MASTERARBEIT

Konzeption und Implementierung eines reaktiven Pfadplanungsverfahrens für 3D-Umgebungen basierend auf dem Elastic Band Framework

Vorgelegt von: Helge Scheel

am: 14.10.2014

zum

Erlangen des akademischen Grades

MASTER OF SCIENCE

(M.Sc.)

Erstbetreuer: Dipl.-Inform. Ingo Boersch

Zweitbetreuer: Dipl.-Ing. Matthias Gruhler
(Fraunhofer IPA)

Zweitbetreuer: Prof. Dr. Jochen Heinsohn

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema

Konzeption und Implementierung eines reaktiven Pfadplanungsverfahrens für 3D-Umgebungen basierend auf dem Elastic Band Framework

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Brandenburg an der Havel, den 14.10.2014

Unterschrift

Zusammenfassung

Moderne autonome mobile Roboter werden mit komplexen Aufgabenstellungen konfrontiert. Zur Erfüllung dieser Aufgaben agieren sie häufig in einer dynamischen Umgebung und müssen bei der Navigation drei Raumdimensionen berücksichtigen. Pfadplanung in solchen Umgebungen ist durch hohe Komplexität geprägt und Gegenstand aktueller Forschung.

In dieser Arbeit wird ein reaktives Pfadplanungsverfahren auf Grundlage des Elastic Band Framework implementiert. Das Elastic Band Framework löst das Problem der hohen Komplexität, indem es die Pfadplanung in zwei Teilprobleme aufteilt. Ein globaler Pfad wird unter Annahme einer statischen Umgebung initial geplant und als ein elastisches Band interpretiert. Interne und externe Kräfte modifizieren das elastische Band kontinuierlich und passen es den dynamischen Änderungen der Umgebung an.

Ein vorhandenes Elastic-Band-Pfadplanungsverfahren wird für den Serviceroboter Care-O-bot® 3 für den Einsatz in einer dreidimensionalen Umgebung erweitert. Dafür werden konzeptionelle Änderungen erarbeitet und unter dem Robot Operating System implementiert. Die Eignung des entwickelten Pfadplanungsverfahrens für den Anwendungsfall wird durch eine Evaluierung in einer Simulationsumgebung und durch einen Funktionsnachweis am physischen Roboter aufgezeigt.

Abstract

Modern autonomous mobile robots are confronted with complex tasks. To fulfill these tasks, the robots often have to navigate in a dynamic environment with three spatial dimensions. Path planning in such environments is complex and is subject to current research.

In this paper, a reactive path planning method is implemented based on the Elastic Band Framework. The Elastic Band Framework solves the problem of high complexity by dividing the path planning into two subproblems. A global path is planned under the assumption of a static environment and is interpreted as an elastic band. Internal and external forces continuously modify the elastic band and fit it to the dynamic changes of the environment.

An existing Elastic Band path planning method for the service robot Care-O-bot® 3 is extended for the use in a three-dimensional environment. For this purpose, conceptual changes are developed and implemented using the Robot Operating System. The suitability of the developed path planning method is verified by an evaluation in a simulation and by tests on the physical robot.

Inhaltsverzeichnis

1 Einleitung

Autonome und teilautonome mobile Roboter finden in vielen Einsatzgebieten weite Verbreitung. Die Aufgaben der Roboter und die Umgebungen, in denen sie agieren, werden im Zuge der weiteren Entwicklung und Verbreitung zunehmend komplexer.

Im Bereich der Servicerobotik wurden bspw. früh einfache Reinigungsroboter eingesetzt, deren Einsatzgebiet genau definiert war. Das Einsatzgebiet wurde ohne Interaktion mit der Umgebung nach einem festen Schema abgearbeitet. Weiterentwickelte Roboter konnten das Einsatzgebiet selbstständig erkunden, unbekannte Hindernisse umgehen und temporär blockierte Bereiche später erneut anfahren. Für solche Roboter ist eine zweidimensionale Modellierung der Umgebung in der Regel ausreichend. Pflege- und Assistenzroboter agieren mit Menschen, sind einer komplexeren Umgebung ausgesetzt und müssen neben der eigenen Sicherheit die Sicherheit der Menschen in ihrer Umgebung gewährleisten. Typische Einsatzgebiete wie Wohnungen, Büros oder Messehallen sind unstrukturiert und durch unbekannte Hindernisse sowie beengte Korridore geprägt. Um in solchen Umgebungen sicher und effizient agieren zu können, müssen die Roboter Kenntnis über ihre Umgebung in allen drei Raumdimensionen besitzen.

