden Punktwolken, werden diese zunächst auf die x,y-Ebene projiziert, bevor sie mit dem veränderten ICP-Algorithmus abgeglichen werden. Das Ergebnis dieses Matching-Vorgangs ist in Abbildung 5.9.7 festgehalten. Abbildung 5.9.7a zeigt den Ausgangszustand. Blau eingetragen sind die Punkte des Referenzbilds, rot die des aktuellen Bilds. Abbildung 5.9.7b zeigt das Ergebnis des ICP-Algorithmus. Die berechnete Transformationsmatrix wird der Andockplanung und -Regelung zur Verfügung gestellt.

Aufgrund der geringen Anzahl von Pixeln, welche dem ICP-Algorithmus, resultierend aus der implementierten Vorverarbeitung und der geringen Auflösung der Kamera, zur Verfügung gestellt werden, benötigt die Auswertung der 3D-PMD-Bilddaten eine maximale Zykluszeit t_{max} von:

$$t_{max} = 23ms$$

5.9.3 Andockplanung und –Regelung

Der Vorgang des autonomen Andockens kann zunächst in zwei unabhängige Aufgaben aufgeteilt werden. Zunächst gilt es, ebenso wie bei der autonomen Navigation in einer teilweise bekannten Umgebung, eine Trajektorie zum Zielpunkt zu ermitteln. Bei der Andockplanung wird davon ausgegangen, dass sich keine Hindernisse zwischen Deichsel und fahrerlosem Transportsystem befinden. Die Berücksichtigung einer Umgebungskarte ist somit nicht erforderlich.

Zur Initialisierung des Andockvorgangs müssen zunächst zwei Koordinatensysteme definiert werden. Diese sind das Roboterkoordinatensystem $\{Rob\}$ und das

Koordinatensystem der Deichsel $\{D\}$, wie dies in Abbildung 5.9.8 dargestellt ist. Während sich das Koordinatensystem des Roboters im Mittelpunkt der hinteren Achse befindet, befindet sich das Deichselkoordinatensystem im Mittelpunkt der Deichselöse. Dessen x-Vektor ist entgegen der Richtung des optimalen Andockens definiert.

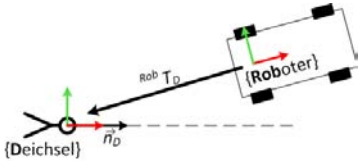


Abbildung 5.9.8: Definition der Koordinatensysteme

Die Andocktrajektorie entspricht einer Geraden, definiert durch den Richtungsvektor $\vec{n}_D$ in Richtung des x-Vektors des Deichselkoordinatensystems $\{D\}$, sowie den Ursprungskoordinaten des Selbigen als Stützpunkt. Zu Beginn des Andockvorgangs ist die ungefähre Position und Orientierung der Deichsel im Weltkoordinatensystem $\{WKS\}$ bekannt. Die anfängliche Andocktrajektorie kann somit berechnet werden. Sollte die Deichsel mit Hilfe der PMD-Kamera erkannt werden, wird die Position der Deichsel dementsprechend korrigiert und die Trajektorie erneut berechnet.

Der implementierte Regler zur Steuerung des Andockvorgangs ist zweigeteilt, erstens in die Regelung des Lenkwinkels in Abhängigkeit des Orientierungs- und Positionsfehlers des Roboters bezüglich der Trajektorie, zweitens in die Regelung der Geschwindigkeit in Abhängigkeit des Abstands zwischen Roboter und Deichsel.

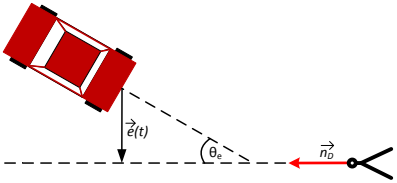


Abbildung 5.9.9: Skizze zur Andockregelung

Als Eingangsgrößen der Positions- und Orientierungsregelung dienen die Abweichung $\vec{e}(t)$ des Roboters von der vorgegebenen Trajektorie und der Orientierungsfehler $\theta_e(t)$.

Zur Berechnung dieser Parameter werden zunächst die Position des fahrerlosen Transportsystems und dessen Orientierung bezogen auf das Deichselkoordinatensystem berechnet.

Die hierfür benötigte Transformationsmatrix ${}^{Rob}T_D$ berechnet sich bei bekannter Position des Roboters und der Deichsel im Roboterkoordinatensystem zu:

$$ {}^{D}T_{Rob} = {}^{WKS}T_{D}^{-1} \cdot {}^{WKS}T_{Rob} \tag{5.110} $$

Aus der berechneten Transformationsmatrix können Positions- und Orientierungsfehler des Roboters zur Andockgeraden bestimmt werden. Aus Abbildung 5.9.9 ist erkenntlich, dass der Positionsfehler $\vec{e}(t)$ der Position in y-Richtung des Roboters betrachtet im Deichselkoordinatensystem entspricht.

$$ \vec{e}(t) = {}^{D}T_{Rob} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{5.111} $$

Der Orientierungsfehler entspricht dem Winkel zwischen beiden Koordinatensystemen, betrachtet in der x/y-Ebene:

$$ \theta_e = f \cdot \arccos\left(\frac{{}^{D}\vec{x}_{Rob} \cdot \vec{n}_D}{\left|{}^{D}\vec{x}_{Rob}\right| \cdot |\vec{n}_D|} \right) \tag{5.112} $$

mit:

$$ f = \begin{cases} +1, & {}^{D}x_{Rob,2} > 0 \\ -1, & {}^{D}x_{Rob,2} < 0 \end{cases} \tag{5.113} $$

Mit dem Positions- und Orientierungsfehler des mobilen Roboters bezüglich der Andockgeraden, kann die in Kapitel 5.6 hergeleitete Reglergleichung nach [Thrun06] verwendet werden:

$$ \varphi_m(t) = \theta_e(t) + \arctan\left(\frac{k \cdot e(t)}{v_x(t)} \right) \tag{5.114} $$

Hierbei entspricht k dem Verstärkungsfaktor und $v_x(t)$ der Geschwindigkeit zum Zeitpunkt t in Richtung des Lenkwinkels und berechnet sich zu:

$$ v_x(t) = v(t) \cdot \cos \varphi_m(t) \tag{5.115} $$

Gl.(5.114) bezieht sich auf das Vorwärtsfahren eines autonomen Fahrzeugs und muss für das Rückwärtsandocken verifiziert werden. Bei positivem Orientierungsfehler muss der Roboter mit einem Gegensteuern mit nach linksgerichtetem Lenkwinkel, welches in der Robotersteuerung ebenfalls einem negativen Lenkwinkel entspricht, reagieren. Folglich muss beim Rückwärtsfahren, um den Orientierungsfehler auszugleichen, ein negativer Lenkwinkel eingestellt werden. Ein positiver Positionsfehler kann wie beim Vorwärtsfahren durch einen positiven Lenkwinkel korrigiert werden. Durch die Berücksichtigung der Geschwindigkeit $v_x(t)$ resultiert ein negativer zweiter Teil der Gl.(5.114), welches zu einem falsch gerichteten Lenkwinkel führt. Aufgrund dessen muss Gl.(5.114) wie folgt angepasst werden:

$$\varphi_m(t) = -\theta_e(t) + \arctan\left(\frac{k \cdot e(t)}{|v_x(t)|}\right) \tag{5.116}$$

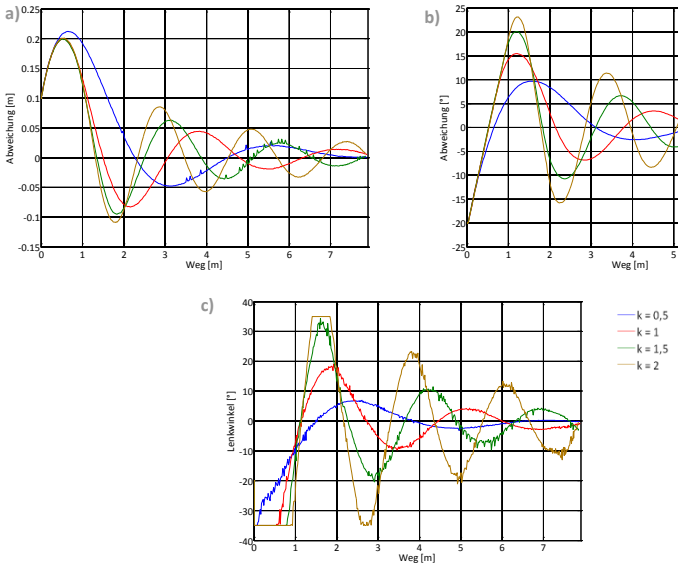


Abbildung 5.9.10:Auswertung des Reglerparameters
a) Positionsabweichung b) Orientierungsabweichung c) Lenkwinkel

Der Verstärkungsfaktor k dient der Gewichtung des Einflusses von Positions- und Orientierungsfehler auf den einzustellenden Lenkwinkel φ_m. Mit zunehmendem Verstärkungsfaktor erhöht sich der Einfluss des Positionsfehlers im Gegensatz zum

Orientierungsfehler. Die Folge ist, dass der Positionsfehler schneller ausgeglichen wird, allerdings führt dies gleichzeitig dazu, dass der Roboter über die gewünschte Position hinausfährt und es somit zum Überschwingen des Positionsfehlers kommt.

Zur Verdeutlichung dieses Sachverhalts sind in den Graphen von Abbildung 5.9.10 die Abweichungen in Position und Orientierung sowie die hierzugehörigen Lenkwinkel für verschiedene Verstärkungsfaktoren k eingetragen. Ausgangsposition für die Messungen war eine Positionsabweichung von $e(0) = 0{,}1m$ und eine Orientierungsabweichung von $\theta_e(0) = -20°$ bezogen auf eine Gerade entlang der x-Achse des Deichselkoordinatensystems, welche die geplante Robotertrajektorie darstellt. Zu erkennen ist, dass für einen höheren Verstärkungsfaktor k der Positionsfehler stärkere Auswirkungen auf den Soll-Lenkwinkel hat. Dies führt zu einem schnelleren Ausgleich des Positionsfehlers bei gleichzeitig stärkerer Zunahme des Orientierungsfehlers. Die Zunahme des Orientierungsfehlers führt, aufgrund des geringen Einflusses auf den Soll-Lenkwinkel, zu einem starken Überschwingen des Positionsfehlers. Verstärkt wird das Überschwingen zusätzlich durch die Begrenzung des Lenkwinkels φ. Für diesen gilt bei Berücksichtigung der Eigenschaften des verwendeten autonomen Fahrzeugs:

$$-35 \leq \varphi \leq 35$$

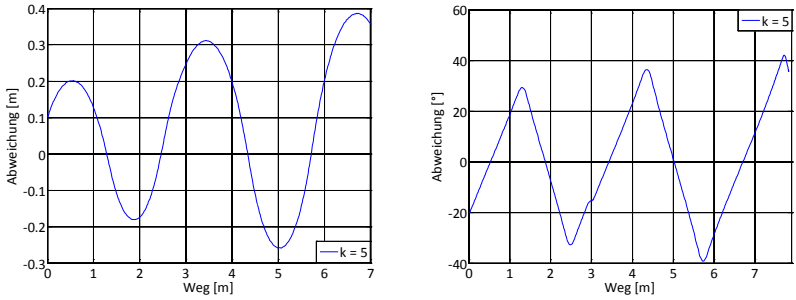


Abbildung 5.9.11: Positions- und Orientierungsfehler bei einer Regelung mit k = 5

Die Auswirkungen eines zu hohen Verstärkungsfaktors sind in den Graphen auf Abbildung 5.9.11 für den Positions- und Orientierungsfehler festgehalten. Schon bei einem Verstärkungsfaktor von $k = 5$ ist der Orientierungsfehler beim Nulldurchgang des Positionsfehlers größer als der initiale Orientierungsfehler. Es kommt zu einem Aufschwingen des Positionsfehlers, da der Einfluss der Orientierungsabweichung auf

den Soll-Lenkwinkel des Roboters gegenüber dem Einfluss des Positionsfehlers vernachlässigbar gering ist.

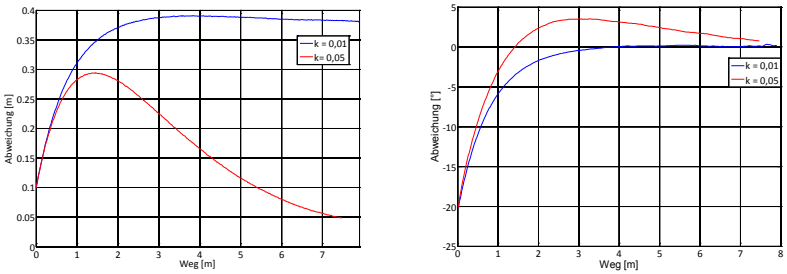


Abbildung 5.9.12: Positions- und Orientierungsfehler bei geringem Verstärkungsfaktor

Ist der Verstärkungsfaktor zu gering, wird die Positionsabweichung nur sehr langsam oder gar nicht korrigiert, wie dies in Abbildung 5.9.12 für die Verstärkungsfaktoren $k = 0{,}01$ und $k = 0{,}05$ zu sehen ist. Für diese Verstärkungsfaktoren ist der Anteil des aus dem Positionsfehler resultierenden Lenkwinkels gegenüber dem Orientierungsfehler so gering, dass lediglich der Orientierungsfehler aber nicht der Positionsfehler korrigiert wird.

Für die Anwendung des implementierten Reglers beim autonomen Andocken des fahrerlosen Transportsystems kann folglich davon ausgegangen werden, dass für eine optimale Andockregelung der Verstärkungsfaktor in den Grenzen:

$$0{,}5 \leq k \leq 1$$

liegen sollte.

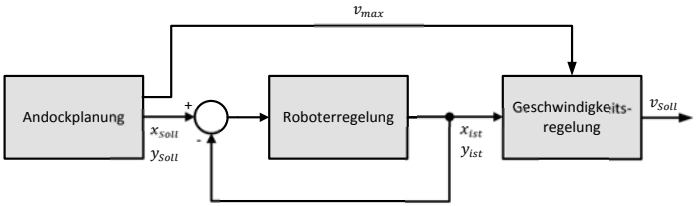


Abbildung 5.9.13: Roboterregelung inklusive Geschwindigkeitsregelung

Der verwendete mobile Roboter besitzt keine aktive Bremse. Deswegen muss eine Geschwindigkeitsregelung der eigentlichen Roboterregelung untergeordnet werden, welche in Abhängigkeit der Entfernung zwischen Roboter und Anhängerdeichsel eine schrittweise Reduzierung der Robotergeschwindigkeit vornimmt. Die Sollgeschwindigkeit v_{Soll} wird berechnet zu:

$$v_{Soll} = \begin{cases} v_{max}, & x_{ist} \geq 1{,}0m, \\ v_{max} \cdot (x_{ist} - 0{,}1), & 0{,}1m < x_{ist} < 1{,}0m \\ 0, & x_{ist} \leq 0{,}1m \end{cases} \quad (5.117)$$

5.9.4 Auswertung

Durch die Kombination der Deichselerkennung mit der Andockplanung und -Regelung konnte ein System aufgebaut werden, welches das autonome Andocken eines mobilen Roboters an eine Anhängerdeichsel mittels PMD-Sensorik ermöglicht.

Abbildung 5.9.14: Andocken an ein Deichselmodell

Exemplarisch gibt Abbildung 5.9.14 einen Andockvorgang des Versuchsträgers an das Anhängerdeichselmodell wieder. Zur Initialisierung der Andockplanung wird eine vorgegebene Anhängerposition angenommen. Wurde die Andockplanung durchgeführt, wird der Roboter mit dem in Kapitel 5.9.3 hergeleiteten Regler in Richtung der Deichsel gesteuert. Sobald die Deichsel mit der PMD-Kamera erkannt und deren Position bezüglich des Roboters mit dem ICP-Algorithmus unter Einbeziehung des anfangs aufgenommenen Referenzbild berechnet ist, findet eine neue Planung statt

und der Roboter wird, mit dem entwickelten nichtlinearen „feedback“-Regler, auf die neue Bahn geregelt.

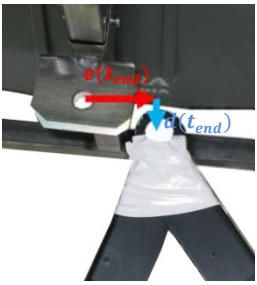


Abbildung 5.9.15: Fehler beim autonomen Andocken

Der bei dem Andockmanöver resultierende Fehler kann nach Abbildung 5.9.15 in zwei Anteile unterteilt werden. Zum einen ist dies der Fehler $d(t_{end})$ in $\vec{x}$-Richtung des Roboterkoordinatensystems, zum anderen der Fehler $e(t_{end})$ in $\vec{y}$-Richtung. Beide Fehler unterliegen unterschiedlichen Fehlerquellen. Diese sind zum Beispiel das falsche Messen der Deichselposition bei der Aufnahme des Referenzbilds, Kameramessfehler aufgrund verrauschter Messwerte, fehlerhafte Berechnung der Transformationsmatrix mit dem ICP-Algorithmus durch stark abweichende Punktwolken, sowie eine resultierende Reglerabweichung des verwendeten Reglersystems. Eventuelle Fehler, entstehend aus der Roboter-Kamera-Kalibrierung, haben keinen Einfluss auf den Positionsfehler des Roboters beim Andocken. Da durch die Verwendung eines Referenzbilds eine absolute Messung der Deichselposition mit der Kamera entfällt, spielt ein Kalibrierfehler keine Rolle. Jedoch muss die Kalibrierung für eine bessere Segmentierung der Deichsel im 3D-Kamerabild durchgeführt werden.

Problematisch ist ebenfalls die Implementierung des Geschwindigkeitsreglers. Aufgrund der nicht vorhandenen Bremse des Robotersystems wird die Robotergeschwindigkeit abhängig von der Entfernung zwischen Roboter und Deichsel langsam herabgesetzt. Die letzten Zentimeter rollt das Fahrzeug aus und somit ist die Größe des Fehlers $d(t_{end})$ abhängig von der Höhe der Startgeschwindigkeit des Robotersystems und der gefahrenen Trajektorie und kann nur schwer korrigiert beziehungsweise vorausgesagt werden. Abhilfe könnte hier eine aktive Bremse schaffen mit der der Roboter ausgestattet werden könnte. Alle aufgeführten Mess- und Berechnungsfehler könnten durch den Einsatz einer PMD-Kamera mit höherer Auflösung oder einem verminderten Messrauschen verringert werden.

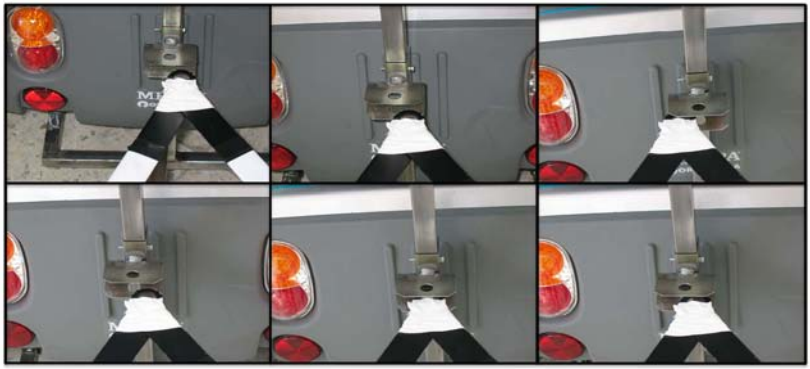


Abbildung 5.9.16: Fehler beim Andocken

Eine qualitative Aussage über die Genauigkeit des autonomen Andockens kann nur sehr schwer getroffen werden, da diese von verschiedenen Faktoren abhängig ist und reproduzierbare Messungen nicht gewährleistet werden können. Wird zum Beispiel die Deichsel erst spät von der Kamera erkannt, ist das Fahrzeug schon sehr nah an der Deichsel. Dem Fahrzeug bleibt nur wenig Zeit das Andockmanöver auszuführen. Üblicherweise führt dies zu einer erhöhten Regeldifferenz zwischen Deichsel- und Roboterendposition.

5.10 Zusammenfassung und Auswertung

In diesem Kapitel wurde der Aufbau eines fahrerlosen Transportsystems, basierend auf einem Elektro-Scooter der Firma Meyra, beschrieben. Die Roboterplattform wurde mit zwei PMD-Kameras ausgerüstet. Eine der Kameras mit 200x200 Pixeln wurde an der Front des Fahrzeugs auf einer kippbaren Halterung, die andere wurde am Heck angebracht. Zur Berechnung des Kameraneigungswinkels und somit zur Berechnung einer Transformationsmatrix zum Überführen der Kameradaten in das Roboter-koordinatensystem, wurde in Kapitel 5.3 ein entsprechendes Kalibrierverfahren vorgestellt. Die vordere Kamera dient der Selbstlokalisierung und der Überwachung des Fahrwegs. Zur Selbstlokalisierung wurden zwei Methoden entwickelt. Erstere ist die Bestimmung der relativen Position und Orientierung bezüglich der Ausgangsposition unter Berechnung der Verschiebung zwischen den PMD-Kamerabildern mittels ICP-Algorithmus. Diese relative Poseschätzung wurde anschließend mit den Daten der am Roboter verbauten Radencoder, durch Schätzung des Fehlervektors mit einem Kalman-

Filter, fusioniert. Die zweite implementierte Methode zur Selbstlokalisierung ist die absolute Bestimmung der Roboterpose anhand von künstlichen Landmarken, deren ermittelte Pose mit der bekannten Pose innerhalb der Umgebungskarte abgeglichen wird. Hierdurch kann eine Korrekturmatrix, mit welcher das Ergebnis der relativen Lokalisierung kontinuierlich korrigiert wird, berechnet werden.

Der Entwurf eines geeigneten Bahnplaners wurde in Kapitel 5.5 behandelt. Hierzu wurde eine geeignete Umgebungsdarstellung entworfen. Die Umgebungsinformationen, gegeben durch eine CCD-Karte, wurden geladen und in Quadtrees aufgeteilt. Dies ermöglicht eine effektive Kollisionsabfrage zwischen Hindernissen und Robotersystem. Zusätzlich wurde ein Suchgraph entwickelt, welcher eine Kombination von Voronoidiagramm und Sichtbarkeitsgraph darstellt. Das implementierte Bahnplanungsverfahren ist ein zweistufiger Algorithmus. Zunächst findet eine Vorplanung ohne Berücksichtigung der Roboterkinematik statt. Diese wird mittels A*-Algorithmus anhand des Suchgraphen durchgeführt. Wird hierbei eine mögliche Verbindung zwischen Start- und Zielposition über die in der Karte eingetragenen Stützpunkte ermittelt, wird ein zweiter Bahnplanungsschritt durchgeführt. Ebenfalls mit dem A*-Algorithmus wird eine kollisionsfreie Bahn zwischen den in der Vorplanung berechneten Stützpunkten ermittelt. Die Kombination einer Vorplanung zur Berechnung von Trajektorienstützpunkten mit anschließender detaillierter Bahnplanung unter Berücksichtigung der Roboterkinematik ermöglicht somit eine sichere und effiziente Methode zur Trajektorienplanung.

Anschließend wurden in Kapitel 5.7 Methoden zur Hinderniserkennung mit einer PMD-Kamera beschrieben. Hierbei wird zunächst, basierend auf der durch das Kalibrierverfahren bestimmten Registrierung der Kamera innerhalb des Roboterkoordinatensystems, eine Vorverarbeitung der Kameradaten durchgeführt. Diese dient der Filterung nicht relevanter Pixel, welche zum Beispiel den Fahrweg repräsentieren. Mittels des RANSAC-Verfahrens wird versucht Ebenen in die verbliebenen Punkte zu legen. Diese Ebenen werden durch die verwendeten Punkte beschränkt und liefern somit Informationen zur Ausrichtung und Ausdehnung möglicher Hindernisse. Die Trajektorie wird anschließend mit den erkannten Hindernissen auf Kollision überprüft.

Kapitel 5.8 beschäftigte sich mit der Erweiterung des Bahnplanungsverfahrens zu einem reaktiven Planungsverfahren, indem die Hinderniserkennung mit der Umgebungskarte fusioniert wurde. Hierfür wird eine gesonderte Hinderniskarte verwendet, in welcher die erkannten Hindernisse eingetragen werden. Bevor dies geschieht, werden sie mit den bestehenden Hindernissen abgeglichen. Es gibt drei

verschiedene Möglichkeiten. Sollte das Hindernis vorhanden sein, wird es verworfen. Ist es nur teilweise in der Karte vorhanden, wird das Hindernis dementsprechend erweitert. Ist es nicht vorhanden, wird es in der Hinderniskarte gespeichert. Die Verwendung einer zweiten Karte für die mit der Kamera detektierten Hindernisse hat den Vorteil, dass diese Karte, sobald die Roboterplattform am Ziel angelangt ist, auf einfache Weise gelöscht werden kann. Sind die neuen Hindernisse in die Karte eingetragen und wurde eine Kollision des Roboters auf der geplanten Trajektorie detektiert, kann die Robotertrajektorie nach beschriebenem Schema erneut berechnet werden.

Das Fahren des Roboters auf einer vorgegeben Bahn benötigt eine Lenkwinkelreglelung um mögliche Positionsabweichungen zu vermindern. Dieser wurde in Kapitel 5.6 implementiert und verifiziert.

Durch die Entwicklung von speziell auf die Eigenschaften der PMD-Kamera angepassten Methoden konnte die PMD-Technik als neuartiges Sensorkonzept zur echtzeitfähigen Selbstlokalisierung und Hinderniserkennung in der mobilen Robotik etabliert werden. Die Verwendung der PMD-Kamera ermöglicht zum einen die Verbesserung der Genauigkeit zur Selbstlokalisierung des mobilen Roboters gegenüber der alleinigen Verwendung von Radencodern. Zum anderen dient die PMD-Kamera zur Detektion von Hindernissen in einer nur teilweise bekannten Umgebung. In Kombination mit dem entwickelten reaktiven Bahnplaner können auf diese Weise Kollisionen rechtzeitig erkannt und alternative Trajektorien zum Ziel gefunden werden.

Eine weitere zu lösende Aufgabe bei Verwendung von fahrerlosen Transportsystemen ist das autonome Andocken rückwärts an einen Anhänger. Es wurde unter Verwendung der PMD-Kameratechnik ein neues Verfahren entwickelt, welches eine schnelle 3D-Erkennung der Anhängerdeichsel und somit die genaue Lokalisierung der Deichselposition und -Orientierung ermöglicht. Das Anbringen einer PMD-Kamera mit einer lateralen Auflösung von 64x50 Pixeln ermöglicht die Überwachung des Raums hinter dem Fahrzeug. Mittels Haar-Like-Feature wurde eine robuste Methode zur Erkennung der Anhängerdeichsel implementiert. Die Positionsbestimmung der Deichsel bezüglich des Roboters erfolgt anschließend durch Abgleich eines Referenzbilds, mit bekannter Position und Orientierung der Deichsel bezüglich des Roboters, mit dem aktuellen PMD-Bild unter Zuhilfenahme des ICP-Algorithmus. Das Annähern des Roboters an die Deichsel erfolgt mit dem in Kapitel 5.6 verwendeten Regler, welcher auf die entsprechenden Verhältnisse angepasst wurde.

Zusammenfassend kann festgestellt werden, dass die PMD-Kamera erfolgreich in Applikationen zur sicheren Robotersteuerung im Bereich der autonomen mobilen Robotik integriert werden konnte. Die PMD-Kamera ermöglicht ein schnelles Erkennen von künstlichen Landmarken, sowie eine präzise Bestimmung von deren Lage bezüglich des Roboters. Fusioniert mit der Berechnung der Verschiebung aufeinanderfolgender PMD-Bilder, konnte ein zuverlässiges Verfahren zur Selbstlokalisierung entwickelt werden. Des Weiteren konnte durch den Gebrauch der PMD-Kamera eine schnelle dreidimensionale Erkennung von Hindernissen implementiert werden, welches die Basis zur PMD basierten, reaktiven Bahnplanung bildet und erheblich zur sicheren Steuerung des mobilen Roboters in einer nur teilweise bekannten Umgebung beiträgt.

Verglichen mit der Verwendung von Laserscannern zur Hindernisdetektion bietet die PMD-Technik die Möglichkeit einer dreidimensionalen Vermessung der erkannten Hindernisse. Höhe und Breite der Hindernisse können somit in eine Karte zur Bahnplanung eingetragen werden. Vorteilhaft ist zudem die Möglichkeit zur Erkennung negativer Hindernisse. Bei Verwendung eines Laserscanners zur Hinderniserkennung müsste ein zweiter eingesetzt werden, welcher zum Fahrweg geneigt ist. Gegenüber Stereovisionsystemen bietet die PMD-Kamera den Vorteil, dass selbst große Objekte mit geringer Struktur erkannt werden. Während das Stereovisionsystem hierbei nur Distanzwerte der äußeren Bereiche des Hindernisses liefert, liefert die PMD-Kamera eine gleichmäßig über das Hindernis verteilte Punktwolke, welches eine robustere Erkennung und einfacher zu implementierende Erkennungsalgorithmen erlaubt. Als nachteilig beim Gebrauch der PMD-Kamera stellte sich die begrenzte Reichweite heraus. Objekte konnten aufgrund der geringen Ausleuchtung durch das aktiv modulierte Licht nur bis zu einer Distanz kleiner 5m mit zufriedenstellender Genauigkeit aufgenommen werden. Ein Weiteres Defizit ist der geringe Öffnungswinkel der verwendeten PMD-Modelle im Vergleich zu Laserscannern. Jedoch stellte sich die geringe laterale Auflösung der Kameras als nicht entscheidendes Kriterium heraus. Mit der LynCube, welche eine laterale Auflösung von 200x200 Pixeln besitzt, ließ sich zwar die Pose künstlicher Landmarken präziser bestimmen, zur Erkennung dieser sowie zur Detektion von Hindernissen genügte jedoch schon eine Kamera mit geringerer lateraler Auflösung. Zusätzlich bietet diese den Vorteil eines geringeren zu verarbeitenden Datenvolumens und somit zu einer schnelleren Verarbeitung der 3D-Daten.

6 Die PMD-Kamera als neuartiges Sensorkonzept zur sicheren Steuerung von Handhabungsrobotern

In Abschnitt 5 wurde die Verwendung der PMD-Kamera zur Realisierung des sicheren autonomen Fahrens mobiler Roboter beschrieben und Vor- sowie Nachteile gegenüber konventioneller Sensorik aufgeführt.

Die Verwendung der PMD-Kamera bietet in der Industrierobotik ebenso wie in der mobilen Robotik ein breites Spektrum von Anwendungsmöglichkeiten. Industrielle Sicherheitsstandards erfordern die Absicherung von Roboterarbeitsräumen gegen das unbeabsichtigte Eindringen von Personen bei in Betrieb befindlichen Robotern. Um dies zu gewährleisten, werden üblicherweise Sicherheitszäune eingesetzt. Die Installation von Sicherheitszäunen ist verhältnismäßig teuer und verhindert eine Kooperation zwischen Mensch und Roboter, welches jedoch bei einer Vielzahl von möglichen Anwendungen wünschenswert wäre. So wurde in den letzten Jahren dazu übergegangen 2D-Laserscannersysteme einzusetzen, welche den Roboterarbeitsbereich überwachen. Dies ermöglicht eine Abschaltung des Roboters, falls eine Person die Roboterzelle betritt [Ebert02]. Eine vollständige Überwachung des Roboters kann jedoch hiermit nicht gewährleistet werden. Der Einsatz von 3D-Laserscannern ist für die Überwachung der Roboterzelle aufgrund der geringen Bildrate nicht geeignet.

Zur dreidimensionalen Überwachung des Roboterarbeitsraumes, um Veränderungen jeglicher Art innerhalb der Roboterzelle zu registrieren und so Kollisionen des Roboters mit seiner Umwelt zu vermeiden, gibt es neben der Überwachung der Roboterzelle mit Laserscannern auch Ansätze hierzu Stereovisionsysteme einzusetzen, welche die Aufnahme einer dreidimensionalen Roboterumgebung ermöglichen. Dies hat sich jedoch aufgrund der geringen Messauflösung bisher nicht etablieren können.

Im Folgenden soll ein System aufgebaut werden, welches die Überwachung einer Roboterzelle mit PMD-Kameras ermöglicht. Personen und Fremdobjekte sollen auf einfache Weise aus den 3D-Daten der Kamera extrahiert werden. Wenn mögliche Fremdobjekte erkannt werden, soll der Roboter im ersten Schritt das Abfahren der programmierten Trajektorie abbrechen. Sollte sich ein Mensch in der Roboterzelle aufhalten, muss dies erkannt werden und anhand der berechneten Entfernung zwischen Roboter und Mensch, die Wahl zwischen Reduzierung der Verfahrensgeschwindigkeit und Anhalten des Roboters getroffen werden. Sollten lediglich Veränderungen in der Roboterzelle durch Fremdobjekte detektiert werden, werden diese auf mögliche Kollision mit dem Roboter auf der gegebenen Trajektorie überprüft. Beim Erkennen einer Kollision muss der Roboter stehenbleiben bis das Hindernis entfernt wurde. Sollte die Trajektorie weiterhin kollisionsfrei passierbar sein, führt der Roboter seine Arbeit fort. Im letzten Schritt soll die Funktionsfähigkeit der PMD basierten Roboterüberwachung durch eine Online-Trajektorienplanung erweitert werden, so dass der Roboter die Möglichkeit besitzt, bei erkannter Kollision unter Berücksichtigung der 3D-PMD-Daten eine alternative Trajektorie zu planen.

Im nächsten Abschnitt wird zunächst der Aufbau der Testumgebung, inklusive des verwendeten Robotersystems und der Sensoren, erläutert. Die Verwendung der PMD-Kamerainformationen für die Kollisionsberechnungen im Roboterkoordinatensystem erfordert eine Kalibrierung zwischen diesem und dem Kamerakoordinatensystem. Die Implementierung der Kalibrierung wird in Unterkapitel 6.3 beschrieben und verifiziert. Ist das System kalibriert, können die Kameradaten für die Überwachung der Roboterzelle verwendet werden. Die speziell auf die Möglichkeiten der PMD-Kamera angepassten Algorithmen zur Überwachung der Roboterzelle werden im Anschluss der Kalibrierung behandelt. In Abschnitt 6.5 erfolgt die Beschreibung des entwickelten Online-Bahnplanungsverfahrens, welches die Trajektorienplanung anhand der PMD-Daten in Kombination mit einer CAD-Karte des Roboterarbeitsraums durchführt. Im letzten Abschnitt dieses Kapitels wird ein Fazit aus der Verwendung der PMD-Kamera im Bereich der Roboterarbeitsraumüberwachung für Handhabungsroboter gezogen.

6.1 Versuchsaufbau

Im Folgenden werden die einzelnen Komponenten des Versuchsträgers zur autonomen Erkennung und Vermeidung von Kollisionen zwischen einem Handhabungsroboter und möglichen Hindernissen basierend auf PMD-Daten vorgestellt. Zunächst wird auf die Eigenschaften der einzelnen Komponenten und das Design der Roboterzelle eingegangen. Anschließend werden die Möglichkeiten zur Kommunikation zwischen den einzelnen Systemkomponenten erläutert.

6.1.1 Verwendete Komponenten

Das Robotersystem

Für die Entwicklung einer PMD basierten Roboterüberwachung wurde ein KUKA KR3 verwendet. Der KR3 ist ein sechsachsiger Handhabungsroboter mit Gelenkkinematik und ist konzipiert für Anwendungen mit geringen Traglasten. Er eignet sich zur Maschinenbestückung und Teilehandhabung, für Labortechnik, Produktprüfung und vielen Bearbeitungs- und Montageaufgaben.

Abbildung 6.1.1: Kuka KR3 mit Schaltschrank

Die Nenn-Traglast des Roboters ist auf 3kg begrenzt und die maximale Reichweite beträgt 635mm.

Der Steuerungsrechner basiert auf zwei parallellaufenden Betriebssystemen. Zum einen ist dies, ein für die Visualisierung zuständiges auf embedded-Windows XP basierendes Betriebssystem und zum anderen das echtzeitfähige VXWorks. Auf diesem erfolgt die Regelung und Steuerung des Roboters.

Die Programmierung des KUKA-Roboters erfolgt mittels KUKA Robot Language (KRL).

Die PMD-Kamera

Für die Überwachung der Roboterzelle wurde die [PMDVision]3kS verwendet, welche im Laufe der Entwicklung von einer [PMDVision] S3 ersetzt wurde. Beide Kameras besitzen eine laterale Auflösung von 64x48 Pixeln. Vorteil der [PMDVision] S3 ist jedoch der größere Öffnungswinkel des Objektivs von 30°(h)x40°(v).

6.1.2 Design der Roboterzelle

In Abbildung 6.1.2 ist der Aufbau der Roboterzelle zu sehen. Die PMD-Kamera, auf der Abbildung eine [PMDVision] S3, ist oberhalb des Roboters angebracht. Die optische Achse der Kamera ist nahezu senkrecht zu dem Roboter ausgerichtet. Der Roboter ist auf einem Tisch montiert, so dass sich ein Abstand von $\mathrm{d} = 2800\mathrm{mm}$ zwischen Kamer- und Roboter ergibt.

Abbildung 6.1.2: Versuchsaufbau der Roboterzelle

Die [PMDVision] S3 besitzt einen Öffnungswinkel von 30°(h)x40°(v). Bei der angegebenen Einbauhöhe von $d = 2800mm$ beträgt die beobachtbare Grundfläche auf Tischhöhe $A \approx 1{,}5m \times 2{,}0m$. Der beobachtbare Bereich beträgt bei Verwendung der [PMDVision] 3kS, aufgrund des geringeren Öffnungswinkels von ungefähr 12°(h)x10°(v), jedoch nur $A \approx 1{,}0m \times 0{,}98m$.

Zur Verarbeitung der Bild- und Roboterdaten wird ein Intel Core 2 Duo mit 2,13GHz und 2GB RAM verwendet.

6.1.3 Netzwerkarchitektur

Der verwendete PC dient als zentrales Element zur PMD basierten Überwachung der Roboterzelle. Hier fließen die Roboter- und Kameradaten zusammen und werden verarbeitet. Abbildung 6.1.3 zeigt die verwendete Netzwerkarchitektur.

Die PMD-Kamera wird modellabhängig entweder mittels Firewire oder über Ethernet mit dem PC verbunden. Zur Kommunikation wird auf ein von dem Kamerahersteller mitgeliefertes Software-Interface zurückgegriffen, welches in das entwickelte Programm integriert wurde und das Auslesen von Entfernungs- und Intensitätsbildern, sowie das Einstellen der Integrationszeit, um die Kamera auf die gegebenen Lichtverhältnisse und Reflexionseigenschaften innerhalb der beobachteten Szene anzupassen, ermöglicht.

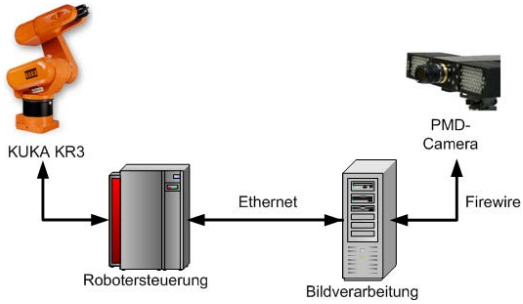


Abbildung 6.1.3: Netzwerkarchitektur der Roboterzelle

Die Robotersteuerung wird ebenfalls über eine Ethernetverbindung an den PC angeschlossen. Die Kommunikation erfolgt über ein XML-Protokoll. Dafür wurde roboterseitig das KUKA KRL-XML-Interface verwendet. Dies ist ein zusätzliches vom Hersteller angebotenes Programm, welches eine Kommunikation über Ethernet mittels XML-Protokoll ermöglicht. PC-seitig wurde ein Interface zur XML basierten Kommunikation unter Verwendung der QT-XML Bibliothek für C++ entworfen.

Das Auslesen von Roboterdaten, speziell von dessen Position, Orientierung und Gelenkstellungen, und das Auslesen der 3D-Kamerabilder erfolgt in zwei parallellaufenden Programmabschnitten, so dass jedem Bild eine definierte Roboterposition und Orientierung zugewiesen werden kann.

6.2 Kalibrierung

Die Verwendung der PMD-Kamera in Verbindung mit dem Handhabungsroboter erfordert, ebenso wie der Gebrauch der Kamera zur Steuerung des mobilen Roboters in Kapitel 5.3, eine Kalibrierung zwischen den unterschiedlichen Koordinatensystemen. Zur Verarbeitung der Kameradaten zur Hinderniserkennung und Bahnplanung müssen diese in dem gleichen Koordinatensystem vorliegen. Als gemeinsames Koordinatensystem wird im Folgenden das Roboterkoordinatensystem *{Rob}* verwendet.

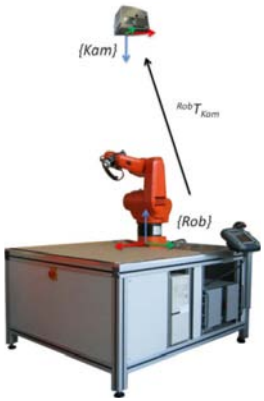


Abbildung 6.2.1: Roboterzelle und ihre Koordinatensysteme

Definiert werden die Koordinatensysteme, wie in Abbildung 6.2.1 skizziert ist.

Es muss eine Methode entwickelt werden um die Ermittlung der Transformationsmatrix ${}^{Rob}T_{Kam}$ zu ermöglichen, so dass gilt:

$$\vec{p}_{Rob_i} = {}^{Rob}T_{Kam} \cdot \vec{p}_{Kam_i} \qquad i = 1 \dots n \tag{6.1}$$

$\vec{p}_{Rob_i}$ entspricht den in das Roboterkoordinatensystem transformierten 3D-Bildpunkten $\vec{p}_{Kam_i}$ des Kamerakoordinatensystems.

Zur Berechnung der Transformationsmatrix ${}^{Rob}T_{Kam}$ zwischen Kamera- und Roboterkoordinatensystem wurde ein speziell auf die Messeigenschaften des PMD-Systems angepasster Algorithmus entwickelt. Zur Bestimmung der Roboterpose wurde ein Kalibrierkörper am Tool-Center-Point (TCP) des Roboters angebracht. Dieser besteht

aus 4 retroreflektierenden Kugeln, deren Position mittels Kamera detektiert werden kann. Hierdurch können Bewegungen des Roboters mit der Kamera gemessen werden.

Die Transformationsmatrix ${}^{Rob}T_{Kam}$ setzt sich aus einem Translations- und einem Rotationsanteil zusammen.

$$ {}^{Rob}T_{Kam} = \begin{bmatrix} & {}^{Rob}R_{Kam} & & {}^{Rob}\vec{t}_{Kam} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2} $$

Die einzelnen Anteile können durch spezielle Kalibrierbewegungen des Roboters ermittelt werden.

6.2.1 Bestimmung der Rotationsmatrix

Zur Bestimmung der Rotation ${}^{Rob}R_{Kam}$ wird eine feste Orientierung des Referenzkörpers zum Roboterkoordinatensystem eingestellt. Die Roboterbewegung gestaltet sich in der Art, dass der Referenzkörper parallel zu jeder Achse des Roboterkoordinatensystems verfahren wird. Für jede Achse wird während der Roboterbewegung kontinuierlich die Position des Referenzkörpers mit der PMD-Kamera gemessen. Das Ergebnis sind drei Punktwolken, die jeweils eine parallele Achse zu dem Roboterkoordinatenachsen in Kamerakoordinaten bilden. Durch Verwendung des RANSAC-Algorithmus in Kombination mit der Methode der linearen Näherung von Geraden in dreidimensionale Punktwolken, werden hieraus die drei Einheitsvektoren $\vec{x}_{Kam}$, $\vec{y}_{Kam}$, $\vec{z}_{Kam}$ auf den Koordinatenachsen berechnet. Aus diesen kann die Rotationsmatrix ${}^{Rob}R_{Kam}$ ermittelt werden.

$$ {}^{Rob}R_{Kam} = (\vec{x}_{Kam}, \vec{y}_{Kam}, \vec{z}_{Kam}) \tag{6.3} $$

6.2.2 Berechnung der Translation

Der Translationsanteil der Transformationsmatrix ${}^{Rob}T_{Kam}$ ist nur indirekt bestimmbar. Hinsichtlich dessen zeigt Abbildung 6.2.2 den verwendeten Kalibrieraufbau.