Fahrerlose Transportsysteme agierten zunächst in einer stark vereinfacht modellierten Umgebung, folgten einem festen Plan und konnten nicht auf veränderte Situationen in der Umgebung reagieren. Aktuelle Transportroboter können dynamisch auftretenden Hindernissen ausweichen, mit anderen Transportrobotern koordiniert vorgehen und selbstständig ihre Pläne ändern, um robuster und effizienter zu agieren. Außerhalb von ebenen Räumen eingesetzte Transportroboter müssen unter anderem die Unebenheiten des Bodens modellieren und bei der Bewegung berücksichtigen. Dazu müssen Höhenunterschiede gemessen und dreidimensional erfasst werden. Roboter für Such- und Rettungseinsätze sind besonders mit unebenen und komplexen Bodenstrukturen konfrontiert.

Aktuelle Entwicklungen im Bereich der Automobilbranche führen zu immer komplexeren Fahrassistenzsystemen bis hin zu autonom und im Kollektiv agierenden Fahrzeugen. Obwohl z. B. Autobahnen als ebene und strukturierte Umgebungen aufgefasst werden können,

werden Objekte auch dreidimensional erfasst, bspw. Verkehrsschilder oder andere Fahrzeuge. Im Stadtgebiet und bei unebenen Straßenverhältnissen ist eine Pfadplanung in einer 3D-Umgebung unablässig.

Roboter in anderen Anwendungsgebieten müssen ihre Umgebung nicht nur dreidimensional modellieren, sondern sich auch in drei Dimensionen bewegen. Dies betrifft z. B. Unterwasserroboter und Flugroboter, wie Drohnen, autonome Hubschrauber oder Lenkflugkörper. Für sie ist Pfadplanung in einer dreidimensional modellierten Umgebung unabdingbar.

1.1 Motivation

Viele Roboter müssen ihre Umgebung dreidimensional modellieren und ihre Bewegungen in dieser 3D-Umgebung im Voraus berechnen. Die Bewegungen müssen in der Regel geplant werden, damit der Roboter seine Aufgaben erfüllen und gleichzeitig Kollisionen mit Hindernissen vermeiden kann. Die Pfadplanung ist ein wichtiges Teilgebiet der Robotik und wird seit über 30 Jahren erforscht. Pfadplanung ist ein komplexes Problem, das hohen Rechenaufwand erfordert. In [CR87] wird nachgewiesen, dass eine vollständige Pfadplanung in einer dynamischen Umgebung NP-hart ist. Um dennoch Pfadplanung zu ermöglichen, werden vereinfachende Annahmen getroffen, z. B. durch eine zweidimensional modellierte Umgebung, obwohl der Roboter in drei Raumdimensionen agiert, oder die Annahme einer vollständig bekannten Umgebung. Für einfache Anwendungsfälle ist dadurch eine Pfadplanung möglich, die optimale Bewegungen des Roboters erstellt.

Oftmals sind Einschränkungen auf solch einfache Anwendungsfälle nicht möglich. Die Umgebungen, in denen Roboter agieren, können unstrukturiert als auch unbekannt sein und über sich bewegende Hindernisse mit unbekannten Bahnen verfügen. In vielen Anwendungsfällen ist auch die Einschränkung auf eine 2D-Umgebung nicht möglich. Um in solchen Anwendungsfällen Pfadplanung zu ermöglichen, werden existierende Pfadplanungsverfahren weiterentwickelt und die Komplexität durch andere Einschränkungen reduziert.

Das Elastic Band Framework ist ein reaktives Pfadplanungsverfahren. Die Komplexität der Pfadplanung wird reduziert, indem nicht mit jeder Änderung in der Umgebung ein neuer Pfad vollständig geplant wird. Stattdessen wird ein initial geplanter Pfad dynamisch an die Umgebung angepasst. Der Pfad wird als elastisches Band aufgefasst, das durch Hindernisse verformt wird und diesen dadurch ausweicht. Dadurch kann das Elastic Band Framework auch in unstrukturierten 3D-Umgebungen mit unbekannten Hindernissen eingesetzt werden.

1.2 Aufgabenstellung

Ziel dieser Arbeit ist die Konzeption und Implementierung eines reaktiven Pfadplanungsverfahrens für einen autonomen mobilen Roboter in einer 3D-Umgebung. Der Roboter soll unter Berücksichtigung seiner aktuellen Konfiguration und 3D-Sensorendaten kollisionsfrei zu einer definierten Zielpose navigieren. Basis für das Pfadplanungsverfahren ist das Elastic Band Framework.

Zur Umsetzung ist das Software-Framework Robot Operating System (ROS) vorgesehen. Für ROS ist ein Elastic-Band-Pfadplanungsverfahren vorhanden, das in einer 2D-Umgebung agiert. Dieses soll für den Einsatz in einer 3D-Umgebung erweitert werden.