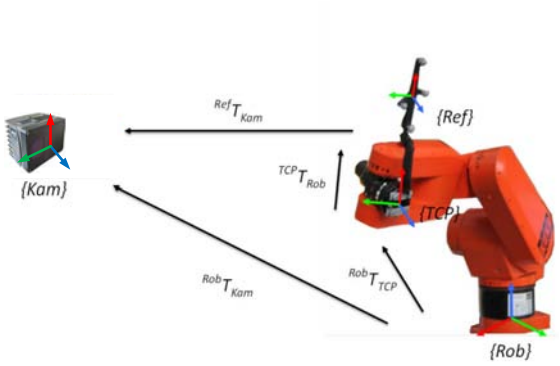


Abbildung 6.2.2: Transformationskette für die Kalibrierung zwischen Roboter- und Kamerakoordinatensystem

Die Translation ${}^{Rob}\vec{t}_{Kam}$ zwischen Roboter- und Kamerakoordinatensystem im Roboterkoordinatensystem ist dem zufolge über die Addition der entsprechenden Anteile bestimmbar. Es gilt:

$$ {}^{Rob}\vec{t}_{Kam} = {}^{Rob}\vec{t}_{TCP} + {}^{TCP}\vec{t}_{Ref} + {}^{Ref}\vec{t}_{Kam} \tag{6.4} $$

Zur fehlerfreien Addition müssen die einzelnen Translationsanteile im selben Koordinatensystem vorliegen. Die Translation ${}^{Rob}\vec{t}_{TCP}$ kann für jede beliebig angefahrene Position über die Vorwärtskinematik nach Denhavit und Hartenberg [Lamikiz08] aus den Gelenkstellungen des Roboters entnommen werden.

Die beiden übrigen Translationsanteile ${}^{TCP}\vec{t}_{Ref}$ und ${}^{Ref}\vec{t}_{Kam}$ können nicht direkt im Roboterkoordinatensystem ermittelt werden, sondern müssen durch Messung mit der PMD-Kamera ermittelt werden. Die Bestimmung von ${}^{TCP}\vec{t}_{Ref}$ erfolgt durch eine Drehbewegung des Referenzobjekts um den TCP. Bei kontinuierlicher Messung der Referenzkörperposition entsteht eine Punktwolke, welche eine Kreisbahn um den TCP beschreibt.

$$ \vec{k} = \vec{m} + \vec{r} \cdot \cos(\alpha) + \vec{s} \cdot \sin(\alpha) \tag{6.5} $$

Durch das Anpassen der Kreisgleichung nach Gl.(6.5), wobei $\vec{m}$ den Mittelpunkt und $\vec{r}$ sowie $\vec{s}$ die Spannvektoren des Kreises in Kamerakoordinaten bilden, kann die Verschiebung ${}^{TCP}\vec{t}_{Ref}$ bestimmt werden. Entsprechend wird eine beliebige Stellung des sechsten Robotergelenks bei unveränderter Position eingestellt. Für diese wird die

Translation ${}^{Rob}\vec{t}_{TCP}$ aus der Robotersteuerung entnommen. Anschließend wird durch Messung der Referenzkörperposition mit der PMD-Kamera die Translation ${}^{Kam}\vec{t}_{Ref}$ ermittelt. Die letzte unbekannte Verschiebung ${}^{Ref}\vec{t}_{TCP}$ kann unter Zuhilfenahme des Kreismittelpunkts berechnet werden.

$$ {}^{Ref}\vec{t}_{TCP} = \vec{m} - {}^{Kam}\vec{t}_{Ref} \tag{6.6} $$

Dies entspricht der Subtraktion der aktuellen Referenzkörperposition im Kamerakoordinatensystem vom Mittelpunkt des durch die Roboterbewegung aufgespannten Kreises. Durch Addition der beiden, im Kamerakoordinatensystem vorliegenden Vektoren, erhält man die Translation für die aktuelle Position zwischen Kamera und Roboter-TCP. Zur Berechnung der vollständigen Translationskette besteht die Möglichkeit, diese in das Roboterkoordinatensystem, durch Multiplikation mit der bereits hergeleiteten Rotationsmatrix ${}^{Rob}R_{Kam}$, zu überführen.

$$ {}^{Rob}\vec{t}_{TCP} = {}^{Rob}R_{Kam} \cdot {}^{Kam}\vec{t}_{TCP} \tag{6.7} $$

Folglich kann der Translationsvektor zwischen Roboter- und Kamerakoordinatensystem berechnet werden.

$$ {}^{Rob}\vec{t}_{Kam} = {}^{Rob}\vec{t}_{TCP} + {}^{TCP}\vec{t}_{Kam} \tag{6.8} $$

Hieraus ergibt sich für die geforderte Transformationsmatrix:

$$ {}^{Rob}T_{Kam} = \begin{bmatrix} & {}^{Rob}R_{Kam} & & {}^{Rob}\vec{t}_{Kam} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.9} $$

6.2.3 Erkennung des Referenzkörpers

Der für die Kalibrierung verwendete Referenzkörper ist auf Abbildung 6.2.3 zu sehen. Wie bereits erwähnt, besteht dieser aus vier Kugeln mit retroreflektierender Oberfläche.

Abbildung 6.2.3: Kalibrierkörper für die Roboter-Kamera Kalibrierung

Die Kugeln besitzen sehr gute Reflexionseigenschaften für Wellenlängen im Spektralbereich des aktiv modulierten Kameralichts. Aufgrund dessen werden die Kugeln durch sehr helle Bereiche innerhalb der Bilder dargestellt und bieten eine hervorragende Möglichkeit zur Identifizierung. Allerdings können diese während des Messprozesses zu Überbelichtungen innerhalb der PMD-Aufnahme und somit zur Verfälschung der gemessenen Phasenwerte führen. Fehlmessungen aufgrund überbelichteter Pixel lassen sich jedoch durch Überwachung der Intensitätswerte erkennen. Sollten diese ein Maximum überschreiten, gilt es durch Reduzierung der Integrationszeit einer Überbelichtung entgegenzuwirken.

Die Erkennung des Kalibrierobjekts erfolgt in mehreren Stufen [Berns09b]. Diesbezüglich sind in Abbildung 6.2.4 die einzelnen Phasen des Erkennungsalgorithmus dargestellt.

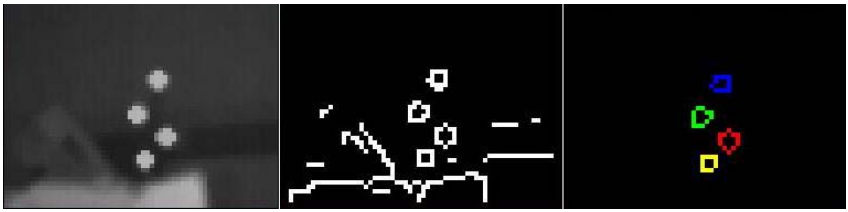


Abbildung 6.2.4: Algorithmenabfolge zur Erkennung des Referenzobjekts

Der erste Schritt zur Erkennung des Referenzkörpers ist die Aufbereitung des Grauwertbildes und der 3D-Daten durch Verwendung eines Mean-Filters [Jähne05]. Hierdurch werden die Auswirkungen von Messfehlern in den PMD-Daten weitestgehend eliminiert. Sind die Daten aufbereitet, startet die eigentliche Erkennung. Durch einen Canny-Filter [Canny86] kann ein Kantenbild aus den Intensitätswerten extrahiert werden.

Das Canny-Filter ist ein Kantenerkennungsfilter, welches Optimalität bezüglich Erkennung, Lokalisierung und Ansprechverhalten bietet. Dies bedeutet, dass nur Kanten erkannt werden, welche eine geringe Abweichung zu den realen Kanten aufweisen, das

die Kanten nicht mehrfach detektiert werden und ihre Breite lediglich ein Pixel beträgt. Abbildung 6.2.4b zeigt das Ergebnis des Canny-Filters, indem die extrahierten Kantenpixel dargestellt sind. Durch ein Regiongrowing-Verfahren können alle im Kantenbild befindlichen Konturen ermittelt werden. Im Weiteren werden nur die Kanten berücksichtigt, die eine geschlossene Kontur aufweisen. Dieses Verfahren ermöglicht zusätzlich die Unterscheidung der verschiedenen Konturen. Da der Referenzkörper aus vier gleichen Kugeln besteht, werden durch verschiedene, definierte Parameter ähnliche Konturen in dem Bild gesucht. Das Resultat ist in Abbildung 6.2.4c dargestellt.

Abschließend werden, durch Mittelung aller 3D-Werte, der auf den Konturen liegenden Pixel, die Schwerpunkte der Kugeln des Referenzkörpers ermittelt. Sind die Positionen der Kugeln bekannt, kann auf die Position des Kalibrierkörpers geschlossen werden.

6.2.4 Analyse der Kamerakalibrierung

In Abschnitt 6.2 wurde der Algorithmus zur Bestimmung der Transformationsmatrix zwischen Roboter- und Kamerakoordinatensystem beschrieben. Dem zufolge werden die notwendigen Schritte nochmals an realen Messdaten verifiziert.

Abbildung 6.2.5 zeigt die Messdaten zur Bestimmung der Translation zwischen Kamera- und Roboterkoordinatensystem. Hierbei wird nach Abschnitt 6.2.1 der TCP des Roboters parallel zu den Achsen des Roboterkoordinatensystems verfahren. Die Messwerte der verschiedenen Achsen sind farblich gekennzeichnet.

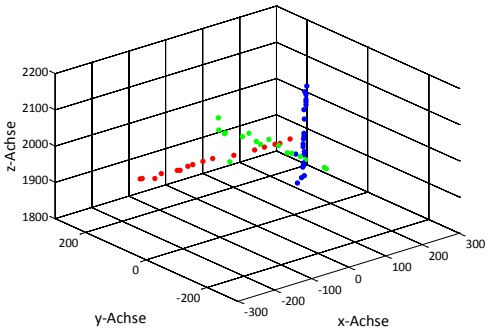


Abbildung 6.2.5: Messdaten zur Bestimmung der Rotation

Auf Basis der aufgenommenen Daten können die Richtungsvektoren der Geraden ermittelt werden. Dies ermöglicht die Berechnung der Orientierung des aufgespannten Koordinatensystems zum Kamerakoordinatensystem. Aufgrund der Parallelität zum Roboterkoordinatensystem kann auf die Rotation zwischen beiden Koordinatensystemen geschlossen werden. In Abbildung 6.2.5 sind Messfehler zu erkennen, die aus Fehldetektionen des Referenzkörpers resultieren. Zur Minimierung von deren Einfluss, wird zur Bestimmung der Richtungsvektoren ebenfalls der RANSAC-Algorithmus verwendet.

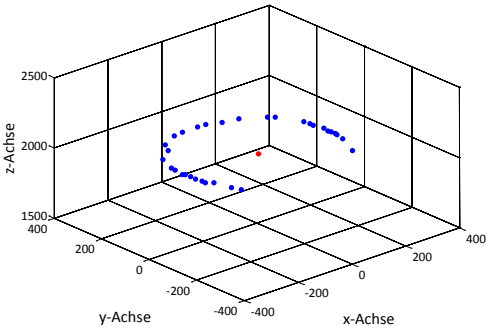


Abbildung 6.2.6: Messdaten zur Bestimmung der Translation

Abbildung 6.2.6 zeigt die Punktwolke, welche zur Ermittlung der TCP-Position im Kamerakoordinatensystem aufgenommen wird. Zu diesem Zweck wird der Referenzkörper auf einer Kreisbahn um den TCP bewegt. Der Mittelpunkt, welcher in Abbildung 6.2.6 eingetragen ist, entspricht somit der aktuellen TCP-Position. Bei bekannter Position kann nach Abschnitt 6.2.2 auf die Translation zwischen den entsprechenden Koordinatensystemen geschlossen werden.

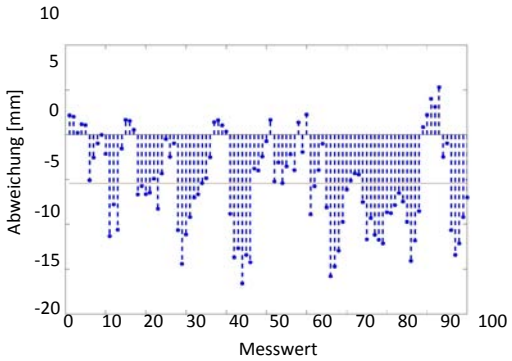


Abbildung 6.2.7: Abweichungen zwischen der Roboterposition bestimmt mit der Kamera und der realen Roboterposition

Zur Überprüfung der Kalibriergenauigkeit werden mit dem Roboter 100 zufällige, im Arbeitsbereich befindliche Positionen angefahren, welche mit der PMD-Kamera detektiert werden. Über die Position des Referenzobjekts kann auf die Position des TCP des Roboters geschlossen werden. Die ermittelte Position wird anschließend in das Roboterkoordinatensystem transformiert und mit der aus der Robotersteuerung erhaltenen Position des TCP's verglichen. Entsprechend sind in dem Graphen in Abbildung 6.2.7 die absoluten Abweichungen des TCP's eingetragen. Die Abweichung d liegt in den Grenzen:

$$-17mm \leq d \leq 6mm$$

Die berechnete Roboterposition weicht im Mittel um $\hat{d} = 5{,}6mm$ von der realen Position ab. Somit liegt die Genauigkeit der Kalibrierung in einem für die Verwendung der PMD-Technik adäquaten Bereich und ist geeignet zur PMD gestützten Überwachung des Roboterarbeitsraums.

6.3 Hindernis- und Objekterkennung

Nachdem die Kalibrierung zwischen Kamera- und Roboterkoordinatensystem durchgeführt wurde, ist die Weiterverarbeitung der 3D-Kameradaten zur Hindernis- und Objekterkennung möglich. Diese basiert auf dem Vergleich der aktuellen PMD-Aufnahmen mit einem Referenzbild. Das Referenzbild wird zu Beginn während einer Initialisierungsphase aufgenommen. Durch Anwendung eines Subtraktionsverfahrens können, innerhalb des aktuellen Bildes, von dem Referenzbild abweichende Pixel auf

einfache Weise ermittelt werden. Nachteilig hierbei ist, dass der Roboter, sobald dieser seine Position ändert, selbst als Hindernis erkannt wird. Folglich muss vor Anwendung des Subtraktionsverfahrens und während der Ermittlung eines Referenzbilds der Roboter innerhalb der PMD-Aufnahme erkannt und die entsprechenden Bildausschnitte von der weiteren Betrachtung ausgeschlossen werden.

6.3.1 Erkennung des Roboters innerhalb einer PMD-Aufnahme

Entwicklung eines vereinfachten Robotermodells

Zur Registrierung des Roboters innerhalb einer PMD-Aufnahme wird im Folgenden ein vereinfachtes Robotermodell entwickelt.

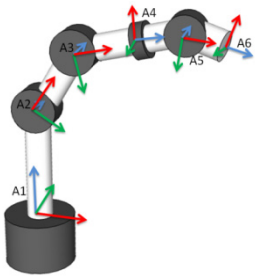


Abbildung 6.3.1: Koordinatensystemdefinition

Die Grundlagen des Modells basieren auf der Berechnung der Vorwärtskinematik nach dem von Denavit und Hartenbeg im Jahre 1955 entwickelten Verfahren. Hierbei wird jedem Gelenk des 6-Gelenkroboters ein Koordinatensystem zugewiesen. Der Ursprung der Koordinatensysteme liegt in den jeweiligen Gelenkmittelpunkten. Die Richtung der z-Achse wird in Richtung des Drehgelenks definiert. Die x-Achse ist senkrecht zur z-Achse und schneidet die z-Achse des vorherigen Gelenks im Ursprung des Selbigen. Zur Veranschaulichung der Denavit-Hartenberg Notation dient Abbildung 6.3.1. Nach [Lamikiz08] können die Transformationsmatrizen $A_1, A_2, \dots, A_6$ zwischen den Gelenkachsen wie folgt beschrieben werden:

$$A_i = \begin{bmatrix} \cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6.10)$$

θ_i entspricht dem Winkel, um welchen das $(i-1)$-te Koordinatensystem gedreht werden muss, so dass die Achse z_{i-1} und die Achse z_i parallel zueinander ausgerichtet sind. Die Entfernung auf der z-Achse zwischen den entsprechenden Gelenkkoordinatensystemursprüngen beträgt d_i. Demzufolge ist a_i die Distanz entlang der x_i-Achse, zwischen dem Schnittpunkt von x_i-Achse und dem Ursprung des i-ten Koordinatensystems. α_i ist der Winkel zwischen der Achse z_{i-1} des $(i-1)$-ten Koordinatensystems und der Achse z_i des i-ten Koordinatensystems.

Die Parameter θ_i, α_i, d_i und a_i sind aus der Robotersteuerung beziehungsweise aus der Robotergeometrie entnehmbar. Unter Anwendung der Vorwärtskinematik von Denhavit-Hartenberg kann für jede Roboterposition die Position und Orientierung der jeweiligen Gelenkkoordinatensysteme ermittelt werden.

Dies eröffnet die Möglichkeit der Definition von einfachen Hüllkörpern, wie in Abbildung 6.3.2 dargestellt ist, welche die einzelnen Glieder des Roboters auf einfache Weise nachbilden. Diese definiert die Eckpunkte des jeweiligen Hüllkörpers in den jeweiligen Gelenkkoordinatensystemen. Über die Transformationsmatrizen A_1, $A_2, \dots, A_6$ können die Hüllkörper $H_1, H_2, \dots, H_6$ in das Roboterkoordinatensystem überführt werden [Berns09a].

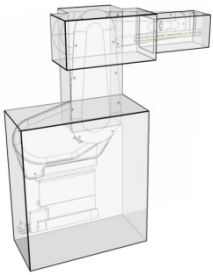


Abbildung 6.3.2: Hüllkörpermodell des KUKA KR3

Da eine exakte Darstellung des Roboters für die Erkennung des Selbigen innerhalb der PMD-Aufnahmen nicht zwangsläufig notwendig ist, bietet das entwickelte vereinfachte Hüllkörpermodell aufgrund der sehr schnellen und einfachen Berechnungsweise ein adäquates Werkzeug zur Filterung der den Roboter beinhaltenden Bildausschnitte.

Filterung der PMD-Aufnahme

Ausgangssituation für die Filterung ist eine PMD-Aufnahme, wie sie auf Abbildung 6.3.3 dargestellt ist. Diese zeigt den auf einem Tisch installierten KUKA KR3.

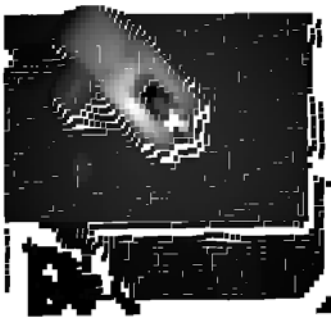


Abbildung 6.3.3: PMD-Aufnahme der Roboterzelle

Hierbei erfasst die über den Roboter angebrachte PMD-Kamera lediglich die Oberfläche der Szene. Die Tiefe eines Objekts kann nicht ermittelt werden. Des Weiteren werden Objekte innerhalb des Bildes von anderen Objekten überlagert und können nicht erfasst werden. Im technischen Sinn entspricht dies einer 2 1/2D Darstellung der Szene. Eine 3D-Szene entspräche einer vollständigen dreidimensionalen Modellierung der Objekte. Für die Filterung des Roboters, sowie für die spätere Kollisionsberechnung werden alle Pixel im Folgenden durch Hüllkörper in Form von Quadern umschlossen. Somit werden alle Bereiche hinter der detektierten Oberfläche als besetzt markiert. Diese Darstellung ermöglicht zum einen die Kollisionsberechnung zwischen Roboter und überwachter Umgebung, zum anderen kann der Roboter durch Berechnung von Überschneidungen der generierten Hüllkörper mit denen des Robotermodells auf einfache Weise herausgefiltert werden.

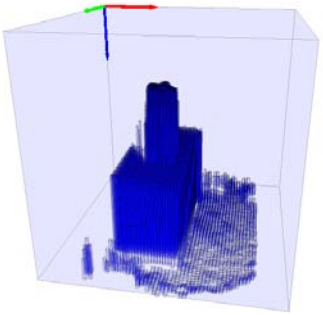


Abbildung 6.3.4: Hüllkörperdarstellung einer PMD-Aufnahme

Gemäß dessen sind die Hüllkörper, entstehend aus einer PMD-Aufnahme, in Abbildung 6.3.4 illustriert. Zu erkennen ist der Roboter, welcher auf einem Tisch montiert ist. Das eingetragene Koordinatensystem entspricht der Ausrichtung der PMD-Kamera. Diese Darstellung bildet die Basis für das Erkennen des Roboters im PMD-Bild. Hierzu wird die Hüllkörperdarstellung zunächst in das Roboterkoordinatensystem überführt. Nach der Transformation werden alle aus der PMD-Aufnahme berechneten Hüllkörper mit denen des Roboters unter Anwendung des Separating-Plane-Theorems [Evgeni97] auf eventuelle Kollision überprüft. Das Separating-Plane-Theorem ist eine dreidimensionale Erweiterung des in Kapitel 5.7 beschriebenen Separating-Axis-Theorems. Unterschied ist hier die Projektion der Hüllkörper, welche zur Erkennung von Überschneidungen auf die jeweiligen Normalen der Seitenflächen projiziert werden und nicht auf die Senkrechte der Achsen. Dies jedoch führt zu einer erhöhten Berechnungszeit. Um jedoch eine Echtzeitfähigkeit des Erkennungsalgorithmus zu gewährleisten, muss eine Validierung, ob ein Pixel im Nahbereich des Roboters liegt, stattfinden. Hierzu wird die Position des Roboter-TCP's berechnet. Alle Pixel p_i für die gilt:

$$\sqrt{x(p_i)^2 + y(p_i)^2} < \sqrt{x(TCP)^2 + y(TCP)^2} \tag{6.11}$$

liegen innerhalb des aktuellen Bewegungsbereichs des Roboters. Nur diese werden während der Filterung mit dem Separating-Plane-Theorem weiter berücksichtigt.