1.3 Einordnung der Aufgabenstellung

Die Steuerung eines autonomen mobilen Roboters von einer Startpose zu einer Zielpose ist eine komplexe Aufgabenstellung, die sich aus mehreren Teilaufgaben zusammensetzt. In diesem Abschnitt wird erläutert, welche Teilaufgaben in dieser Arbeit bearbeitet werden.

Sensorik: Um den Zustand des Roboters und der Umgebung beobachten zu können, werden Sensoren verwendet, die den Roboter oder die Umgebung vermessen und dem Gesamtsystem kontinuierlich Daten zur Verfügung stellen. Typisch eingesetzte Sensoren sind z. B. Laserscanner und Stereokameras zur Beobachtung der Umgebung und Inkrementalgeber zur Beobachtung der Räder. In dieser Arbeit werden kontinuierlich aktualisierte und aufbereitete Sensordaten als gegeben vorausgesetzt.

Lokalisierung: Die Lokalisierung ermittelt die Pose des Roboters in der Umgebung. Sensordaten des Roboters und der Umgebung werden kombiniert, um den wahrscheinlichsten Aufenthaltsort zu ermitteln. Oftmals werden Verfahren zur simultanen Lokalisierung und Kartierung (simultaneous localization and mapping, SLAM) verwendet. Die Lokalisierung wird in dieser Arbeit unabhängig durch externe Systeme realisiert.

Umgebungsmodellierung: Informationen zur Umgebung des Roboters werden meist in einer für den Roboter geeigneten Repräsentation explizit modelliert. Das Weltmodell kann z. B. die Hindernisse der Umgebung, den Zustand des Roboters und Informationen zu Zielstellungen des Roboters beinhalten. In dieser Arbeit wird für das Pfadplanungsverfahren eine Karte zur Repräsentation der Hindernisse erstellt.

Pfadplanung: Die Pfadplanung erstellt eine Sequenz von Konfigurationen, die von einer Startpose zu einer Zielpose führt. Im Fokus dieser Arbeit steht ein Verfahren, das einen

initial geplanten Pfad dynamisch an die Umgebung anpasst und Kollisionen mit Hindernissen vermeidet.

Planausführung: Der geplante Pfad muss durch konkrete Bewegungsbefehle umgesetzt werden. Dabei müssen die Dynamik des Roboters in Form von z. B. Maximalgeschwindigkeiten berücksichtigt und abrupte Beschleunigungen möglichst vermieden werden. Die Planausführung wird in dieser Arbeit berücksichtigt.

Aktorik: Die erstellten Bewegungsbefehle müssen an die verwendeten Aktoren angepasst und in elektrische Steuersignale umgewandelt werden. Von der Hardware wird in dieser Arbeit abstrahiert.

1.4 Anforderungen

Das zu entwickelnde Pfadplanungsverfahren muss einen kollisionsfreien Pfad zu einer definierten Zielpose planen. Hindernisse müssen dreidimensional beachtet werden. Die Umgebung ist dynamisch, d. h. es können sich bewegende Hindernisse mit unbekannten Bahnen auftreten. Falls das Ziel nicht erreicht werden kann, soll die Fahrt abgebrochen und eine entsprechende Warnung ausgegeben werden.

Der Roboter besteht aus einer mobilen Basis, auf die der Roboterkörper und bewegliche Komponenten wie ein Manipulator montiert sind. Bei der Pfadplanung muss die vollständige Geometrie des Roboters inkl. der beweglichen Komponenten berücksichtigt werden. Dazu muss der Roboter approximativ in drei Dimensionen modelliert werden. Die Posen der beweglichen Komponenten können während einer Fahrt durch externe Systeme verändert werden. Für die Pfadplanung werden die Posen der beweglichen Komponenten jederzeit als gegeben vorausgesetzt.

Der Pfad soll für die mobile Basis des Roboters geplant werden. Der Roboter kann Translationsbewegungen auf der x-y-Ebene und Rotationsbewegungen um die z-Achse[1] ausführen. Das Pfadplanungsverfahren muss daher anhand von Daten zur Umgebung in drei Dimensionen einen Plan zur Bewegung entlang von zwei Dimensionen erstellen. Für die Pfadplanung können die beweglichen Komponenten als statisch angenommen werden. Pfadplanung unter Nutzung der Freiheitsgrade der beweglichen Komponenten (whole-body motion planning) ist nicht Ziel dieser Arbeit.