Sollte bei Anwendung des Separating-Plane-Theorems eine Überlappung der Hüllkörper festgestellt werden, zeigt das betreffende Pixel den Roboter und wird aus dem Bild herausgefiltert beziehungsweise in diesem markiert.

a) b)

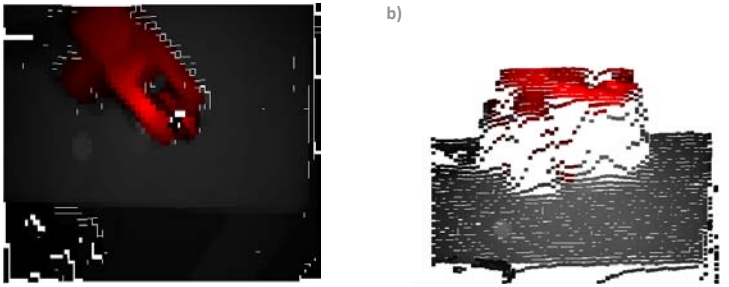


Abbildung 6.3.5: PMD-Aufnahme mit farblich markiertem Roboter a) aus Kameraperspektive b) seitlich verdreht

Abbildung 6.3.5 zeigt PMD-Aufnahmen nach der Erkennung des Roboters. Die betreffenden Pixel sind hier farblich gekennzeichnet. Problematisch bei der Filterung und der späteren Objekt- und Kollisionserkennung unter Benutzung einer PMD-Kamera sind die sogenannten „Flying-Pixel“. Dies sind Pixel, welche scheinbar in der Luft schweben. Sie entstehen, sollte moduliertes Licht von verschiedenen Objekten auf ein Pixel reflektiert werden. Dies geschieht beispielsweise an Objektkanten oder sollten sich Objekte während der PMD-Bildaufnahme im Sichtfeld der Kamera schnell bewegen. Es kommt zu einer Überlagerung der verschiedenen Lichtsignale und somit zu fehlerhafter Distanzmessung. „Flying-Pixel“ sind nur schwer erkennbar und können nur schlecht gefiltert werden. Bei der Erkennung des Roboters im PMD-Bild bietet die Vergrößerung der Hüllkörper des Robotermodells eine gute Methode zur Reduzierung der Auswirkung von „Flying-Pixeln“ innerhalb des Detektionsalgorithmus.

6.3.2 Objekterkennung innerhalb von PMD-Bildern

Wie anfangs geschildert, erfolgt die Überwachung des Roboterarbeitsraums, also die Erkennung von fremden Objekten mit der PMD-Kamera durch ein Subtraktionsverfahren. Hierbei wird jede PMD-Aufnahme mit einem Referenzbild durch Subtraktion der 3D-Daten verglichen. Pixel für welche gilt:

$$i_z(u,v) = ai_z(u,v) - ri_z(u,v) > z_{Offset} \tag{6.12}$$

werden als potentielle Pixel markiert, die ein fremdes Objekt zeigen könnten. Hierbei entspricht $ai_z(u,v)$ den z-Werten des aktuellen und $ri_z(u,v)$ denen des Referenz-PMD-Bildes. z_{Offset} wird so gewählt, dass der Wert über dem Entfernungsmess-

rauschen der verwendeten PMD-Kamera liegt. Dies verhindert, dass Pixel irrtümlich markiert und fälschlicherweise als Objekt interpretiert werden.

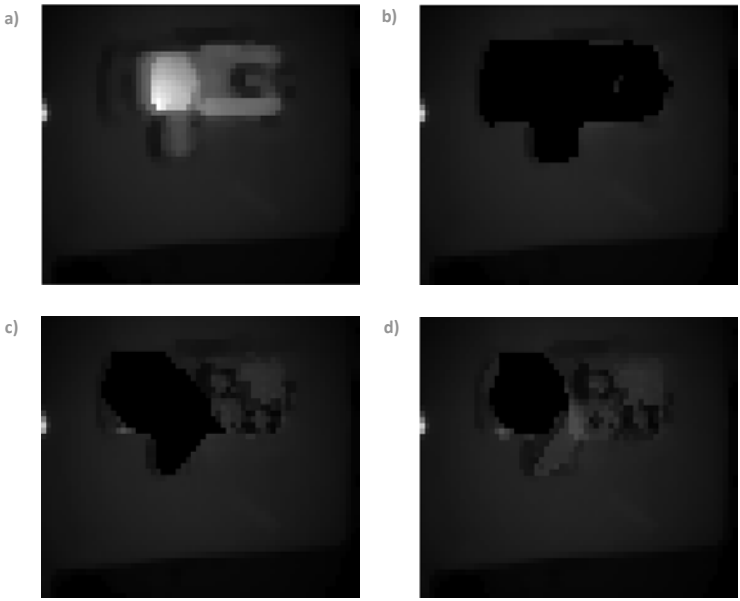


Abbildung 6.3.6: Generierung eines Referenzbildes

In Abbildung 6.3.6 a-d ist die Aufnahme des Referenzbilds dargestellt. Abbildung 6.3.6a zeigt die erste PMD-Aufnahme. Aus dieser wird mittels beschriebener Methode der Roboter herausgefiltert, siehe Abbildung 6.3.6b. Die Pixel die den Roboter zeigten sind undefiniert und werden schwarz dargestellt. Die spätere Objekterkennung mittels Subtraktionsverfahren kann jedoch nur für die definierten Bereiche erfolgen. Aus diesem Grund wird der Roboter auf einer definierten Bahn verfahren. Währenddessen werden weitere Bilder aufgenommen, welche mit dem ersten Bild überlagert werden. Auf diese Weise entsteht ein Referenzbild, wie auf Abbildung 6.3.6d dargestellt, das nur einen minimalen undefinierten Bereich aufweist. Dieser Bereich entspricht dem unbeweglichen Roboterrumpf. Da der Roboter diesen Bereich unabhängig von seiner Position und Orientierung immer verdeckt, braucht dieser bei der Objekterkennung nicht berücksichtigt werden.

Durch den Vergleich des erstellten Referenzbilds mit den aktuellen PMD-Aufnahmen kann unter Verwendung von Gl.(6.12) ein Binärbild $I(u, v)$ extrahiert werden. Es gilt:

$$I(u,v) = \begin{cases} 0, & i_z(u,v) < z_{Offset} \\ 1, & i_z(u,v) \geq z_{Offset} \end{cases} \tag{6.13}$$

Zum Extrahieren einzelner Objekte aus dem Binärbild müssen die verschiedenen Bildregionen, welche durch verbundene Pixel definiert werden, bestimmt werden. Hierfür muss ein Algorithmus verwendet werden, mit welchem eine Aussage über die Zugehörigkeit der Pixel getroffen werden kann, wie viele Regionen gebildet werden können und wie viele Pixel die einzelnen Regionen beinhalten. In der Literatur [Burger06] werden diese Verfahren unter dem Begriff „region labeling" zusammengefasst. Diese dienen dem schrittweisen Einteilen der Pixel in Regionen. Jedem Pixel innerhalb einer Region wird ein entsprechender Wert, die Identifikationsnummer, zugewiesen. Das Binärbild wird wie folgt erweitert:

$$I(u,v) = \begin{cases} 0, & i_z(u,v) < z_{Offset} \\ 1, & i_z(u,v) \geq z_{Offset} \\ 2, \dots n, & Regionenmarkierung \end{cases} \tag{6.14}$$

Zur Regionenmarkierung stehen nach [Burger06] verschiedene Verfahren zur Verfügung. Es wird zwischen Floodfill-Verfahren (flood - Flutwelle) und sequentiellen Regionensuchverfahren unterschieden. Aufgrund der lateralen Auflösung der PMD-Kamera stellte sich das Floodfill-Verfahren als geeignetes Verfahren zur Regionensuche heraus. Hierbei wird zunächst ein mit „1"-markiertes Pixel gesucht. Von diesem ausgehend werden alle verbundenen Pixel gesucht und entsprechend mit der Regionenmarkierung nach Gl.(6.14) versehen. Dies geschieht ähnlich einer Flutwelle, die sich über die jeweilige Region ausbreitet.

Bei der Berechnung benachbarter Pixel wird nach [Jähne05] zwischen verschiedenen Nachbarschaftbedingungen differenziert. Es wird unterschieden zwischen 4er- und 8er Nachbarschaft. Bei der 4er-Nachbarschaft gelten nur die Pixel oberhalb und seitlich des jeweiligen Pixels als benachbart. Bei der 8er Nachbarschaft gelten zusätzlich die Pixel schräg oberhalb und unterhalb des Pixels als benachbart. Abhängig von der Wahl des Nachbarschaftsoperators führt das Regionensuchverfahren zu unterschiedlichen Ergebnissen. Für unsere Zwecke erweist sich die Wahl der 8er-Nachbarschaft am geeignetsten.

Die Floodfill-Verfahren können nach [Burger06] in drei verschiedene Methoden aufgeteilt werden. Die Methoden unterscheiden sich in der Art, wie die noch zu untersuchenden Pixel exploriert werden. Es wird unterschieden in ein rekursives Suchverfahren, ein iteratives Breiten- und ein iteratives Tiefensuchverfahren.

Nachteilig bei dem rekursiven Verfahren ist, dass die Rekursionstiefe proportional zu der Anzahl der Pixel in der entsprechenden Region und somit auch proportional zur Größe des erforderlichen Stack-Speichers ist und dieser bei großen Regionen schnell erschöpft sein kann. Da es sich in unserer Anwendung jedoch um ein Bild mit geringer lateraler Auflösung und somit um Regionen mit geringer Pixelanzahl handelt, muss dieses Defizit nicht weiter berücksichtigt werden. Aus Gründen der einfachen Umsetzung und der geringen Rechenzeit des rekursiven Verfahrens, wurde auf die Umsetzung der beiden iterativen Verfahren verzichtet und das rekursive Floodfill-Verfahren zur Regionensuche implementiert. Algorithmus 6.1 und Algorithmus 6.2 zeigen die implementierte Methode zum Regionensuchverfahren.

RegionLabeling $(\boldsymbol{I})$

Initialisiere die Label-Nummer: $label = 2$
Iterative Suche über das Binärbild: $I(u, v)$

Wenn $I(u, v) = 1$ ist, dann:
$FloodFill(I, u+1, v-1, label)$
$FloodFill(I, u+1, v+1, label)$
$FloodFill(I, u+1, v, \quad label)$
$FloodFill(I, u, \quad v-1, \ label)$
$FloodFill(I, u, \quad v+1, \ label)$
$FloodFill(I, u-1, v-1, label)$
$FloodFill(I, u-1, v, \quad label)$
$FloodFill(I, u-1, v+1, label)$
$label = label + 1$

Setzte die iterative Suche fort bis alle Pixel überprüft wurden.
return

Algorithmus 6.1: Ablauf des Region Labeling Verfahrens

$\boldsymbol{FloodFill(I, u, v, label)}$

Wenn (u,v) innerhalb der Bildgrenzen liegen und $I(u, v) = 1$ ist, dann:

$$
\begin{aligned}
&I(u, v) = label \\
&FloodFill(I, u + 1, v - 1, label) \\
&FloodFill(I, u + 1, v + 1, label) \\
&FloodFill(I, u + 1, v, \quad label) \\
&FloodFill(I, u, \quad v - 1, \ label) \\
&FloodFill(I, u, \quad v + 1, \ label) \\
&FloodFill(I, u - 1, v - 1, label) \\
&FloodFill(I, u - 1, v, \quad label) \\
&FloodFill(I, u - 1, v + 1, label)
\end{aligned}
$$

$\boldsymbol{return}$

Algorithmus 6.2: Ablauf des Floodfill-Verfahrens

Ist das Regionensuchverfahren durchgeführt, kann die Anzahl der zu den Regionen gehörenden Pixel bestimmt werden. Gilt für die Anzahl der Pixel $P_{Num}(label)$:

$$P_{Num}(label) < min_{Num} \tag{6.15}$$

wobei min_{Num} der minimalen Anzahl von zu einer Region gehöriger Pixel entspricht, wird die Region gelöscht und zur Objekterkennung nicht weiter berücksichtigt. min_{Num} wird so gewählt, dass einzelne fälschlich, durch Messrauschen oder durch eine zu schnelle Robotergeschwindigkeit, aufgrund der „Flying-Pixel“ Problematik extrahierte Pixel bei der Filterung nicht entfernt werden. Sollte eine Region übrig bleiben, stellt diese ein erkanntes Objekt dar. Im nächsten Schritt wird die minimale Distanz der übrigen Regionen zum Roboterkoordinatenursprung berechnet. Abhängig von der ermittelten Distanz können dem Roboter verschiedene Fahrbefehle vorgegeben werden.

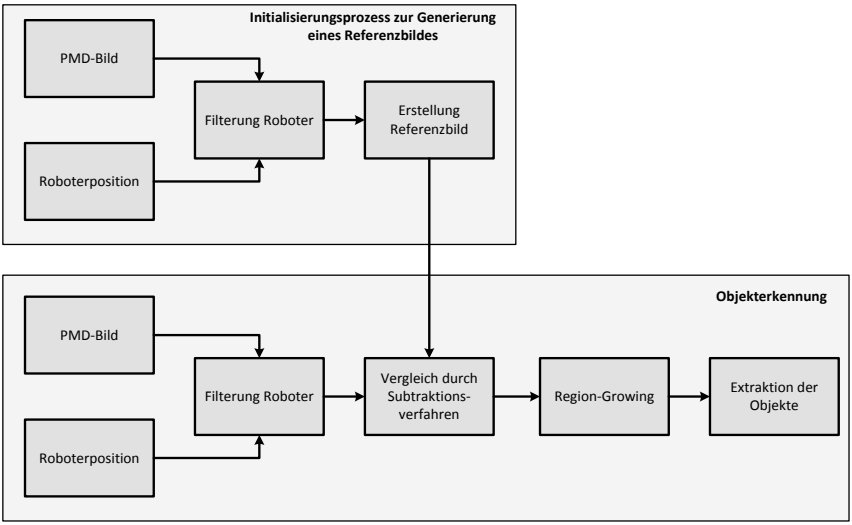


Abbildung 6.3.7: Flussdiagramm zum Objekterkennungsverfahren

Zusammenfassend stellt sich der bisher implementierte Algorithmus zur Erkennung von Hindernissen, wie in Abbildung 6.3.7 skizziert ist, dar. In einem Initialisierungsprozess werden mehrere PMD-Aufnahmen, bei denen der Roboter herausgefiltert wird, zu einem Referenzbild fusioniert. Die eigentliche Objekterkennung vergleicht anschließend das erstellte Referenzbild mit dem aktuellen PMD-Bild, aus welchem ebenfalls der Roboter herausgefiltert wird. Sollten abweichende Pixel erkannt werden, werden diese mit einem Regionensuchverfahren zusammengefügt. Abhängig von der Anzahl zu einer Region gehöhrender Pixel, werden die einzelnen Regionen als neue Hindernisse aufgefasst. Die Entfernung der detektierten Hindernisse zum Roboter, bestimmt die Geschwindigkeit und die Trajektorie des Robotersystems. Sollte die Distanz größer als eine definierte Minimalentfernung sein, kann der Roboter ungehindert weiterverfahren werden. Bei Unterschreitung einer definierten Grenze wird der Roboter entweder verlangsamt oder angehalten, um eine Gefährdung von Personen auszuschließen.

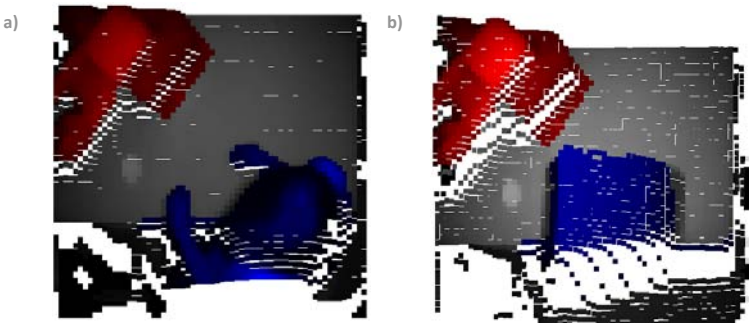


Abbildung 6.3.8: Erkennung von Hindernissen im Roboterarbeitsraum: a) Person b) Karton

Abbildung 6.3.8 a und b zeigen erkannte Hindernisse im Roboterarbeitsraum. Der Roboter ist in rot und das Hindernis in blau dargestellt. Nach dem Erkennen eines fremden Objekts beendet der Roboter das Abfahren der vorgegebenen Trajektorie. Erst nachdem dieses aus dem Arbeitsraum entfernt wurde, fährt der Roboter mit der Ausführung fort.

6.4 Bahnplanung

In Abschnitt 6.3 wurden Algorithmen zur Erkennung von fremden Objekten und Personen im Roboterarbeitsraum implementiert und diskutiert. Diese ermöglichen den gefahrlosen Umgang mit dem Robotersystem ohne Sicherheitszäune. Sollte eine Person den Roboterarbeitsraum betreten, wird diese mit der PMD-Kamera erfasst und dessen Position bezüglich des Roboterkoordinatenursprungs ermittelt. Aufgrund der minimalen Distanz können Entscheidungen über den Bewegungsablauf des Manipulators getroffen werden. Bisher wird der Roboter abgeschaltet, sollte die minimale Distanz eine vordefinierte Schwelle unterschreiten. Entfernt sich die Person wieder aus der Roboterzelle bzw. wird der fremde Gegenstand wieder entfernt, fährt der Roboter mit der Ausführung der programmierten Trajektorie fort. Dies gewährleistet einen sicheren Umgang mit dem Industrieroboter auch ohne Einsatz von Sicherheitszäunen.

Hiermit ist das Potential bei der Verwendung einer PMD-Kamera zur Überwachung von Roboterarbeitsräumen jedoch nicht erschöpft. Sollten Gegenstände in dem Roboterarbeitsraum positioniert werden, muss der Roboter nicht zwangsläufig stehen bleiben. Ebenso wie am Beispiel der mobilen Robotik in Kapitel 5 bereits gezeigt, können auch hier die 3D-Daten der PMD-Kamera dazu verwendet werden, die

programmierte Trajektorie auf Kollision mit den Fremdobjekten zu überprüfen. Sollte der Roboter auf der vorgegebenen Trajektorie nicht kollidieren, kann dieser mit der Bewegung entlang der Trajektorie fortfahren. Bei erkannter Kollision besteht darüber hinaus auch die Möglichkeit eine alternative Trajektorie mittels eines reaktiven Online-Bahnplaners zu berechnen.

Im Folgenden wird die Implementierung eines Online-Bahnplaners für den Industrieroboter beschrieben und diskutiert. Angesichts dessen werden zunächst die in der Literatur beschriebenen Standardverfahren behandelt. Anschließend wird auf die im Projekt umgesetzte Variante eingegangen.

6.4.1 Verfahren zur Bahnplanung in der Industrierobotik

Bahnplanungsverfahren beschäftigen sich mit der Suche eines Pfades zwischen einer vorgegebenen Startposition und einer vorgegeben Zielposition. In der Literatur [Latombe91, LaValle06] gibt es verschiedene Methoden zur Bahnplanung von Robotern. Diese wurden unter anderem bereits in Kapitel 5.5 erläutert. Die Bahnplanungsverfahren unterscheiden sich zum einen in der Art der Umweltrepräsentation und zum anderen in der Art der Pfadsuche in der gewählten Umgebungsdarstellung.

Bei den Bahnplanungsverfahren wird zwischen der Planung im geometrischen Raum und der Planung im Konfigurationsraum unterschieden [LaValle06].

Bei der geometrischen Bahnplanung findet die Planung in kartesischen Koordinaten statt. Die Roboterposition ist somit eine Funktion der Koordinaten x, y, z und der Orientierungen α, β und γ des Roboters beziehungsweise bei Manipulatoren des Roboterendeffektors. Geometrische Bahnplanungsverfahren werden am häufigsten im Bereich der mobilen Robotik eingesetzt. Hier werden zumeist Sensoren verwendet, welche Hindernisse detektieren. Die erkannten Hindernisse müssen in die Umweltrepräsentation des Roboters aufgenommen werden, damit diese für die Planung alternativer Trajektorien berücksichtigt werden können. Vorteil dieser Darstellungsmethode ist die direkte Verarbeitung der Hindernisinformationen in der gewählten Darstellungsmethode. Eine zeitintensive Konvertierung der Hindernisse entfällt. Deswegen wird in Kapitel 5 diese Art der Umweltdarstellung verwendet.

Eine alternative Möglichkeit der Darstellung ist die Definition eines Konfigurationsraums. Diese Methode wurde erstmals von Lozano-Pèrez [Lozano83] in den späten 70ern zur Bahnplanung verwendet. Der Konfigurationsraum eines Roboters wird

beschrieben durch seine Gelenkstellungen. Somit ist die Dimension eines Konfigurationsraums abhängig von der Anzahl der Robotergelenke [LaValle06]. Bei dem in dieser Arbeit verwendeten KR3 entspricht dies, aufgrund der sechs Gelenke, einem 6-dimensionalen Konfigurationsraum. Die Summe aller möglichen Roboter-konfigurationen entspricht dem Konfigurationsraum C. Der Konfigurationsraum kann unterteilt werden in die Konfigurationen C_{free} und C_{Obs}. C_{free} entspricht allen erlaubten, kollisionsfreien Konfigurationen. C_{Obs} beinhaltet alle Konfigurationen, bei denen der Roboter mit seiner Umwelt oder auch mit sich selber kollidieren würde. Vorteil der Verwendung der Konfigurationsraumdarstellung für Bahnplanungsverfahren ist die Reduzierung der Robotergeometrie auf einen Punkt. Die Bahnplanung ist folglich unabhängig von der Robotergeometrie. Dies führt zu einer deutlichen Beschleunigung des eigentlichen Bahnplanungsverfahrens. Allerdings muss zunächst die kartesische Darstellung der Hindernisumgebung in den Konfigurationsraum überführt werden.