Das Elastic Band bzw. Elastic Strip Framework realisiert eine reaktive Pfadplanung. Es modifiziert einen vorgegebenen, kollisionsfreien Pfad zur Zielpose entsprechend der dynamischen

[1] In dieser Arbeit wird ein zweidimensionaler Raum durch die x-Achse und die y-Achse beschrieben. Im dreidimensionalen Raum wird diese Anordnung durch die z-Achse in der Höhe erweitert.

Umgebung. Der initiale Pfad wird als gegeben vorausgesetzt. Das zu entwickelnde Pfadplanungsverfahren soll einen ungültig gewordenen Pfad erkennen. Im Fall eines als ungültig erkannten Pfades kann es jederzeit eine Neuplanung des initialen Pfades veranlassen.

Für die Pfadplanung muss eine geeignete Umgebungsrepräsentation zur Verfügung gestellt werden. Da es sich um eine reaktive Pfadplanung handelt, soll die Umgebungsrepräsentation nicht die gesamte Umgebung abbilden, sondern nur einen aktuell relevanten Ausschnitt. Die vom Roboter erfassten 3D-Sensordaten müssen in die Umgebungsrepräsentation integriert werden. Die Sensordaten werden dabei durch externe Komponenten zur Verfügung gestellt. Sensordatenaufbereitung und SLAM werden nicht vom Pfadplanungsverfahren realisiert.

Das Pfadplanungsverfahren soll Bewegungsbefehle zur Umsetzung der Fahrt entlang des geplanten Pfades generieren. Eine Optimierung der Bewegungsbefehle durch z. B. Regler und Anpassung auf die verwendete Hardware wird durch externe Komponenten realisiert und ist im Rahmen der Pfadplanung nicht notwendig.

Das entwickelte Pfadplanungsverfahren soll in einer Simulation und am physischen Roboter getestet und evaluiert werden. Dazu soll die Güte des geplanten Pfades anhand von Leistungsmerkmalen experimentell ermittelt werden.

1.5 Aufbau der Arbeit

Nach den einführenden Betrachtungen in Kapitel 1 werden in Kapitel 2 Grundlagen zur Pfadplanung und zur gegebenen Aufgabenstellung erläutert. Auf Pfadplanung für eine dynamische 3D-Umgebung wird in Kapitel 3 genauer eingegangen. In Kapitel 4 wird ein Konzept für eine reaktive Pfadplanung basierend auf dem Elastic Band Framework für die gegebene Aufgabenstellung entwickelt. Die Implementierung dieses Konzepts wird in Kapitel 5 beschrieben und in Kapitel 6 evaluiert. Kapitel 7 umfasst ein Fazit zu dieser Arbeit und gibt einen Ausblick über Weiterentwicklungen des umgesetzten Pfadplanungsverfahrens.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für das zu entwickelnde Pfadplanungsverfahren erläutert. Pfadplanungsverfahren abstrahieren generell von der eingesetzten Hardware und haben den Anspruch, für unterschiedlichste Roboter geeignet zu sein. Für eine konkrete Implementierung müssen jedoch die Eigenschaften des eingesetzten Roboters beachtet werden. Eigenschaften wie der Antrieb, die vorhandene Rechenleistung oder die verfügbaren Sensoren können wesentliche Einschränkungen mit sich bringen, die bereits bei der Konzeption berücksichtigt werden müssen. Daher wird in Abschnitt 2.1 zunächst der in dieser Arbeit verwendete Roboter beschrieben. Das zu entwickelnde Pfadplanungsverfahren baut auf vorhandenen Komponenten auf, die Teil des Software-Frameworks ROS sind. Zum besseren Verständnis des allgemeinen Aufbaus der Implementierung werden in Abschnitt 2.2 die Grundzüge von ROS erklärt. Anschließend wird die Integration des vorhandenen Pfadplanungsverfahrens in ROS in Abschnitt 2.3 beschrieben. Das Problem der Pfadplanung wird in Abschnitt 2.4 definiert. Darauf aufbauend werden in Abschnitt 2.5 grundlegende Pfadplanungsverfahren erläutert.

2.1 Care-O-bot® 3

Als Roboter wird für diese Arbeit der am Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA (Fraunhofer IPA) entwickelte Care-O-bot® 3 [GRH+09] verwendet. Der Serviceroboter basiert auf einer 75 x 55 cm messenden Antriebsplattform, in die ein omnidirektionaler Antrieb mit vier Rädern und einer Maximalgeschwindigkeit von ca. 1,5 m/s integriert ist. Für die Pfadplanung kann der Antrieb als holonom angenommen werden, d. h. jede Bewegungsrichtung kann direkt angesteuert werden.