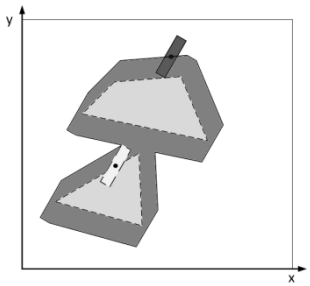


Abbildung 6.4.1: Konvertierung der Hindernisse in den Konfigurationsraum

Abbildung 6.4.1 zeigt die Generierung einer Konfigurationsraumdarstellung eines rechteckförmigen Roboters innerhalb einer zweidimensionalen Umwelt. Der Konfigurationsraum entspricht in diesem Fall einem dreidimensionalen Raum mit den Achsen x, y und der Orientierung θ des Roboters. Die Hindernisse sind in einem hellen grau dargestellt. Diese werden durch Bildung der Minkowski-Summe [Latombe91] erweitert. Abbildung 6.4.1 zeigt, dunkelgrau dargestellt, die resultierende Hindernisumgebung eines rechteckförmigen Roboters in Konfigurationsraum-darstellung für eine definierte Orientierung des Roboters.

Die Konfigurationsraumdarstellung bietet eine sehr gute Grundlage zur Bahnplanung von autonomen Robotern. Sollte jedoch die Hindernisumgebung dynamisch sein, gestaltet sich das Hinzufügen von Hindernissen als problematisch. Die Berechnung der Hindernisse im Konfigurationsraum ist zeit- und rechenintensiv, da jede mögliche

Roboterkonfiguration auf Kollision überprüft werden muss. Dies steht im Wiederspruch zu einer schnellen reaktiven Bahnplanung, welche für die genannte Anwendung erforderlich ist. Aus diesem Grund findet die Bahnplanung im Folgenden zwar im Konfigurationsraum statt, die Berechnung des Konfigurationsraums erfolgt jedoch erst während der Planung für die während dessen verwendeten Gelenkkonfigurationen, so dass die Berechnung des vollständigen Konfigurationsraums entfällt.

6.4.2 Umgebungsdarstellung

Zur Umgebungsdarstellung wird eine Methode benötigt, welche zum einen die Fusion von der CAD-Darstellung der Roboterzelle mit den 3D-Daten der PMD-Kamera und zum anderen eine schnelle Kollisionsabfrage ermöglicht. Basierend auf den Erfahrungen mit der 2-dimensionalen Quadtree-Darstellung der Umgebungskarte im Bereich der mobilen Robotik wird diese Darstellung lediglich um eine dritte Dimension erweitert. Mit der auf diese Weise entstehenden Octree-Darstellung kann der Roboterarbeitsraum in besetzte und unbesetzte Partitionen aufgeteilt werden. Dies ermöglicht eine schnelle Kollisionsabfrage.

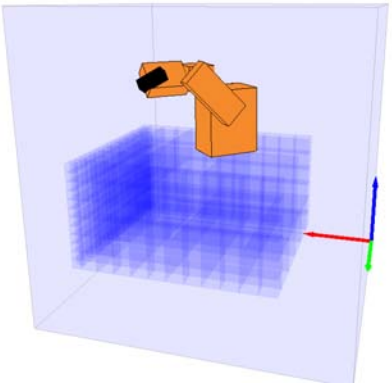


Abbildung 6.4.2: Octree-Darstellung des Roboterarbeitsraums inklusive des Robotermodells

Abbildung 6.4.2 zeigt die Darstellung des Roboterarbeitsraums mit Octrees inklusive des Robotermodells. Zur Kollisionsberechnung können unter Berücksichtigung der Gelenkstellungen über die Vorwärtskinematik die Positionen der einzelnen Roboterglieder berechnet werden. Durch einfache Bereichsanfragen kann ermittelt werden, ob diese sich in einem als besetzt markierten Bereich befinden. Besetzte Octrees sind in der Karte dargestellt.

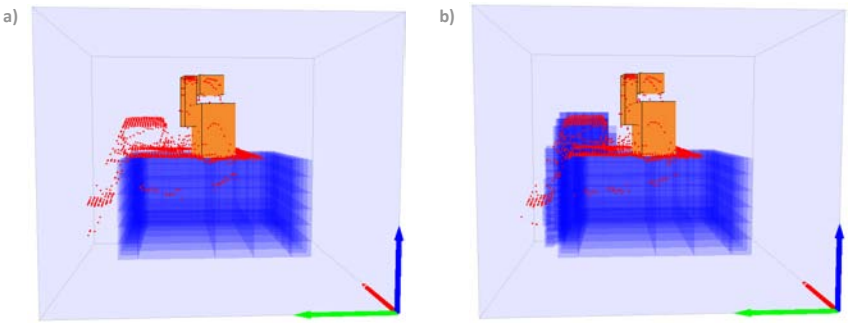


Abbildung 6.4.3: Octree-Darstellung des Roboterarbeitsraum a) mit 3D-Kameradaten b) Fusion des Arbeitsraum-Octreemodells mit den Kameradaten in gemeinsames Octreemodell

In Abbildung 6.4.3a ist das Octreemodell der Roboterumgebung, inklusive der in das Roboterkoordinatensystem überführten PMD-Daten zu sehen. Hier ist nochmals die Qualität der Roboter-Kamera-Kalibrierung ersichtlich. Die Oberfläche des Tisches im Octreemodell, ebenso wie die des Roboters, entspricht der mit der PMD-Kamera detektierten Oberfläche. Links neben dem Roboter befindet sich ein Karton, welcher als ein fremdes Objekt im Roboterarbeitsraum, aufgrund der implementierten Erkennungsalgorithmen, detektiert wird. Über die Hüllkörperdarstellung der entsprechenden Pixel besteht die Möglichkeit das erkannte Hindernis im Octreemodell hinzuzufügen. Diesbezüglich zeigt Abbildung 6.4.3b das um das Hindernis erweiterte Octreemodell. Dieses kann im Folgenden zur Planung einer alternativen Trajektorie herangezogen werden.

6.4.3 Umsetzung des Bahnplanungsverfahrens

Die Robotertrajektorie kann zum Beispiel bei einer Palettierungsaufgabe in drei verschiedene Abschnitte gegliedert werden. Diese sind, wie in Abbildung 6.4.4 dargestellt, die Bewegung von einem Startpunkt hin zu einem Stützpunkt oberhalb der Ausgangsposition, einer großen Translationsbewegung zu einem Stützpunkt oberhalb des Zielpunkts und von dort zum Ziel, wo beispielsweise ein Gegenstand mit dem Roboter gegriffen oder abgelegt werden kann.

Abbildung 6.4.4: Gliederung der Bahnplanung

Im Folgenden soll nur die Planung der großen Translationsbewegung zwischen den Stützpunkten behandelt werden, da die genaue Greif- und Positionierungsplanung bei gegebener Installation der PMD-Kamera oberhalb des Roboters nicht geeignet ist, weil die Distanzwerte bei der gegebenen Entfernung zwischen Roboter und Kamera für eine präzise Greifplanung zu ungenau sind. Hierfür würde sich die Installation der Kamera am Greifer empfehlen. Ist im Folgenden von Start- und Zielpunkt die Rede, sind die jeweiligen Stützstellen der Robotertrajektorie oberhalb der jeweiligen Start- und Zielpositionen gemeint.

Bei der Planung einer Trajektorie unter Verwendung einer Hindernisdarstellung im Konfigurationsraum, muss zunächst die Auflösung des Konfigurationsraums beziehungsweise die Diskretisierung der Gelenkwinkel festgelegt werden. Es gilt, je geringer die Auflösung des Konfigurationsraums, beziehungsweise je größer die Schrittweite der Gelenkstellungen bei der Bahnplanung ist, desto geringer ist die für die Planung benötigte Zeit. Darauf basierend und unter Berücksichtigung der maximalen Auflösung der PMD-Kamera empfiehlt es sich, eine Diskretisierung in 5°-Schritten vorzunehmen. Wie bereits erwähnt, findet die Berechnung des Konfigurationsraums während der Trajektorienplanung statt. Auf diese Weise entfällt die Berechnung des vollständigen Konfigurationsraums.

In der Literatur stehen verschiedene Verfahren zur Planung einer Trajektorie für Handhabungsroboter zur Verfügung. Nach [Latombe91] kann zwischen Roadmap-Verfahren und der Potentialfeldmethode unterschieden werden. Bei den Roadmap-Methoden wird die Roboterzelle beziehungsweise die Karte der Roboterzelle in verschiedene Abschnitte untergliedert. Diese werden untereinander verbunden, sollte ein kollisionsfreier Pfad zwischen ihnen vorhanden sein. Ist die Roadmap erstellt, kann mit verschiedenen Verfahren ein Pfad von der Startposition zur Zielposition gefunden werden. Hierzu gehören Zufallssuchverfahren und Graphensuchverfahren, wie zum Beispiel der Dijkstra- und A*-Algorithmus, sowie andere Breitensuchverfahren. Die Erstellung einer Roadmap ist, bei Verwendung der Bahnplanung im

Konfigurationsraum, in diesem Fall ohne Weiteres nicht möglich, da die Berechnung des Konfigurationsraums während der Bahnplanung nur für die währenddessen explorierten Positionen bzw. Gelenkstellungen durchgeführt werden soll. Aus diesem Grund wird auf die Erstellung einer expliziten Roadmap-Darstellung, wie diese zum Beispiel durch eine Zellzerlegung, einen Sichtbarkeitsgraph oder durch Voronoi-Diagramme erzeugt werden, verzichtet. Stattdessen wird die 5°-Diskretisierung des Konfigurationsraums zur Pfadsuche mittels Graphensuchverfahren übernommen. In Kapitel 5.5.2 wurden bereits verschiedene Graphensuchverfahren erläutert und der A*-Algorithmus schließlich für die Pfadsuche verwendet.

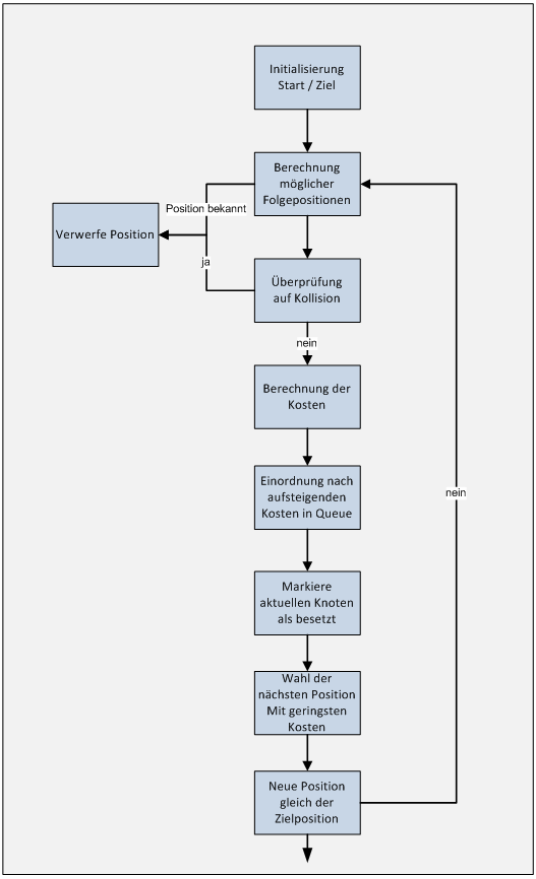


Abbildung 6.4.5: Flussdiagramm zur Bahnplanung mit dem A*-Algorithmus

Abbildung 6.4.5 zeigt das Flussdiagramm des auf Basis des A*-Algorithmus basierenden, implementierten Bahnplanungsverfahrens. Zunächst werden Start- und Zielposition initialisiert, indem aus den vorgegebenen kartesischen Positionsangaben des Start- und Zielpunkts mittels Rückwärtskinematik die entsprechenden Gelenkstellungen des Robotersystems berechnet werden. Ausgehend von den Gelenkstellungen der Startposition werden die möglichen Folgepositionen ermittelt. Jede Roboterposition besitzt im 6-dimensionalen Konfigurationsraum 12 Folgepositionen, die sich durch Addition der jeweiligen Gelenkwinkel mit dem der Schrittweite entsprechendem Winkel ergeben. Für die Folgepositionen im Konfigurationsraum wird anschließend, unter Einbeziehung der Octree-Umgebungsdarstellung ermittelt, ob diese kollisionsfrei sind und somit im freien Raum $\boldsymbol{C_{free}}$ des Konfigurationsraums liegen. Ist dies nicht der Fall, wird die entsprechende Position verworfen und für die weitere Berechnung nicht mehr berücksichtigt. Für die verbleibenden Positionen wird unter Zuhilfenahme der Kostenfunktion:

$$K = r \cdot K_{hin} + s \cdot K_{r\ddot{u}ck} \tag{6.16}$$

die entsprechenden Kosten der Roboterposition berechnet. Hierbei entspricht $K_{r\ddot{u}ck}$ der Länge des bisher zurückgelegten Weges und wird berechnet zu:

$$K_{r\ddot{u}ck} = \sum_{i=1}^{N} S \tag{6.17}$$

welches der Summe aller bisher zurückgelegten Positionsschritte multipliziert mit der Schrittweite S entspricht. K_{hin} ist die Differenz zwischen aktuellem und Zielgelenkwinkel:

$$K_{hin} = \|\vec{p}_{akt}(\theta_1, \dots, \theta_6) - \vec{p}_{Ziel}(\theta_1, \dots, \theta_6)\| \tag{6.18}$$

Die Parameter r und s dienen der Gewichtung der entsprechenden Elemente der Kostenfunktion. Diese Werte müssen so eingestellt werden, dass eine wegoptimale Lösung bei der Bahnplanung gefunden werden kann. Üblicherweise gilt:

$$r = 0{,}25 \quad \text{und} \quad s = 0{,}75$$

Nach der Berechnung der Kostenfunktion für die explorierten, kollisionsfreien Positionen wird die aktuelle Position als exploriert markiert und die neuen Positionen in eine Warteschlange (Queue) einsortiert. Diese wird nach absteigenden Kosten unter

Verwendung des Heapsort-Algorithmus sortiert. Im Folgenden wird die Position mit den geringsten Kosten gewählt. Entspricht diese der Zielposition wird das Planungsverfahren abgebrochen. Ist dies nicht der Fall wird entsprechend dem in Abbildung 6.4.5 dargestellten Flussdiagramm mit der Exploration der Folgepositionen fortgefahren.

Der implementierte Bahnplanungsalgorithmus führt, sollte eine Lösung der Bahnplanungsaufgabe existieren, in jedem Fall zu einem Ergebnis. Nachteilig ist jedoch, dass bei einer komplexen Hindernisumgebung für eine erfolgreiche Bahnplanung sehr viele Roboterpositionen exploriert werden müssen und dies zu einer exponentiell steigenden Berechnungszeit führt. Aufgrund dessen ist der bis hierher implementierte Trajektorienplaner für komplexe Hindernisumgebungen nicht geeignet. Eine schnellere Pfadplanung bei Verwendung eines Industrieroboters ist bei der Anwendung der Potentialfeldmethode [Hwang92] zu erwarten.

Die Potentialfeldmethode berechnet für die Bahnplanung ein künstliches Potentialfeld, welches eine Kraft auf einen punktförmigen Roboter in Richtung des Zielpunkts ausübt. Bei der Definition der Potentiale wird zwischen anziehenden Potentialen U_{attr} (attraktive Kräfte) und abstoßenden Potentialen U_{rep} (repulsive Kräfte) unterschieden. Das anziehende Potential wird durch die Position des Zielpunktes definiert und nimmt mit geringer werdendem Abstand zum Ziel ab. Das Potential U_{attr} wird dem zufolge als globales Potential definiert, da dieses zu jeder Zeit auf den Roboter einwirkt. Durch die Hindernisse innerhalb der Roboterzelle werden die abstoßenden Potentiale U_{rep} definiert. Diese führen nur eine Kraft in ihrer unmittelbaren Nähe auf den Roboter aus. Somit werden diese als lokale Potentiale definiert.

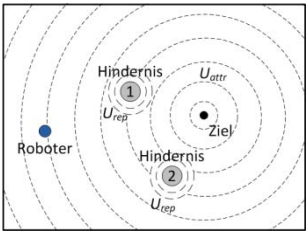


Abbildung 6.4.6: Definition der Potentialfelder innerhalb des Roboterarbeitsraums

Werden die Potentiale U_{attr} und U_{rep} überlagert, führt dies zu einem Potentialfeld U_{res}:

$$\vec{U}_{res}(\vec{p}(t),\vec{v}(t)) = \vec{U}_{attr}(\vec{p}(t),\vec{v}(t)) + \vec{U}_{rep}(\vec{p}(t),\vec{v}(t)) \tag{6.19}$$

Hierbei entspricht $\vec{p}(t)$ der Position und $\vec{v}(t)$ der Geschwindigkeit des Roboters zum Zeitpunkt t. Zur Berechnung der Potentialfunktionen müssen zwei Forderungen jederzeit erfüllt sein.

1. Das Potential $\vec{U}_{res}$ muss zu jeder Zeit eine Kraft $\vec{F}_{res}$ auf den Roboter ausführen.
2. Die Kraftwirkung erfolgt immer in Richtung der Zielposition.

Nach [Ge02] wird die Potentialfunktion wie folgt berechnet:

$$\vec{U}_{attr}(\vec{p}(t),\vec{v}(t)) = \alpha_p \|\vec{p}_{Ziel}(t) - \vec{p}(t)\|^m + \alpha_v \|\vec{v}_{Ziel}(t) - \vec{v}(t)\|^n \tag{6.20}$$

$\|\vec{p}_{Ziel}(t) - \vec{p}(t)\|$ entspricht der euklidischen Distanz zwischen aktueller Position und Zielposition und $\|\vec{v}_{Ziel}(t) - \vec{v}(t)\|$ der Größe der relativen Geschwindigkeit zwischen Ziel und Roboter zum Zeitpunkt t. α_p, α_v, n und m sind positive Skalare beziehungsweise Konstanten, welche zur Regelung des Bahnplanungsvorgangs dienen. Für größere Werte erfolgt eine schnellere Annäherung an die Position, beziehungsweise die Geschwindigkeit des Ziels. Aus der Potentialfunktion kann durch Bildung des negativen Gradienten die resultierende Kraft berechnet werden.

$$\vec{F}_{attr}(\vec{p},\vec{v}) = -\nabla\vec{U}_{attr}(\vec{p},\vec{v}) = -\nabla_{\mathrm{p}}\vec{U}_{attr}(\vec{p},\vec{v}) - \nabla_{\mathrm{v}}\vec{U}_{attr}(\vec{p},\vec{v}) \tag{6.21}$$

mit $\nabla_{\mathrm{p}}\vec{U}_{attr}(\vec{p},\vec{v}) = \frac{\vec{U}_{attr}(\vec{p},\vec{v})}{\partial p}$ und $\nabla_{\mathrm{v}}\vec{U}_{attr}(\vec{p},\vec{v}) = \frac{\vec{U}_{attr}(\vec{p},\vec{v})}{\partial v}$

Aus Gl.(6.21) folgt für die Kraft $\vec{F}_{attr}$:

$$\vec{F}_{attr}(\vec{p},\vec{v}) = \vec{F}_{attr1}(\vec{p}) + \vec{F}_{attr2}(\vec{v}) \tag{6.22}$$

$$\vec{F}_{attr1}(\vec{p}) = m\alpha_p \|\vec{p}_{Ziel}(t) - \vec{p}(t)\|^{m-1} \cdot \vec{n}_{RT} \tag{6.23}$$

$$\vec{F}_{attr2}(\vec{v}) = n\alpha_p \|\vec{v}_{Ziel}(t) - \vec{v}(t)\|^{n-1} \cdot \vec{n}_{vRT} \tag{6.24}$$

$\vec{n}_{RT}$ und $\vec{n}_{vRT}$ entsprechen den Einheitsvektoren in Richtung des Ziels beziehungsweise in Richtung der relativen Geschwindigkeit.

Abstoßende Potentiale befinden sich nur in der näheren Umgebung von Hindernissen. Diese müssen nach [Ge02] ebenfalls zwei Forderungen erfüllen.

1. Das abstoßende Potential wirkt nur eine Kraft auf den Roboter in unmittelbarer Umgebung der Hindernisse aus.
2. Die aus dem abstoßenden Potential entstehende Kraft ist von den Hindernissen weggerichtet.

Aus den Bedingungen kann eine Gleichung zur Berechnung des abstoßenden Potentials formuliert werden.

$$\vec{U}_{rep}(\vec{p},\vec{v}) = \begin{cases} 0, & \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m(\vec{v}_{Ro}) \geq \rho_0 \ \vee \ v_{Ro} \leq 0 \\ \mu\left(\frac{1}{\rho_s(\vec{p},\vec{p}_{obs}) - \rho_m(\vec{v}_{Rob})}\right) - \frac{1}{\rho_0}, & 0 < \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m(v_{Ro}) < \rho_0 \quad \vee \ v_{Ro} > 0 \\ nicht\ definiert, & v_{Ro} > 0 \ \wedge \ \rho_s(\vec{p},\vec{p}_{obs}) < \rho_m(\ v_{Ro}) \end{cases} \quad (6.25)$$

mit: $\vec{v}_{Ro} = (\vec{v}(t) - \vec{v}_{obs}(t))^t \cdot \vec{n}_{Ro}$

Hierbei entspricht $\vec{n}_{Ro}$ dem Einheitsvektor in Richtung des Hindernisses. Es gilt $v_{Ro}(t) < 0$. Dies entspricht einer Bewegung weg vom Hindernis. Die Länge des maximal benötigten Bremswegs ist ρ_m und $p_s(\vec{p},\vec{p}_{obs})$ ist der kürzeste Abstand zwischen Roboter und Hindernis. ρ_0 ist die maximale Entfernung, in welcher das vom Hindernis ausgehende Potential noch wirkt. Wenn ein Zusammenstoß zwischen Roboter und Hindernis nicht mehr vermieden werden kann, also $\rho_s(\vec{p},\vec{p}_{obs}) < \rho_m(v_{Ro})$ ist, ist das Potential nach [Ge02] nicht mehr definiert.

Über die Bildung des negativen Gradienten kann auch für das abstoßende Potential die auf den Roboter wirkende Kraft berechnet werden.

$$\vec{F}_{rep}(\vec{p},\vec{v}) = -\nabla\vec{U}_{rep}(\vec{p},\vec{v}) = -\nabla_{\mathrm{p}}\vec{U}_{rep}(\vec{p},\vec{v}) - \nabla_{\mathrm{v}}\vec{U}_{rep}(\vec{p},\vec{v}) \quad (6.26)$$

$$\vec{F}_{rep}(p,v) = \begin{cases} 0, & \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m(v_{Ro}) \geq \rho_0 \ \vee \ v_{Ro} \leq 0 \\ \vec{F}_{rep1} + \vec{F}_{rep2}, & 0 < \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m(v_{Ro}) < \rho_0 \quad \vee \ v_{Ro} > 0 \\ nicht\ definiert, & v_{Ro} > 0 \ \wedge \ \rho_s(\vec{p},\vec{p}_{obs}) < \rho_m(v_{Ro}) \end{cases} \quad (6.27)$$

mit:

$$\vec{F}_{rep1} = \frac{-\mu}{\rho_s(\vec{p},\vec{p}_{obs}) - \rho_m(\vec{v}_{Ro})}\left(1 + \frac{v_{Ro}}{a_{max}}\right)\vec{n}_{Ro}$$

und

$$\vec{F}_{rep2} = \frac{\mu v_{Ro} v_{Ro}}{\rho_s(\vec{p},\vec{p}_{obs})a_{\max}\left(\rho_s(\vec{p},\vec{p}_{obs}) - \rho_m(v_{Ro})\right)}\left(1 + \frac{v_{Ro}}{a_{max}}\right)\vec{n}_{Ro}$$

Die resultierende repulsive Kraft entspricht der Summe aller von den Hindernissen ausgehenden Kräfte:

$$\vec{F}_{rep} = \sum_{i=1}^{n_{Obs}} \vec{F}_{rep_i} \tag{6.28}$$

Bei der beschriebenen Herleitung wird von dem Fall eines dynamischen Online-Bahnplanungsverfahrens, bei dem sowohl dem Ziel als auch den Hindernissen eine Geschwindigkeit ungleich Null zugeordnet wird, ausgegangen. In unserem Fall weisen jedoch weder die Zielposition noch die Hindernisse eine Geschwindigkeit auf. Es kann vielmehr von einem reaktiven statischen Bahnplanungsverfahren ausgegangen werden. Folglich muss für die Bahnplanung die Geschwindigkeit der Hindernisse, des Ziels und auch des Roboters nicht weiter betrachtet werden. Somit können obige Gleichungen für das anziehende, vom Ziel ausgehende Potential folgendermaßen vereinfacht werden:

$$\vec{U}_{attr}(\vec{p}(t)) = \alpha_p \|\vec{p}_{Ziel}(t) - \vec{p}(t)\|^m \tag{6.29}$$

Hieraus ergibt sich die anziehende Kraft:

$$\vec{F}_{attr}(\vec{p}(t)) = m\alpha_p \|\vec{p}_{Ziel}(t) - \vec{p}(t)\|^{m-1} \cdot \vec{n}_{RT} \tag{6.30}$$

Für die aufgrund der Hindernisse resultierenden abstoßenden Potentiale folgt:

$$U_{rep}(p,v) = \begin{cases} 0, & \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m \geq \rho_0 \\ \mu\left(\frac{1}{\rho_s(\vec{p},\vec{p}_{obs}) - \rho_m}\right) - \frac{1}{\rho_0}, & 0 < \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m < \rho_0 \\ nicht\ definiert, & \rho_s(\vec{p},\vec{p}_{obs}) < \rho_m \end{cases} \quad (6.31)$$

Und unter Berücksichtigung von Gl.(6.30) und Gl.(6.31) gilt für die abstoßenden Kräfte:

$$F_{rep}(p,v) = \begin{cases} 0, & \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m \geq \rho_0 \\ \frac{-\mu}{\rho_s(\vec{p},\vec{p}_{obs}) - \rho_m} n_{Ro}, & 0 < \rho_s(\vec{p},\vec{p}_{obs}) - \rho_m < \rho_0 \\ nicht\ definiert, & \rho_s(\vec{p},\vec{p}_{obs}) < \rho_m \end{cases} \quad (6.32)$$

An der Zielposition weist die resultierende Potentialfunktion das globale Minimum auf. Die Pfadsuche erfolgt in Richtung minimalen Potentials. Beim Erreichen eines Minimums wird die Planung abgebrochen. Jedoch kann es im Einzelfall passieren, dass die Potentialfunktion mehrere Minima aufweist und der Roboter bei der Planung in einem lokalen Minimum verharrt. Die Pfadsuche führt in diesem Fall zu keiner Lösung.

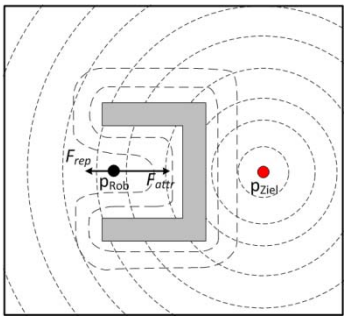


Abbildung 6.4.7: Entstehung eines lokalen Minimums

Ein Beispiel bezüglich dessen verdeutlicht Abbildung 6.4.7. Durch das U-förmige Hindernis entsteht ein lokales Minimum, in welchem die Bahnplanung endet. Sollte die Bahnplanung durch das Aufkommen eines lokalen Minimums beendet werden, obwohl dieses nicht der Zielposition entspricht, muss die Bahnplanung unter Verwendung einer anderen Methode zum Ziel geführt werden. In dem vorliegenden Fall, wird die Bahnplanung, wie es im Flussdiagramm in Abbildung 6.4.8 erkenntlich ist, durch

Anwendung des bereits beschriebenen Verfahrens des A*-Algorithmus, fortgeführt. Dies ermöglicht eine sichere Bahnplanung trotz des Vorliegens lokaler Minima.[5]

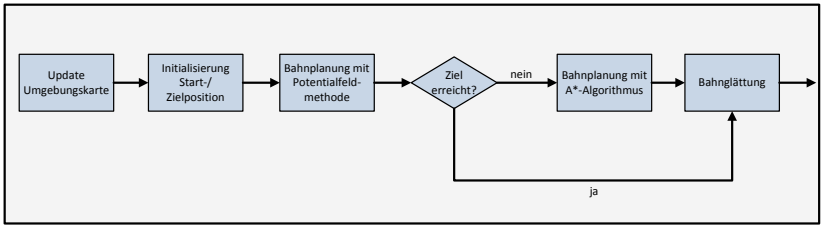


Abbildung 6.4.8: Flussdiagramm zum Bahnplanungsverfahren

Abbildung 6.4.9 visualisiert das Ergebnis des Bahnplanungsverfahrens. Die Positionen des TCP-Koordinatensystems, entsprechend der während der Bahnplanungsphase im Konfigurationsraum berechneten Pfadpunkte, sind durch die roten Punkte in Abbildung 6.4.9 a und b gekennzeichnet. Abbildung 6.4.9a zeigt alle berechneten Pfadpunkte, die durch die 5°-Diskretisierung des Konfigurationsraums, berechnet wurden. Durch das auf die Bahnplanung folgende Bahnglättungsverfahren, werden überflüssige Stützpunkte entfernt. Die reduzierten Stützpunkte sind in Abbildung 6.4.9b dargestellt. Die Reduzierung der Bahnpunkte führt zu einer geglätteten Bewegung des Roboters entlang der geplanten Trajektorie.

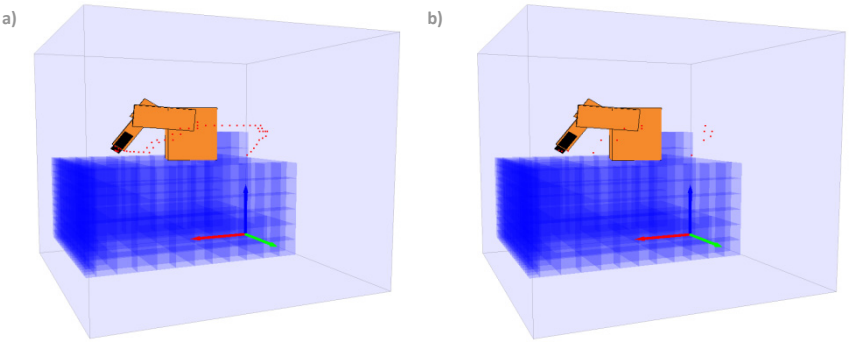


Abbildung 6.4.9: Bahnplanung a) ohne Bahnglättung b) mit Bahnglättung

[5] Basierend auf der Fusion des A*-Algorithmus und Potentialfeldmethode wird im folgenden ein neues robustes Bahnplanungsverfahren entwickelt.

Die Berechnungszeit der Bahnplanung liegt, abhängig von der Komplexität der Roboterumgebung im Bereich bis zu wenigen Sekunden. Die Echtzeitfähigkeit ist folglich nicht gegeben. Diese ist in der Aufgabenbeschreibung allerdings nicht gefordert gewesen.

Auf Basis der bisher entwickelten Algorithmen können Hinderniserkennung und Bahnplanung fusioniert werden. Da jedoch keine Unterscheidung zwischen Mensch und beliebigen Objekt getroffen werden kann, ist eine automatisierte Ausführung des Bahnplaners und das Fortfahren der Roboterfahrt auf einer alternativen Trajektorie nicht erlaubt. Der Roboter darf aus Sicherheitsgründen keine Bewegung ausführen, wenn die Möglichkeit besteht, dass sich eine Person im Arbeitsbereich des Roboters befindet. Aufgrund dessen wurde die Hinderniserkennung und die Bahnplanung mit PMD-Daten, wie im Flussdiagramm in Abbildung 6.4.10 dargestellt ist, fusioniert.

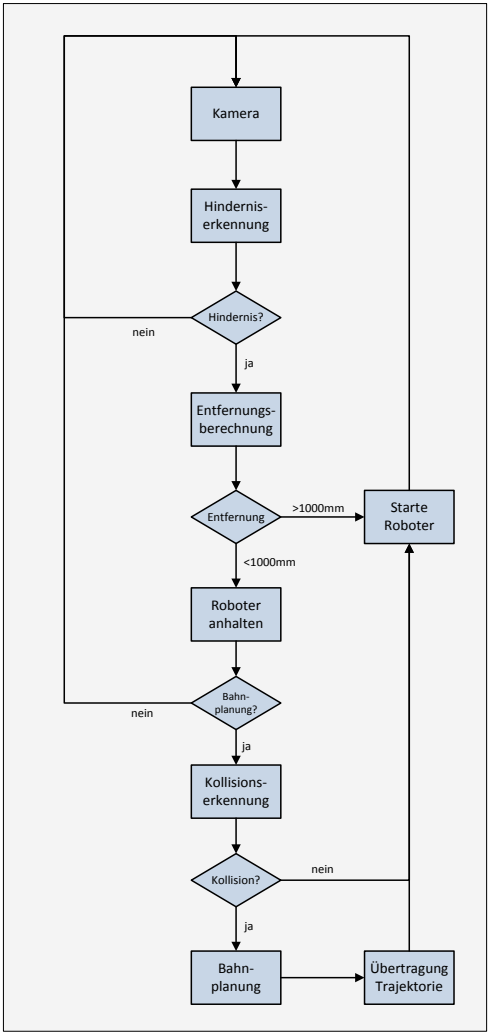


Abbildung 6.4.10: Flussdiagramm zur Fusion von Hinderniserkennung und Bahnplanung

Diese gestaltet sich wie folgt. Zunächst werden die Kameradaten zur Hinderniserkennung, nach Abschnitt 6.3, verwendet. Sollte kein neues Hindernis erkannt werden, wird die Hinderniserkennung mit den nächsten Kameradaten erneut ausgeführt. Sollte jedoch ein Hindernis detektiert werden, wird im Folgenden die Entfernung zwischen neuem Hindernis und dem Robotersystem ermittelt. Liegt diese

innerhalb einer vorher definierten Grenze wird der Roboter angehalten, bis das Hindernis entfernt wird, die Person sich nicht mehr im Arbeitsbereich des Roboters befindet oder von einem Anwender bestätigt wird, dass sich kein Mensch in der Roboterzelle befindet. Sollte die Entfernung größer als die Mindestentfernung sein, fährt die Hinderniserkennung mit der Auswertung des nächsten Kamerabildes fort. Sollte der Anwender bei vorhandenem Hindernis bestätigen, dass sich keine Person im Arbeitsbereich aufhält, wird eine Kollisionserkennung gestartet. Das neue Hindernis wird in die Karte aufgenommen und die Prüfung der aktuellen Trajektorie auf Kollision wird veranlasst. Sollte keine Kollision detektiert werden, fährt der Roboter mit der Ausführung seiner Trajektorie fort. Liegt jedoch eine Kollision vor, wird mit der Hilfe des Bahnplanungsverfahrens ein neuer Trajektorienverlauf ermittelt.

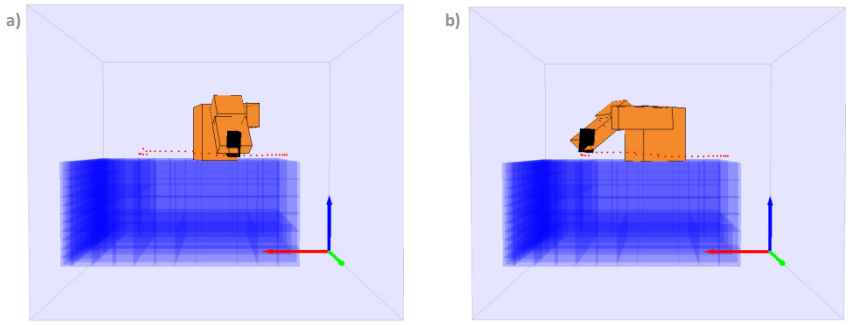


Abbildung 6.4.11: Trajektorienplanung ohne Hindernis

Zur Verifikation der Funktionsweise des Bahnplanungsverfahrens ist in Abbildung 6.4.11 der simulierte Trajektorienverlauf von Start- zu Zielposition dargestellt. Die Aufnahmen in Abbildung 6.4.12 zeigen das Abfahren der geplanten Trajektorie des realen Robotersystems.

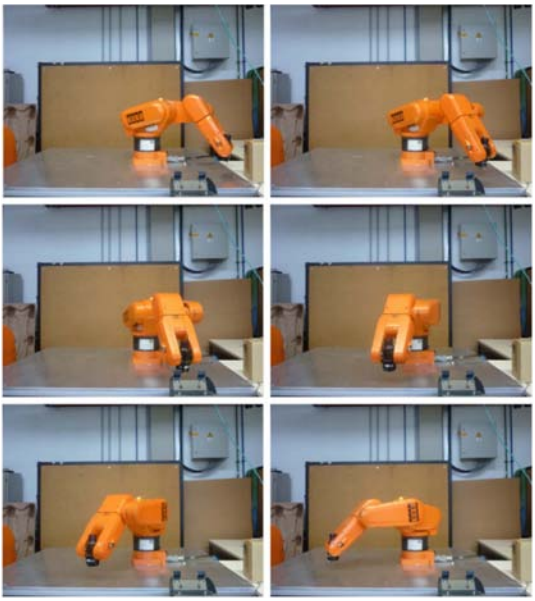


Abbildung 6.4.12: Der Roboter fährt die geplante Trajektorie ab

Wird jedoch von der Kamera eine Veränderung innerhalb der Roboterzelle detektiert, bleibt der Roboter stehen. Der Benutzer kann jedoch das erkannte Hindernis der Umgebungskarte hinzufügen und die Neuplanung der Trajektorie veranlassen.

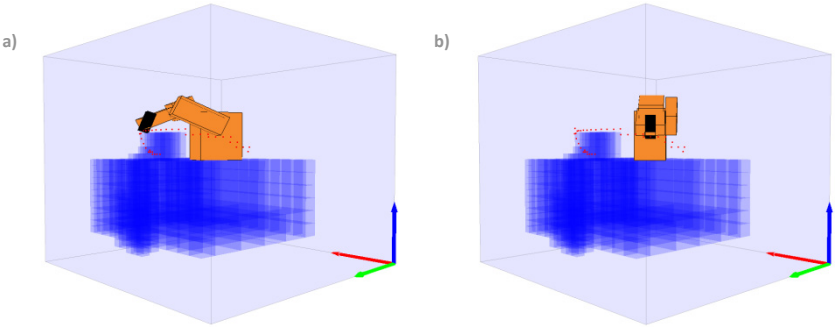


Abbildung 6.4.13: Neuplanung der Trajektorie nach erfolgreicher Hinderniserkennung

Abbildung 6.4.13b zeigt die neue Trajektorie als Ergebnis des Bahnplanungsverfahrens.

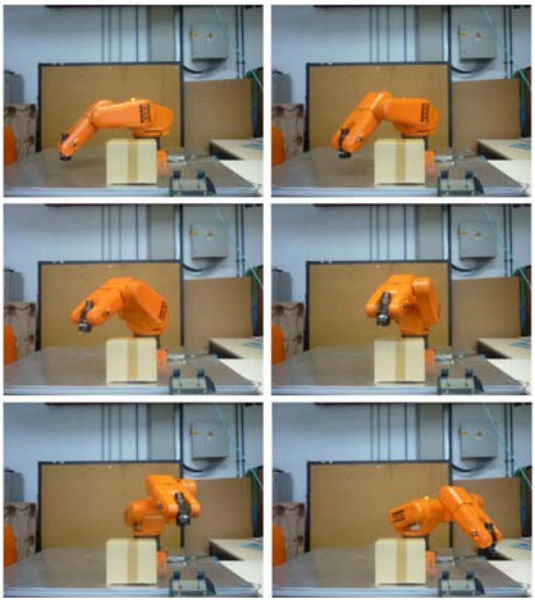


Abbildung 6.4.14: Neuplanung einer Trajektorie in Anwesenheit eines Hindernisses

Dementsprechend zeigen die Aufnahmen in Abbildung 6.4.14 das reale Robotersystem beim Abfahren der neu geplanten Trajektorie. Zu erkennen ist, dass der Roboter trotz des Hindernisses kollisionsfrei von Start- zur Zielposition gelangt.

6.5 Auswertung

In Kapitel 6.3 wurde ein geeigneter Aufbau für die Überwachung des Roboterarbeitsraums gefunden. Es wurde ein Kalibrierverfahren zur präzisen Registrierung des Kamerasystems innerhalb des Roboterkoordinatensystems entwickelt. Dies ermöglichte die Entwicklung von Algorithmen zur Überwachung des Roboterarbeitsraums. Die implementierten Methoden bieten eine echtzeitfähige Erkennung von fremden Objekten innerhalb des überwachten Bereichs der Roboterzelle. Auf diese Weise können Personen, die in den Roboterarbeitsraum eindringen, erkannt und deren Position bezüglich des Roboters bestimmt werden, so dass eine Gefährdung von in den Roboterarbeitsraum eindringenden Personen ausgeschlossen werden kann.

In Kapitel 6.4 wurde ein Bahnplanungsverfahren entwickelt. Dieses ermöglicht die Planung einer Trajektorie basierend auf einer Octree-Umgebungsdarstellung, welche die CCD-Umgebungsdarstellung des Roboterarbeitsraums mit den 3D Daten der verwendeten PMD-Kamera in einer gemeinsamen Umgebungsdarstellung fusioniert. Des Weiteren ermöglicht diese Darstellungsweise durch einfache Bereichsanfragen schnelle Kollisionsabfragen zwischen Roboter und Umgebung. Somit wird eine Beschleunigung des implementierten Bahnplanungsverfahrens erreicht. Da jedoch zurzeit keine Unterscheidung zwischen Mensch und Fremdobjekt getroffen wird, muss die Überprüfung auf Kollisionsfreiheit der programmierten Trajektorie, sowie die Neuplanung einer alternativen Trajektorie, unter Berücksichtigung der mit der PMD-Kamera aufgenommen Umgebung, durch den Benutzer veranlasst werden. Der Roboter fährt erst mit der Ausführung seiner Bewegung fort, sollte das neu detektierte Hindernis aus der Roboterzelle entfernt werden, beziehungsweise im Falle einer sich in der Zelle befindlichen Person, sich die Person aus der Zelle entfernt haben oder wenn der Robotersteuerung vom Benutzer bestätigt wird, dass sich keine Person in der Roboterzelle aufhält.

In diesem Kapitel wurde gezeigt, dass die PMD-Kamera erfolgreich zur Gewährleistung einer sicheren Steuerung von Industrierobotern verwendet werden kann. Es konnten Algorithmen entworfen werden, welche mit der PMD-Kamera die Überwachung des Roboterarbeitsraums ermöglichen. In die Roboterzelle eindringende Personen, sowie Veränderungen innerhalb dieser, welche zur Kollision führen können, werden erkannt und der Roboter bleibt stehen. Dies bietet eine gute Alternative zur Verwendung von Sicherheitszäunen, welche üblicherweise in der Industrie eingesetzt werden und das Betreten der Roboterzelle verhindern. Zudem liegen die Investitionskosten einer PMD-Kamera unter denen eines Sicherheitszauns. Dies führt somit zur Kostenminimierung bei der Installation einer Roboterzelle. Vorteil bei dem Gebrauch der PMD-Technik gegenüber anderen Ansätzen die Roboterzellen mit Lichtschranken oder Laserscannern gegenüber unbefugtem Betreten schützen, ist die Aufnahme einer 3D Umgebung des Roboterarbeitsraums. Somit ist ebenfalls die Erkennung von Veränderungen, die beispielsweise von einer Person in der Roboterzelle verursacht werden, möglich. Die Robotertrajektorie kann auf mögliche Kollisionen überprüft werden, bevor der Roboter mit dem Ausführen seiner Bewegung fortfährt. Bei der Verwendung von Lichtschranken und Laserscannern ist dies nicht möglich. Des Weiteren müssen Translationsbewegungen des Roboters zwischen zwei vorgegebenen Positionen bei Inbetriebnahme des Roboters nicht mehr manuell eingelernt werden. Stattdessen kann die Planung anhand der 3D-Daten der PMD-Kamera autonom durchgeführt werden.