Auf die Antriebsplattform ist der Körper des Roboters montiert. Dieser umfasst unter anderem die Rechnerarchitektur, einen separat steuerbaren Manipulator und einen Sensorkopf zur 3D-Umgebungserfassung. Die Rechnerarchitektur beinhaltet drei Rechner zur Steuerung des Roboters. Die Pfadplanung wird auf einem Rechner mit einer Intel Core2 Duo T7400 CPU und 3 GB RAM ausgeführt.

a) 

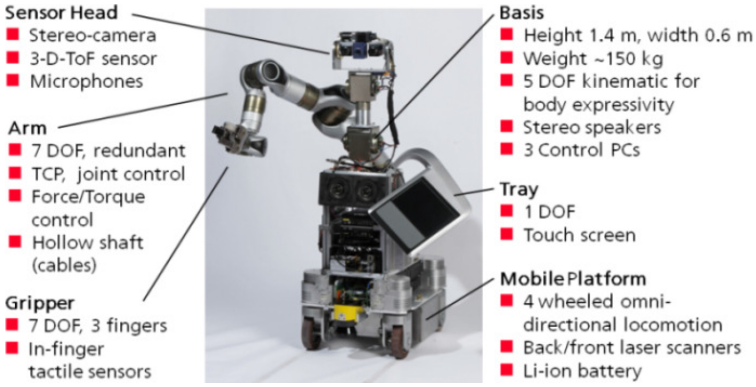
b) 

Abbildung 2.1: Care-O-bot® 3: Teilabbildung a) zeigt den in dieser Arbeit verwendeten Care-O-bot® 3-3. Der Aufbau des Roboters und seine Hardware-Komponenten sind in b) dargestellt.
Quelle: [GRH+09, S. 140]

Care-O-bot® 3 ist mit einem Manipulator ausgestattet, an dessen Ende ein Aktor montiert ist. Manipulator und Aktor variieren in verschiedenen Ausstattungsvarianten des Care-O-bot® 3. In dieser Arbeit wird der Care-O-bot® 3-3 verwendet, der einen Manipulator vom Typ Kuka LBR iiwa 7 R800[1] mit sieben Freiheitsgraden (degrees of freedom, DOF) und eine 3-Finger-Greifhand SDH der Firma Schunk[2] mit sieben DOF führt. Abbildung 2.1 zeigt Care-O-bot® 3-3 und den Aufbau der Hardware des Roboters.

Die Antriebsplattform besitzt zwei Laserscanner des Typs SICK S300 Professional CMS[3] und einen des Typs Hokuyu URG-04LX[4], die zusammen einen Erfassungsbereich von 360 Grad abdecken. Weitere Sensoren sind im Sensorkopf montiert. Der Sensorkopf verfügt über vier DOF und integriert ein Stereokamerasystem, bestehend aus zwei AVT Pike 145 C[5] Kameras, und eine Microsoft Kinect[6] als 3D-Kamera in das Gesamtsystem. Die Kinect verfügt über einen horizontalen Sichtbereich von 57 Grad und einen vertikalen Sichtbereich von 43 Grad. Details zu den technischen Daten sind auf der offiziellen Webseite des Roboters[7] zu finden.

Das vorrangige Anwendungsgebiet des Care-O-bot® 3 liegt im Assistenz- und Pflegedienst. Der Roboter ist für den Einsatz im häuslichen Umfeld konzipiert, kann einfache Aufgaben wie bspw. Hol- und Bringdienste übernehmen und mit Menschen interagieren. Weiterhin kann er als interaktiver Assistent bei z. B. Messen und als Forschungs- und Entwicklungsplattform verwendet werden.

[1] http://www.kuka-labs.com/NR/rdonlyres/5367918D-1B61-450E-9DB6-62673168A900/0/KUKA_iiwa_DE.pdf
[2] http://mobile.schunk-microsite.com/de/produkte/produkte/servoelektrische-3-finger-greifhand-sdh.html
[3] http://www.sick.com/group/DE/home/products/product_news/optoelectronic_protective_devices/Seiten/s300.aspx
[4] http://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx.html
[5] http://www.alliedvisiontec.com/de/produkte/kameras/firewire/pike/f-145bc.html
[6] http://www.xbox.com/en-US/kinect
[7] http://www.care-o-bot.de/de/care-o-bot-3/hardware/technical-data.html

2.2 Robot Operating System

Die Steuerung des Care-O-bot® 3 erfolgt unter Nutzung des Software-Frameworks Robot Operating System [QCG+09]. Im folgenden Abschnitt wird die Funktionsweise von ROS erläutert, indem die grundlegenden Elemente und ihre Interaktionen beschrieben werden. Anschließend wird auf die Verwendung von ROS eingegangen.