Beim Vergleich der PMD-Kamera mit einem Stereovisionsystem, welches in [Ebert02] zur Überwachung der Roboterzelle verwendet wird, kann festgestellt werden, dass die PMD-Technik Vorteile hinsichtlich der Datenakquirierung bietet. Die 3D-Daten der PMD-Kamera ebenso wie deren Intensitätswerte werden direkt auf dem Sensor generiert. Eine softwareseitige Berechnung entfällt. Dies erspart erheblich an Rechenleistung und Rechenzeit. Somit ist die mittels PMD-Kamera erreichbare maximale Bildrate höher als bei Verwendung von Stereovisionsystemen. Weiterer Vorteil ist die homogene Punktwolke, welche prinzipbedingt von der PMD-Kamera erzeugt wird und eine einfache Hinderniserkennung durch Anwendung eines Subtraktionsverfahrens, wie es in Kapitel 6.3.2 erläutert wurde, ermöglicht. Stereovisionsystemen erlauben nur die Generierung von 3D-Daten für Punkte bei denen korrespondierende Pixel in den Bildern der einzelnen Kameramodule gefunden werden können. Dies führt jedoch zu inhomogenen Punktwolken. Folglich kann ein Subtraktionsverfahren mittels Referenzbild nicht eingesetzt werden, welches zum Gebrauch aufwendigerer Algorithmen führt.

Nachteilig jedoch ist der geringe Öffnungswinkel der PMD-Kamera von φ = 30°(h)x 40°(v). Aufgrund dessen kann zurzeit nur ein vergleichsweise geringer Bereich der Roboterzelle beobachtet werden. Eine Lösung wäre die Integration eines Multi-PMD-Kamerasystems zur Überwachung der Roboterzelle. Hier sei verwiesen auf das EU-Projekt „Safros“ [Nicolai10]. Ein Teilaspekt dieses Verbundprojekts ist der Aufbau eines Multi-PMD-Kamerasystems zur Überwachung eines Operationssaals in der roboter-assistierten Chirurgie [Berns10]. Neben dem größeren Überwachungsbereich eines solchen Systems, können durch eine optimierte Installation der Kameras innerhalb der Roboterzelle auch Überdeckungen verschiedener Objekte detektiert werden. Dies ermöglicht die Detektion von mehreren freien Bereichen, die vorher aufgrund von Überdeckungen irrtümlich als besetzt markiert wurden, welches somit zu einer erheblichen Zunahme der Robustheit des entwickelten Bahnplanungsverfahrens führt.

7 Zusammenfassung und Ausblick

Im Zuge dieser Arbeit wurden neue Methoden zur umfassenden Integration der PMD-Kamera in unterschiedlichen Bereichen der Robotik erfolgreich entwickelt und verifiziert.

Im Bereich der mobilen Robotik wurden Verfahren zur Registrierung der Kamera innerhalb des Roboterkoordinatensystems entworfen. Auf Basis der Registrierung können die PMD-Daten im Roboterkoordinatensystem verarbeitet werden. Ein Schwerpunkt wurde auf die Realisierung einer präzisen Selbstlokalisierung der mobilen Roboterplattform gelegt. Zu diesem Zweck wurden verschiedene Methoden implementiert, welche auf der Verwendung der PMD-Kamera beruhen. Zum einen wurden die Odometriedaten des Roboters mit Bewegungsvektoren, welche unter Verwendung des ICP-Verfahrens berechnet werden, fusioniert. Der ICP-Algorithmus ermöglicht die Bestimmung der Verschiebung zwischen aufeinanderfolgenden PMD-Aufnahmen und somit die Berechnung der relativen Position des Roboters. Zum anderen wurden Algorithmen, basierend auf der Methode der Haar-Like-Feature, zur Erkennung von künstlichen Landmarken implementiert. Diese absolute Selbstlokalisierung in Kombination mit den relativen Selbstlokalisiermethoden ermöglicht eine präzise Bestimmung von Position und Orientierung des mobilen Roboters. Beruhend auf den verschiedenen Methoden zur Selbstlokalisierung konnten neue echtzeitfähige Algorithmen zur Kollisionsvermeidung verwirklicht werden. Die Erzeugung homogener dreidimensionaler Punktwolken ermöglicht eine schnelle und robuste Detektion von Hindernissen. Durch Kameraregistrierung und Selbstlokalisierung ist es möglich diese in das Koordinatensystem der Umgebungskarte zu transformieren. Dies erlaubt den Abgleich der erkannten Hindernisse mit den in der Umgebungskarte vorhandenen Objekten sowie eine Kollisionserkennung zwischen dem Roboter und dem detektierten Hindernis. Zusätzlich zur Kameradatenverarbeitung wurde ein zweistufiges reaktives Bahnplanungsverfahren entwickelt, das die Neuplanung der Trajektorie nach erkannter Kollision ermöglicht.

Ferner wurde eine Methode zur schnellen Erkennung und Lokalisierung einer Anhängerdeichsel mit der am Heck angebrachten PMD-Kamera adaptiert. Basierend auf der exakten Berechnung der Lage der Deichsel wurde ein Positionsregler entwickelt, der ein autonomes Andocken an die Deichsel ermöglicht. Alle entworfenen Methoden wurden anhand des Versuchsträgers ausgiebig getestet und die Funktionalität und Leistungsfähigkeit der PMD-Kamera im Vergleich zu den aufgeführten alternativen optischen Messsensoren nachgewiesen. Verglichen mit 2D-Laserscaner bietet die PMD-Kamera die Möglichkeit der dreidimensionalen Vermessung von Objekten. Diese ermöglicht neben der Erkennung von Breite und Entfernung eines Hindernisses, ebenso die Bestimmung von deren Höhe, welches zum einen die Erkennung von negativen Hindernissen (Löcher, Treppen) als auch die Überprüfung von Hindernissen auf eventuelle Passierbarkeit erlaubt. Passierbare Hindernisse wie Tische unter denen der Roboter beispielsweise hindurchfahren kann oder Bordsteinkanten die von dem Roboter gegebenenfalls überwunden werden können. Ferner gestattet die PMD-Technik, aufgrund der zusätzlichen Generierung von Intensitätsbildern, die Verwendung klassischer 2D-Bildverarbeitungsalgorithmen. Auf diese Weise können Methoden zur Identifizierung von definierten Objekten implementiert werden. Gegenüber Stereovisionsystemen gewährleistet die PMD-Kamera eine homogene 3D-Datenaufnahme. Unabhängig von der aufzunehmenden Szene steht dem Anwender jederzeit eine konstante Anzahl von gleichverteilten Punkten zur Verfügung. Bei Stereovisionsystemen können nur Entfernungswerte für markante Punkte innerhalb der Bilder berechnet werden, welches eine Ermittlung von 3D-Punkten bei gleichfarbigen Flächen unmöglich macht. Sollten Wände oder Böden mit Stereovisionsystemen aufgenommen werden, kann nur eine geringe Anzahl an Entfernungswerten erzeugt werden. Des Weiteren benötigt die PMD-Kamera zur Erstellung der Entfernungswerte nur wenige Rechnerressourcen, da die 3D-Werte hardwareseitig ermittelt werden. Aufgrund der kompakten Bauweise konnte die Kamera zudem problemlos in das mobile Robotersystem integriert werden. Negativ jedoch fiel der Aspekt des eingeschränkten Messbereichs in Kombination mit dem zu Laserscannern vergleichsweise geringen Öffnungswinkel auf. Dies hat zur Folge, dass Hindernisse als auch die Deichsel gelegentlich spät erkannt werden, so dass eine Reaktion des Roboters zunächst ausbleibt und das autonome Ausweichen beziehungsweise Andocken eventuell nicht mehr möglich ist. Aus diesem Grund wurde ein softwareseitiger Notaus implementiert, der zum sofortigen Anhalten des Roboters führt, sollte ein Ausweichen oder Andocken aufgrund einer zu geringen Distanz nicht mehr möglich sein.

	IST	Innovation
Mobile Robotik	▪ Hoher Integrationsaufwand durch Fusion verschiedener Sensoren: • Laserscanner zur Fernfeldüberwachung • Stereovisonsysteme zur Nahfelderfassung ▪ Hohe Kosten ▪ Hoher Anschaffungspreis durch Summierung der Kosten der einzelnen Systeme ▪ Entwicklung von Software zur Umgebungsüberwachung ist aufwendig	▪ Verwendung der PMD-Kamera ermöglicht die Überwachung des Fern- und Nahfeldes ▪ Verzicht auf Fusion mit weiterer bildgebender Sensorik ▪ Kostenreduzierung ▪ Verringerung des Integrationsaufwands ▪ Ermöglicht die Verwendung einfacher Bildverarbeitungsalgorithmen zur Hinderniserkennung
Autonomes Andocken	▪ Andocken fahrerloser Transportsysteme an Anhänger wird zurzeit in der Industrie manuell durchgeführt ▪ Mit konventionellen Bildsensoren ist keine robuste dreidimensionale Erkennung der Anhängerdeichsel möglich	▪ Robuste PMD basierte Erkennung der Anhängerdeichsel ▪ PMD-Technik erlaubt eine 6 dimensionale Berechnung der Deichselposition bezüglich des Roboterkoordinatensystems ▪ Andocktrajektorienplanung auf Basis der berechneten Deichselpose
Handhabungs-robotik	▪ Roboter werden durch Sicherheitszäune vor dem Betreten unbefugter Personen geschützt ▪ Hohe Kosten durch Installation ▪ Unflexibler Ansatz ▪ Trajektorien werden größtenteils durch Teach-In Prozesse eingelernt ▪ Änderungen in der Roboterzelle erfordern eine neue Inbetriebnahme des Roboters (neuer Teach-In-Prozess, Erstellung einer neuen CAD-Umgebung) ▪ Ansätze zur Überwachung mit Laserscannern oder taktilen Böden ermöglichen lediglich eine zweidimensionale Überwachung • Roboter kann auf Personen durch Stoppen reagieren • Keine reaktive Bahnplanung möglich	▪ Verwendung der PMD-Kamera ermöglicht auf Basis der entwickelten Algorithmen eine echtzeitfähige Erkennung von Hindernissen innerhalb der Roboterzelle ▪ Roboter kann auf neue Hindernisse reagieren ▪ Erlaubt den Verzicht auf Sicherheitszäune ▪ Dreidimensionale Vermessung der erkannten Objekte erlaubt die reaktive Planung alternativer Trajektorien zum Ziel sollten Kollisionen erkannt werden ▪ Teach-In Prozesse, sowie eine eventuelle Generierung einer CAD-Umgebungskarte entfallen

Tabelle 3: Gegenüberstellung zwischen Industriestandard und der Innovation der vorliegenden Arbeit

Im Bereich der Handhabungsrobotik konnte die Überwachung des Roboterarbeits-raums, durch Anbringung einer PMD-Kamera oberhalb des Robotersystems,

ermöglicht werden. Ebenso wie in der mobilen Robotik wurden Algorithmen entwickelt, welche eine Registrierung der Kamera im Roboterkoordinatensystem ermöglichen. Auf diese aufbauend ließen sich Funktionen zur Erkennung von Fremdobjekten in der Roboterzelle realisieren. Die Positionen der erkannten Objekte bezüglich des Roboters können ermittelt werden. Dies ermöglicht eine echtzeitfähige Reaktion des Roboters auf auftretende Hindernisse, welches ein gefahrloses Betreten der Roboterzelle erlaubt. Der Aufbau von Sicherheitszäunen kann somit entfallen. Die Funktionalität wurde im Laufe dieser Arbeit um einen reaktiven Bahnplaner erweitert. Die erkannten Objekte können auf Wunsch der Hindernisumgebung hinzugefügt und bei einer Trajektorienneuplanung, sollte eine Kollision erkannt werden, berücksichtigt werden. Ebenso wie bei den Applikationen der mobilen Robotik kann hier im Vergleich zu den aufgeführten alternativen Messverfahren, die Generierung einer homogenen 3D-Punktwolke, für die Entwicklung von einfachen Objektregistrierungsverfahren, ausgenutzt werden. Die Überwachung der Roboterzelle bei gleichzeitiger reaktiver Bahnplanung kann bei Verwendung von 2D-Laserscannern nicht erfolgen. Stereovisionsysteme haben den Nachteil einer inhomogenen Verteilung der 3D-Messwert. Für Bildbereiche mit einer geringeren Dichte von 3D-Punkten kann keine verlässliche Aussage über das Vorhandensein von Hindernissen getroffen werden.

Ähnlich wie bei der mobilen Robotik besteht hier ebenfalls das Problem des eingeschränkten Sichtfelds der PMD-Kamera, welches nur die Beobachtung eines Teilbereichs der Roboterzelle erlaubt. Zur besseren Übersicht der Errungenschaften dieser Arbeit, stellt Tabelle 7 die in dieser Arbeit erzielten Innovationen und den heutigen Industriestandard gegenüber.

Zur Vermeidung der aufgeführten Probleme bei der Verwendung der PMD-Kameras müsste der Objektivöffnungswinkel vergrößert werden. Der Öffnungswinkel der Kamera muss an den Öffnungswinkel des Beleuchtungsmoduls angepasst sein, so dass sich ein Überlappungsbereich zwischen beiden ergibt. Nur unter diesen Bedingungen können Entfernungswerte hinreichend genau ermittelt werden. Ein Vergrößern des Sichtbereichs erfordert demnach das Vergrößern des Beleuchtungsbereichs. Ein größerer Beleuchtungsbereich wiederum benötigt eine höhere Lichtleistung, so dass die Qualität der Entfernungsmessung konstant bleibt. Für zukünftige Arbeiten besteht folglich Optimierungsbedarf zur Eliminierung dieser defizitären Aspekte.

Eine weitere Methode den Sichtbereich zu erhöhen ist die Verwendung eines Multi-PMD-Kamerasystems. Durch überlappende Bildbereiche der einzelnen PMD-Module könnte der effektive Beobachtungsbereich beliebig erhöht werden.

Hierdurch ließe sich nicht nur der beobachtbare Bereich vergrößern, sondern innerhalb der Handhabungsrobotik durch eine optimierte Anbringung der Kameras in der Roboterzelle, auch Überdeckungen zwischen unterschiedlichen Objekten vermeiden.

Ein zukünftig relevantes Thema ist die Ausfallsicherheit der verwendeten PMD-Kameras. Da es sich bei den implementierten Systemen um sicherheitsrelevante Applikationen handelt, müssen die verwendeten Systeme redundant aufgebaut werden. Dies bedeutet, dass für einen realen Einsatz der PMD-Kameras entweder die Kamerasysteme redundant aufgebaut werden müssen und so die erforderlichen Sicherheitsstandards erfüllt werden oder dass jede Kamera durch den Vergleich mit den Daten anderer Kameras auf ihre Funktionsfähigkeit während des Betriebs überprüft werden muss.

Wenngleich die PMD-Kameras noch Schwächen im Bezug auf den maximalen Messbereich aufweisen, welcher durch den Objektivöffnungswinkel in Kombination mit der lateralen Auflösung beschränkt wird, bieten die derzeitig aktuellen Modelle dennoch bedeutende Vorteile im Hinblick auf innovative Anwendungsmöglichkeiten in zahlreichen Gebieten der Robotik. Aufgrund der Generierung eines homogenen 3D-Datensatzes in Kombination mit einer hohen Bildwiederhohlfrequenz kann die PMD-Kamera, wie in dieser Arbeit bewiesen wurde, mit großem Erfolg zur sicheren Steuerung von mobilen Roboterplattformen als auch im Bereich der Handhabungs-robotik zur Absicherung der Roboterzellen gegenüber, in den Arbeitsbereich eindringenden Personen, eingesetzt werden. Bei erkannten Kollisionen des Handhabungsroboters mit seiner Umgebung konnten zudem Methoden zur Planung alternativer Trajektorien basierend auf den PMD-Daten implementiert werden.

Dank dieser Arbeit konnte die Eignung und Bedeutung von Time-of-Flight Kameras für gegenwärtige und zukünftige Forschungen im umfangreichen Feld der Robotik verifiziert und belegt werden.

Anhang

A. Bildverarbeitung

A.1 Nachbarschaftsoperatoren in der Bildverarbeitung

Die in dieser Arbeit verwendeten Filteralgorithmen zur Bildverarbeitung, wie zum Beispiel das Mean-Filter, Sobel-Filter und auch das Medianfilter gehören zur Gruppe der Nachbarschaftsoperatoren. Diese dienen der Kombination von Pixeln in einer kleinen Umgebung. Hierdurch lassen sich weitere Informationen innerhalb eines Bildes ableiten. Nachbarschaftsoperatoren sind somit ein zentrales Element der Bildverarbeitung und können zum Beispiel zur Detektion von Kanten, zur Verringerung des Rauschens in Bildern durch Glättung oder zur Texturanalyse dienen. Da die Anwendung von Nachbarschaftsoperatoren auf Bilder generell verknüpft ist mit dem Verlust von Informationen, werden diese üblicherweise in der Literatur als Filter bezeichnet.

Nachbarschaftsoperatoren benutzen eine Maske der Größe $(\boldsymbol{2R+1}) \times (\boldsymbol{2R+1})$ mit einer ungeraden Anzahl an Punkten. Dies gewährleistet, dass das resultierende Bild nicht verzehrt wird, sondern jedem Punkt des Ausgangsbildes ein Punkt im Ergebnisbild zugeordnet werden kann. Die Verknüpfung vom Grauwerten in einer Nachbarschaft erfolgt über die diskrete Faltung mit der Maske $\boldsymbol{M}$. Dies geschieht durch Multiplikation der Bildpunkte im Bereich der Filtermaske mit den entsprechenden Gewichtungsfaktoren mit anschließender Addition aller berechneten Werte. Das Ergebnis der Faltung wird an der Stelle der zentralen Position eingetragen.

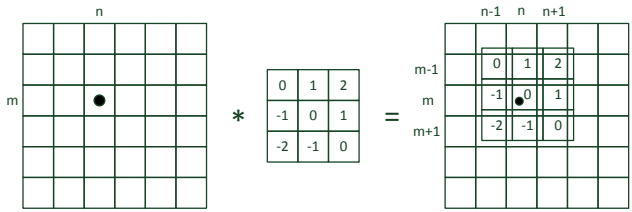


Abbildung A.1: Skizze zur diskreten Faltung mit einer 3x3 Filtermaske

Zur Illustration der Faltung zeigt Abbildung A.1 die diskrete Faltung mit einer (3×3) Filtermaske. Diese wird für jedes Pixel durchgeführt.

A.2 Mean-Filter

Das Mean-Filter gehört zur Gruppe der Nachbarschaftsoperatoren und ist einer der einfachsten Glättungsfilter. Dieser bewirkt die Glättung eines Bildes durch lokale Mittelung des jeweiligen Pixels. Aufgrund dessen weißt der Mean-Filter einen Tiefpasscharakter auf. Durch Anwendung der Filtermaske H

$$H = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

wird das arithmetische Mittel der Pixelwerte in einer (3×3) Region des jeweiligen Bildpunkts ermittelt.

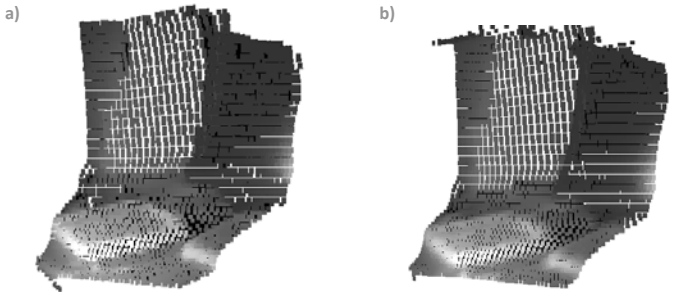


Abbildung A.2: Anwendung des Mean-Filters a) original PMD-Aufnahme b) PMD-Aufnahme nach Anwendung des Mean-Filters

Abbildung A.2 zeigt die Anwendung eines Mean-Filters auf eine PMD-Aufnahme. Zu erkennen ist eine Glättung des verrauschten Bilds. Nachteilig bei der Verwendung des Filters ist jedoch die Verwischung der Kanten innerhalb der Aufnahme. Dies führt dazu, das Kanten innerhalb des PMD-Bilds nicht genau bestimmbar sind und kleinere Strukturen nicht mehr erkennbar sind.

A.3 Rangordnungsfilter

Eine weitere Gruppe von Nachbarschaftsoperatoren sind die Rangordnungsfilter. Während bei Verwendung einer Filtermaske die Berechnung über Gewichtung und Addition der Pixelwerte innerhalb einer definierten Region um das entsprechende

Pixel erfolgt, erfolgt die Berechnung über Vergleich und Selektion der Pixelwerte in der Nachbarschaftsumgebung.

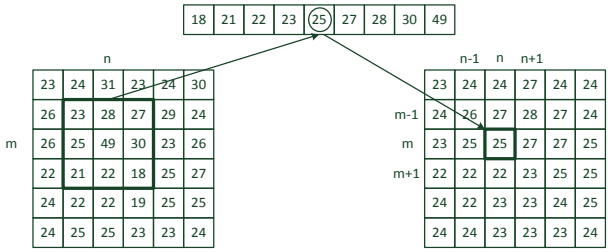


Abbildung A.3: Funktionsweise eines Rangordnungsfilters anhand eines 3x3 Medianfilters

Abbildung A.3 zeigt das Vorgehen bei Verwendung eines Rangordnungsfilters am Beispiel eines (3×3) Median-Filters. Hierbei werden die Pixel in einer (3×3) großen Umgebung des Pixels in aufsteigender Reihenfolge sortiert. Auf diese Weise können Minimum, Maximum oder auch der Mittelwert durch Wahl des ersten, letzten bzw. mittleren Wertes bestimmt werden. Die Wahl des mittleren Pixels entspricht dem Medianfilter. Der Vorteil der Verwendung des Median-Filters gegenüber dem Mean-Filter ist, dass der resultierende Pixelwert unabhängiger gegenüber starkem Rauschen ist und Kanten innerhalb des Bildes nicht in dem Maße verwischt werden.

A.4 Canny-Filter

Der Canny-Filter [Jähne05] ist ein robuster Filter zur Kantendetektion und vereint eine gute Kantenerkennung mit einer präzisen Lokalisierung. Der verwendete Algorithmus ist ein mehrstufiges Verfahren. Damit durch Pixelrauschen bedingte Fehlinterpretationen während der Berechnung minimiert werden, wird das Ausgangsbild zunächst durch Filterung zum Beispiel mit der Filtermaske N:

$$\boldsymbol{N} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{2} & \mathbf{4} & \mathbf{2} \\ \mathbf{1} & \mathbf{2} & \mathbf{1} \end{bmatrix}$$

in der Art geglättet, dass annähernd eine Normalverteilung entsteht.

Anschließend werden durch Anwendung des Sobeloperators [Jähne05] die Gradienten in x- und y-Richtung bestimmt. Hierzu wird das geglättete Bild mit den Faltungsmasken $\boldsymbol{S_x}$ und $\boldsymbol{S_y}$ gefaltet.

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Aus den berechneten Gradientenbildern G_x und G_y kann die resultierende Gradientenamplitude G ermittelt werden:

$$|G(n,m)| = \sqrt{G_x(n,m)^2 + G_y(n,m)^2} \tag{A.1}$$

Zur Verringerung der Rechenkomplexität wir diese durch Berechnung der Manhattan Distanz genähert.

$$|G(n,m)| = |G_x(n,m)| + |G_y(n,m)| \tag{A.2}$$

Neben der Gradientenamplitude kann die Orientierung der Kante in Richtung des aufsteigenden Gradienten ermittelt werden.

$$\theta(n,m) = \begin{cases} arctan\left(\frac{G_y(n,m)}{G_x(n,m)}\right) & , G_y(n,m) \neq 0 \\ 0° & , G_x(n,m) = 0 \wedge G_y(n,m) = 0 \\ 90° & , G_x(n,m) = 0 \wedge G_y(n,m) \neq 0 \end{cases} \tag{A.3}$$

Da jedes Pixel insgesamt 8 Nachbarpixel aufweist, kann Gl.(A.3) lediglich vier Gradientenrichtungen aufweisen. Diese betragen $0°$, $45°$, $90°$ und $135°$. Die berechneten Werte müssen folglich entsprechend der möglichen Winkel gerundet werden.

Sind die Gradientenamplituden sowie die Gradientenrichtungen bestimmt, kann das Maximum entlang der Gradientenrichtung ermittelt werden. Alle Nichtmaxima werden auf null gesetzt. Übrig bleiben die relevanten Kantenpunkte.

B. Kommunikationsschnittstelle des KUKA KR3

Die Programmierung des Roboters erfolgt mittels KRL-KUKA Robot Language auf dem Robotersystem. Die KRL ist eine speziell von KUKA entwickelte Roboterprogrammiersprache, welche auf einer Pascal ähnlicher Syntax basiert.

Das implementierte Roboterprogramm kann in drei Teile untergliedert werden. Der erste Teil entspricht dem Hauptprogramm. In diesem werden die benötigten Variablen, die Interruptsteuerung definiert und die Bewegungssteuerung des Roboters übernommen. Die Interrupts dienen der Ausführung der beiden anderen Teile. Ein weiterer Teil übernimmt die Roboter-PC Kommunikation. Dieser wird durch eine zeitabhängige Interruptsteuerung ausgeführt. Alle 50ms wird ein Interrupt durchgeführt und überprüft ob eine Positionsabfrage oder Datenübertragung vom PC erfolgt. Liegt eine Positionsanfrage vor, werden die Roboterdaten zum PC übertragen, wo diese zur Berechnung des Robotermodells verwendet werden. Werden neue Positionsdaten übertragen, werden diese ausgewertet und in ein Array abgespeichert, welches im Folgenden von dem Hauptprogramm abgearbeitet wird. Der letzte Programmabschnitt übernimmt die Steuerung des Interrupts zum Abbruch der Bewegungsdurchführung, sollte von der Hinderniserkennung ein fremdes Objekt innerhalb der Roboterzelle erkannt werden.

B.1 Ethernet-XML-Protokoll der Kommunikation zwischen Roboter und PC

Die Kommunikation zwischen Robotersystem und bildverarbeiteten PC erfolgt über ein XML-Protokoll. Hierzu wurde auf dem Roboter das zusätzlich erhältliche Softwaretool KRL-XML-Ethernet-Interface des Roboterherstellers verwendet. Dieses stellt eine Ethernet-Schnittstelle zum Austausch von XML-Daten zur Verfügung und dient gleichzeitig als Parser zum Erstellen und Auslesen von XML-Protokollen. PC-seitig wird zur Erzeugung und Auswertung von XML-Protokollen und zur Realisierung der Ethernet-Schnittstelle die auch für die grafischen Benutzeroberflächen genutzte QT-Bibliothek, verwendet.

XML steht für Extensible Markup Language und bedeutet ins Deutsche übersetzt so viel wie erweiterte Auszeichnungssprache. Die Sprache XML wurde vom World Wide Web Konsortium (W3C) entwickelt und soll zum Austausch strukturierter Daten zwischen beliebigen Plattformen dienen. Ein Vorteil dieser Sprache ist, dass diese in ASCII definiert ist und somit vom Menschen leicht lesbar ist. XML-Dokumente geben geschachtelte Baumstrukturen wieder. Sie bestehen aus verschiedenen Tags, welche beliebig ineinander geschachtelt werden können. Jedes Element beginnt mit einem

Start-Tag wie zum Beispiel <Robot> und endet mit einem End-Tag </Robot> oder />. Ein Tag kann Attribute enthalten. Attribute bestehen aus einem Attributnamen einem Gleichheitszeichen gefolgt von einem beliebigen Wert in Anführungszeichen. Ebenso kann ein Tag einfachen Text und weitere Tags enthalten.

Informationen, welche im Zuge der Kommunikation und Datenverarbeitung zwischen Roboter und datenverarbeitenden Computer ausgetauscht werden müssen, sind die Angaben der aktuellen Roboterposition. Hier werden sowohl die Winkel Stellungen der Roboterglieder als auch die Roboterpose des TCPs an den Computer gesendet. Das Senden beider Datensätze erspart im Zuge der Robotermodellierung die Berechnung der inversen beziehungsweise der Vorwärtskinematik. Diese Daten werden alle 50ms vom Robotersystem aktualisiert und nach Aufforderung des Computers mittels Ethernet versendet. Abbildung B.1 zeigt das dazugehörige XML-Protokoll.

```
<Robot>
   <Data>
      <ActPos  X="" Y="" Z="" A="" B="" C=""
               A1="" A2="" A3="" A4="" A5="" A6=""/>
   </Data>
</Robot>
```

Abbildung B.1: XML-Protokoll zur Roboter-PC Kommunikation

```
<Elements>
  <Element Tag="Robot" Type="STRUCTTAG" Stacksize="5" />
  <Element Tag="Robot.Mode" Type="INTEGER" Stacksize="5" />
  <Element Tag="Robot.Drive" Type="INTEGER" Stacksize="5" />
  <Element       Tag="Robot.maxNumPosition"       Type="INTEGER"
                 Stacksize="5" />
  <Element Tag="Robot.numPosition" Type="INTEGER" Stacksize="5"
                 />
  <Element Tag="Robot.Position.A1" Type="REAL" Stacksize="5" />
  <Element Tag="Robot.Position.A2" Type="REAL" Stacksize="5" />
  <Element Tag="Robot.Position.A3" Type="REAL" Stacksize="5" />
  <Element Tag="Robot.Position.A4" Type="REAL" Stacksize="5" />
  <Element Tag="Robot.Position.A5" Type="REAL" Stacksize="5" />
  <Element Tag="Robot.Position.A6" Type="REAL" Stacksize="5" />
</Elements>
```

Abbildung B.2: XML-Protokoll zur PC-Roboterkommunikation

Die Daten, welche von dem PC an das Robotersystem gesendet werden, beinhalten Informationen zu dem Betriebsmodus, dem Bewegungsstatus des Roboters, sowie der Anzahl, Index und Wert der zu übermittelnden Gelenkstellungen. Abhängig von dem

Wert „Mode“ kann der Roboter in verschiedene Modi versetzt werden. Diese sind der Normalbetrieb, hierbei wird die Kamera zur Hindernisdetektion verwendet, Kalibriermodus zur Kameraregistrierung und der Übertragungsmodus bei dem der Roboter durch Angabe der Gelenkstellungen die Informationen über die neugeplante Trajektorie erhält. „Drive“ gibt an ob der Roboter sich bewegen soll oder nicht. „maxNum Position“ gibt die maximale Anzahl der zu übertragenden Positionen an und „numPosition“ entspricht dem Index der aktuell zu übertragenden Position. Diese werden auf der Robotersteuerung in ein Array abgespeichert, welches der Roboter während seiner Bewegung abarbeitet. Abbildung B.2 zeigt die entsprechende Protokoll-Definition der auszulesenden Daten, so dass der Roboterparser diese verarbeiten kann.

Abbildungsverzeichnis

Literaturverzeichnis

[Ayra02] S. Arya, T. Malamatos, D. M. Mount, *Space-Efficient Approximate Voronoi Diagrams*, Proc. 34th ACM Symp. on Theory of Computing (STOC 2002), pp. 721–730

[Bentley75] Jon Lois Bentley, *Multidimensional binary search trees used for associative searching*, Communications of the ACM, Volume 18 Isuue9, Sept. 1975

[Berns09a] J. Bernshausen, J. Wahrburg, H. Roth: *Obstacle recognition in the workspace of an industrial robot by a 3D-PMD-Camera*, Proceedings European Control Conference 2009, Budapest, Hungary, 2009, pp. 2787-2791

[Berns09b] J. Bernshausen, J. Wahrburg, H. Roth: *Application of a novel 3D PMD-camera to monitor the workspace of an industrial robot*, Robot Vision: New Research, Nova Science Publishers Inc, 2009.

[Berns10] J. Bernshausen, J. Wahrburg, P. Nicolai, H. Mönnich, *PMD-Kameratechnik als Teil eines Sicherheitskonzepts für roboterunterstützte Chirurgie*, curac2010@MEDICA, Chirurgische Interventionen: vom Neanderthaler zur Roboterassistenz, Tagungsband zur 9. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V. (CURAC) vom 18. bis 19. November 2010, Düsseldorf, Tönning, Germany, 2010, pp. 199-202

[Berthold81] Berthold K. P. Horn and Brian G. Schunck, *Determining optical flow*, Artificial Intelligence, vol. 17, no. 1-3, pp. 185--203, 1981

[Besel92] Paul J. Besl, Neil D. McKay, *A Method for Registration of 3-D Shapes*, IEEE Transactions on Pattern and Machine Intelligence, Vol.14 Feb. 1992

[Bogh03] F. Boughorbel, Y.Zhang, Sangkyu Kang, Umayal Chidambaram, Besma Abidi, Andreas Koschan and Mongi Abidilaser, *ranging and video imaging for bin picking*, Assembly Automation, Vol. 23, pp.53-59, March 2003

[Brauer07] Brauer, W., Berns, K., Luksch, T., Hillenbrand, Carsten, Berns, Karsten, *Modulare Sicherheits- und Sensorsysteme für autonome mobile Roboter realisiert im Forschungsfahrzeug Marvin*, Autonome Mobile Systeme 2007, Informatik aktuell 2007, Springer Berlin Heidelberg, S. 133-137

[Burger06] Wilhelm Burger, Mark James Burge, *Digitale Bildverarbeitung,* Eine Einführung mit Java und ImageJ, 2. überarbeitete Auflage, Springer Verlag 2006

[Canny86] Canny, J., *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[Chet02] D. Chetverikov, D. Svirko, P. Krsek, The *Trimmed Iterative Closest Point Algorithm*, ICPR '02 Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3

[Cole06] David M. Cole and Paul M. Newman, *Using Laser Range Data for 3D SLAM in Outdoor Environments*, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida, May 2006

[Deuflhard00] Peter Deuflhard, Andreas Hohmann: *Numerische Mathematik I*. Walter deGruyter, Berlin 2000

[Dijkstra59] E.W. Dijkstra, *A Note on Two Problems in Connexion with Graph*, Numerische Mathematik 1, pp.269-271, 1959

[Ebert02] Dirk Ebert, Dominik Heinrich, Safe *Human-Robot-Cooperation: Image-based collision detection for Industrial Robots*, IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, September 30th-October 5th, 2002

[Evgeni97] Evgeni A. Numinski, *Mathematical Programming*, *Separating plane algorithms for convex optimization*, Volume 76, Number 3, pp.373-397, Springer Verlag 1997

[Frey08] Jochen Frey, *Entwurf und Untersuchung von hochauflösenden 3D-Bildsensoren in CMOS-Technologie*, Reihe Elektrotechnik, Band 3, Sierke Verlag 2008

[Fischler81] Martin A. Fischler, Robert C. Boyles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, Communications of the ACM, Volume 24 Issue 6, June 1981

[Freund99] Yoav Freund, Robert E. Schapire, *A Short Introduction to Boosting*, Japanese Society for Artificial Intelligence, Vol. 14, No. 5. (1999), pp. 771-780.

[Fuchs09] Stefan Fuchs, Sami Haddadin, Maik Keller, Sven Parusel, Andreas Kolb and Michael Suppa, *Cooperative Bin-Picking with Time-of-Flight Camera and Impedance Controlled DLR Lightweight Robot III*, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2010), Taipeh, Taiwan, 2010

[Garcia04] M.A.Garcia, A.Solanas, *3D Simultaneous Loacalisation and Modeling from Stereo Vision*, Proccedings of the 2004 IEEE International Conference of Robotics and Automation, New Orleans, LA, April 204

[GE02] S.S. GE, Y.J. CUI, *Dynamic Motion Planning for Mobile Robots Using Potential Field Method*, Autonomous Robots 13, pp. 207–222, Kluwer Academic Publishers 2002

[Ghobadi08] S. Ghobadi, O. Loepprich, F. Ahmadov, J. Bernshausen, K. Hartmann, O. Loffeld, *Real Time Hand Based Control Using 2D/3D Images*, Advances in visual computing: 4th international Symposium, ISVC 2008 Las Vegas, NV, USA, 2008

[Goetze94] Bernhard Goetze, *Methode der kleinsten Quadrate zur Approximation von Punktmengen durch Ebenen,* Gesellschaft zur Förderung angewandter Informatik, Berlin 1994

[Görz03] G. Görz, C.-R. Rollinger, J. Schneeberger, *Handbuch der Künstlichen Intelligenz*, 4. Auflage, Oldenbourg Wissenschaftsverlag, München 2003

[Haken08] Karl-Ludwig Haken, *Grundlagen der Fahrzeugtechnik*, Carl Hanser Verlag München, 2008

[Hartley03] Richard Hartley and Andrew Zisserman, Multiple View Geometry in computer vision. Cambridge University Press 2003

[Hart68] P. E. Hart, N. J. Nilsson, B. Raphael, *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions on Systems Science and Cybernetics SSC4 (2), pp. 100–107, 1968

[Hwang92] Y. Hwang, N.Ahuja, A Potential Field Approach to Path Planning, IEEE Transactions on Robotics and Automation, Vol. 8, 23-32, February 1992

[IFMO3D100] Datenblatt zur O3D-100, ifm, Essen 2007

[IFMO3D200] Datenblatt zur O3D-200, ifm, Essen 2009

[Jähne05] Bernd Jähne, *Digitale Bildverarbeitung*, 6. Überarbeitete und erweiterte Auflage, Springer Verlag Heidelberg Berlin 2005

[Joochim] Joochim, C., Roth, H., *Development of a 3D mapping using 2D/3D sensors for mobile robot locomotion*, IEEE International Conference on Technologies for Practical Robot Applications, TePRA 2008, pp. 100-105, Woburn, MA Nov. 2008

[Kraft07] Holger Kraft, *Untersuchung und Entwicklung integrierbarer Photomischdetektor (PMD)-Konzepte auf Halbleiterbasis zur Realisierung hochauflösender 3D-Messsysteme*, Dissertation, Universität Siegen, 2007

[Kalman60] Kalman, R.E., *A New Approach to Linear Filtering and Prediction Problems*, Transaction of the ASME, Journal of Basic Engineering, pp. 35-45, 1960

[Kraus97] Karl Kraus, Peter Waldhäusl: *Photogrammetrie,* Band 1. Ferd. Dümmler, Bonn 1997

[Krey07] U. Krey, A. Owen: *Basic Theoretical Physics - A Concise Overview*. Kapitel 7, Springer-Verlag, Berlin 2007

[Kuipers99] Kuipers, Jack B., *Quaternions and rotation Sequences: a Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Princeton University Press 1999

[Kyrill08] K. Safronov, I. Tchouchenkov, H. Wörn, *Hierachical Iterative Object Recognition Method for a PMD-Sensor supplied Bin-Picking System*, 8th Asia-Pacific Conference on Control and Measurement (APCCM 2008), Harbin, China

[Lamikiz08] A. Lamikiz, L. N. López de Lacalle, O. Ocerin, D. Díez and E. Maidagan , *The Denavit and Hartenberg approach applied to evaluate the consequences in the tool tip position of geometrical errors in five-axis milling centres*, The International Journal of Advanced Manufacturing Technology, 2008, Volume 37, Numbers 1-2, Pages 122-139

[Latombe91] Jean-Claude Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991

[Laumond98] Jean-Paul Laumond, *Robot Motion Planning and Control*, Springer Verlag, 1998

[LaValle06] Steven M. LaValle, *Planning Algorithms*, Cambridge University Press, New York 2006

[Lozano83] Thomas Lozano-Perez, *Spatial Planning: A Configuration Space*, IEEE Transactions on Computers, Vol. 32, 1983

[Luan01] Xuming Luan, *Experimental Investigation of Photonic Mixer Device and Development of TOF 3D Ranging Systems Based on PMD Technology*, Siegen 2001

[Maybeck79] Peter S. Maybeck, *Stochastic models, estimation, and control*, Volume 1, Academic Press, New York 1979

[MESA10] Datenblatt zur Swissranger SR-4000, Datasheet Revision 4.8, MESA, Zürich 2010

[Murray97] Don Murray, Cullen Jennings, *Stereo Vision based Mapping and Navigation for Mobile Robot*, Proccedings of the IEEE International Conference on Robotics and Automation, Albuqurque, New Mexico, April 1997

[Nehmzow02] U. Nehmzow, *Mobile Robotik – Eine praktische Einführung*, Springer Verlag Berlin Heidelberg 2002

[Nicolai10] P. Nicolai, H. Mönnich, J. Raczkowsky, H. Wörn, J. Bernshausen: *Überwachung eines Operationssaals für die kooperative robotergestützte Chirurgie mittels neuartiger Tiefenbildkameras*, curac2010@MEDICA, Chirurgische Interventionen: vom Neanderthaler zur Roboterassistenz, Tagungsband zur 9. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V. (CURAC) vom 18. bis 19. November 2010, Düsseldorf, Tönning, Germany, 2010

[PMDS3_09] Datenblatt zur [PMDVision]S3, Datasheet V.No. 20090601, PMDTechnologies GmbH, Siegen 2009

[PMD3kS_05] Datenblatt zur [PMDVision]3kS, PMDTechnologies GmbH, Siegen 2005

[Prusak07] A. Prusak, J.Bernshausen, H. Roth, J. Wahrburg, C. Hille, H.-H. Götting, T. Neugebauer, *Applications of automated guided vehicle (AGV) and industry robots with PMD-camera*, 13th IASTED International Conference on Robotics and Applications (RA 2007), Würzburg/ Germany, 29-31 Aug 2007.

[Rapp07] Holger Rapp, *Investigation of correlating TOF-camera systems* – ToF for dummies: why you are getting bad *data from your camera and what to do about it*, VDM, Müller, Saarbrücken, 2008

[Ringbeck01] Thorsten Ringbeck, *Untersuchung opto-elektrischer Phasenregelkreise auf Basis von Photomischdetektoren hinsichtlich deren Anwendungs-*

potentials für die Photonik, ZESS-Forschungsberichte, Shaker Verlag 2002

[Ringbeck07] Thorsten Ringbeck, *A 3D Time of Flight Camera for Object Detection*, Proccedings in Optical 3D Measurement Techniques, ETH Zürrich, 2007

[Schwarte98] R. Schwarte, H. Heinol, B. Buxbaum, Z. Xu, T. Ringbeck, Z. Zhang, W. Tai, K. Hartmann, W. Kleuver, X. Luan, *Neuartige 3D-Visionsysteme auf der Basis Layout-optimierter PMD-Strukturen*, In: tm – Technisches Messen. Nr. 7-8, S. 264–271, 1998

[SICK10] Hompage Firma SICK

[Siegwart04] Roland Siewart, Illah R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, Massachusette Intitute of Technology, The MIT Press, Cambridge 2004

[Stettmer94] Josef Stettmer, *Sensorgestützte Kollisionsvermeidung bei Industrierobotern*, Shaker Verlag, 1994

[Stobl06] K. H. Strobl and G. Hirzinger, *Optimal Hand-Eye Calibration*, In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), Beijing, China, pp. 4647-4653, October 2006

[Thrun05] Thrun, Sebastian, Wolfram Burgard and Dieter Fox, *Probabilistic Robotics*, The MIT Press: Cambridge, Massachusetts, 2005

[Thrun06] Sebastian Thrun, Mike Montendo, *Stanley: The Robot that Won the DARPA Grand Challenge*, Journal of Field Robotics 23(9), pp. 661-692, 2006

[Viola01] Viola and Jones, *Rapid object detection using boosted cascade of simple features*, Computer Vision and Pattern Recognition, 2001

[Walter51] Grey Walter, *A Machine that Learns*, Scientific American, pp. 60-63, August 1951

[Welch01] Greg Welch, Gary Bishop, *An Introduction to the Kalman Filter*, SIGGRAPH 2001, Course 8, University of Carolina, Los Angeles 2001

[Williams64] J. W. J. Williams, *Algorithm 232 - Heapsort*, 1964, Communications of the ACM 7(6): 347–348