2.2.1 Grundaufbau

ROS agiert als Middleware und ermöglicht die dezentrale Kommunikation unabhängiger Software-Pakete mit verschiedenen Aufgaben, sogenannter ROS-Pakete. Die ROS-Pakete können somit aufgaben- und roboterspezifisch zusammengestellt und flexibel in das Gesamtsystem zur Robotersteuerung integriert werden. Die Aufgaben von ROS-Paketen reichen von der Kapselung von Sensortreibern und der Bereitstellung ihrer Daten über unterstützende Dienste wie die Transformation zwischen Koordinatensystemen bis hin zu höheren Aufgaben wie Lokalisierung und Pfadplanung.

ROS-Metapakete beinhalten mehrere, thematisch oder funktionell zusammengehörige ROS-Pakete und werden zur Organisation von ROS-Paketen verwendet. ROS-Pakete stellen einen oder mehrere Knoten (Nodes) zur Verfügung, die Datenströme via multicast zu bestimmten Themengebieten (Topics) bereitstellen und auf Datenströme von anderen Knoten zugreifen (publish, subscribe). Knoten können zudem Dienste (Services) anbieten. Ein Dienst realisiert eine Ende-zu-Ende-Kommunikation nach dem Prinzip Anfrage/Antwort. Ein zentraler Master-Knoten registriert und verwaltet alle Knoten sowie deren Themengebiete und Dienste. Die Verknüpfung von Knoten untereinander ist in Abbildung 2.2 illustriert.

Die Knoten bilden eigenständige Prozesse, die nach dem Peer-to-Peer-Verfahren ohne zentralen Server kommunizieren. Die Datenströme zwischen Knoten werden mittels typisierter Nachrichten übertragen, die sich aus primitiven Datentypen zusammensetzen und verschachteln lassen [HLN12, S. 329]. ROS koordiniert den Transport der Nachrichten über eine eigene Transportschicht, die auf TCP/IP aufsetzt.

2.2.2 Verwendung

ROS ist unter mehreren UNIX-basierten Betriebssystemen wie Linux oder Mac OS lauffähig, hauptsächlich werden die Linux-Distribution Ubuntu und Mac OS X unterstützt. Die Verwendung unter Microsoft Windows ist experimentell. ROS unterstützt verschiedene Programmiersprachen, primär C++, Python und Lisp. Für die Implementierung wird in

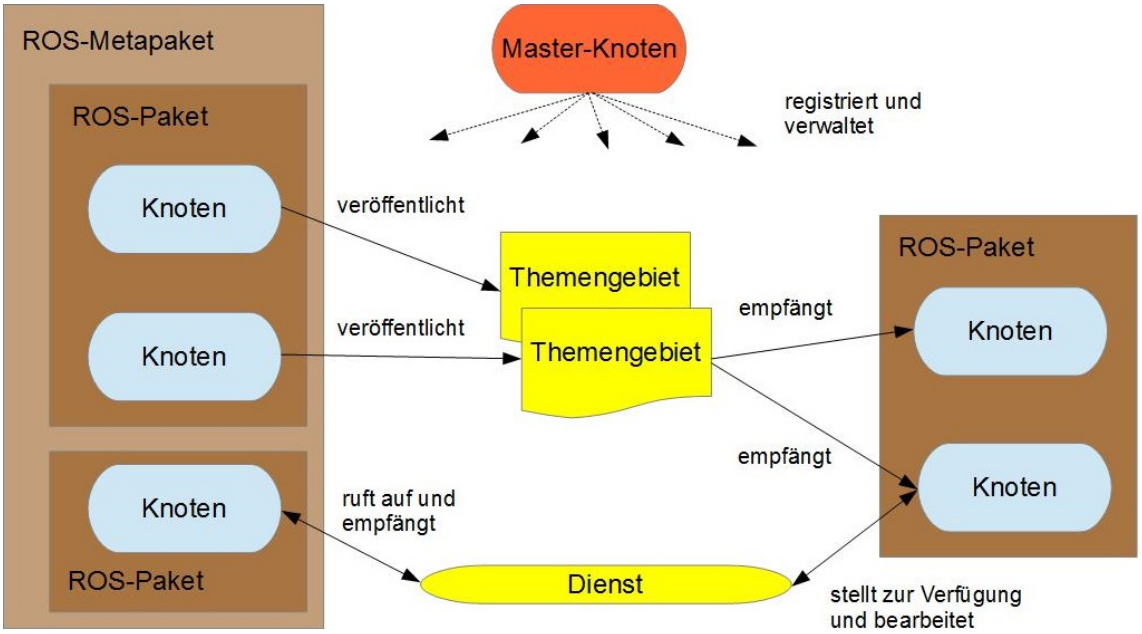


Abbildung 2.2: Schematischer Grundaufbau von ROS: Knoten veröffentlichen Datenströme, empfangen Datenströme über Themengebiete und stellen Dienste zur Verfügung. Der Master-Knoten registriert und verwaltet alle Komponenten. Knoten sind in ROS-Paketen und ROS-Metapaketen organisiert.

dieser Arbeit C++ unter der Linux-Distribution Ubuntu in Version 12.04 verwendet. Der Kern von ROS ist quelloffene Software und untersteht der BSD-Lizenz.

Die Versionierung des ROS-Kerns erfolgt durch sogenannte ROS-Distributionen. Die ROS-Pakete werden durch die jeweiligen Entwickler an die Änderungen des ROS-Kerns angepasst. ROS-Pakete unterschiedlicher ROS-Distributionen können untereinander inkompatibel sein. In dieser Arbeit wird die ROS-Distribution Hydro Medusa verwendet. Die aktuelle ROS-Distribution Indigo Igloo wurde nach Beginn dieser Arbeit veröffentlicht. Die vorherigen ROS-Distributionen Groovy Galapagos und Hydro Medusa werden weiterhin vollständig unterstützt. Ältere ROS-Distributionen werden nicht weiterentwickelt, können aber noch verwendet werden.

ROS wird von mehreren Robotern unterschiedlicher Kategorien verwendet. Exemplarisch seien Adept MobileRobots Pioneer 3 als mobiler Roboter, Care-O-bot® 3 als mobiler Roboter inkl. Manipulator, Schunk SDH als Manipulator, Aldebaran Robotics Nao als humanoider Roboter, Skybotix's CoaX Helicopter als Flugroboter und Nessie V als Unterwasserroboter erwähnt. Eine umfassende Übersicht über Roboter, die ROS verwenden, ist auf dem ROS-Wiki[8] zu finden.

[8] http://wiki.ros.org/Robots

2.3 Systemumgebung

Das vorhandene Pfadplanungsverfahren ist ein integrierter Bestandteil eines komplexen Gesamtsystems zur Steuerung des Roboters. In diesem Abschnitt wird die Systemumgebung und die Integration des Pfadplanungsverfahrens überblicksweise dargestellt. Dazu werden zunächst die wesentlichsten ROS-Pakete und ROS-Metapakete beschrieben, die in der Systemumgebung verwendet werden. Anschließend wird das vorhandene Pfadplanungsverfahren als Teil des Gesamtsystems vorgestellt.

2.3.1 ROS-Umgebung

Abbildung 2.3 zeigt die Interaktion der grundlegenden ROS-Pakete in der Systemumgebung in dieser Arbeit. Gazebo ist eine Simulationsumgebung, die eine modellierbare Umgebung, Roboter und deren Interaktionen mit der Umgebung durch physikalische Prozesse simuliert. Die von Gazebo simulierten Sensordaten und Zustände von Robotern und Objekten der Umgebung werden über ROS veröffentlicht. Andere ROS-Knoten können diese Daten empfangen und analog zu Daten von physikalischen Robotern verwenden. Eine graphische Benutzeroberfläche erlaubt visuelle Überwachung und Interaktion mit der simulierten Umgebung.

Rviz ist ein Visualisierungsprogramm, mit dem physische und simulierte Roboter sowie ihre Umgebung überwacht werden können. Das Programm greift auf veröffentlichte Datenströme zu, um Roboter, die Umgebung und zusätzliche Informationen wie Sensordaten oder von der Pfadplanung genutzte Karten visuell darzustellen. Über die Benutzeroberfläche kann eine Zielpose für den Roboter definiert werden.

Ein Roboter wird durch Gelenke modelliert, für die jeweils ein eigenes Koordinatensystem definiert wird. Das ROS-Paket tf2 verwaltet die Koordinatensysteme in einem Transformationsgraphen und veröffentlicht Transformationen zwischen den Koordinatensystemen. Dadurch kann die Pose eines Gelenks im Koordinatensystem eines beliebigen anderen Gelenks ausgedrückt und manipuliert werden.

Das ROS-Metapaket navigation integriert verschiedene zur Navigation eines Roboters benötigte Funktionalitäten wie Lokalisierung, Umgebungsrepräsentation und Pfadplanung. Für die vorhandene Pfadplanung relevante Funktionalitäten und Unterpakete werden im nächsten Abschnitt behandelt.

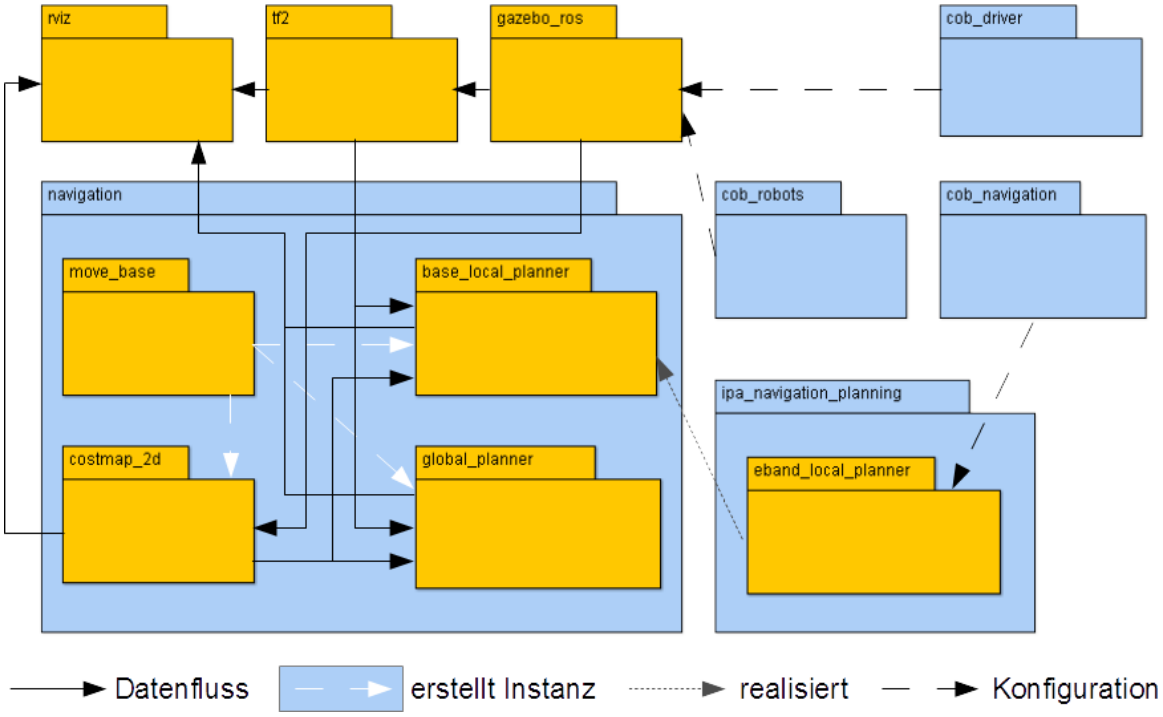


Abbildung 2.3: Überblick Systemumgebung: Gazebo_ros veröffentlicht simulierte Daten für die Umgebung und den Status des Roboters. Tf2 repräsentiert den Roboter als Transformationsgraphen der Koordinatensysteme seiner Gelenke. Die Pfadplanungskomponenten base_local_planner und global_planner nutzen die Datenströme der Umgebungsrepräsentation costmap_2d und von tf2. Rviz greift auf veröffentlichte Daten über Karten, Pfadpläne sowie den Roboter zu und visualisiert sie.

2.3.2 Gesamtsystem der Pfadplanung

Zur Ausführung des Pfadplanungsverfahrens werden mehrere Unterpakete des ROS-Metapakets navigation und andere ROS-Pakete genutzt. Abbildung 2.3 gibt einen Überblick über den strukturellen Aufbau des Gesamtsystems.

Ein Knoten des ROS-Pakets move_base agiert als zentrale Instanz für die Pfadplanung, indem er die benötigten Komponenten erstellt und verwaltet. Das ROS-Paket costmap_2d wird zur Umgebungsrepräsentation verwendet. Die Funktionsweise wird in Abschnitt 4.1.2 beschrieben. Das Paket base_local_planner enthält die Basisklasse des vorhandenen Pfadplanungsverfahrens. Details zur Implementierung werden in Abschnitt 5.1 erläutert. Das Pfadplanungsverfahren hat Schnittstellen zu move_base und zu costmap_2d. Um eine einfache Integration in das Gesamtsystem und die Erweiterbarkeit des Gesamtsystems zu gewährleisten, sind Änderungen an den vorhandenen Schnittstellen gering zu halten.

Die ROS-Metapakete cob_robots, cob_driver und cob_navigation umfassen Dateien, die zur Modellierung der Roboter des Fraunhofer IPA und der Ausführung der Pfadplanung benötigt

werden. Das ROS-Metapaket cob_robots enthält unter anderem das Robotermodell des Care-O-bot® 3 als XML-Baum. Gerätetreiber für die Hardware-Komponenten des Roboters liegen im ROS-Metapaket cob_driver. Konfigurationsdateien für die Navigation einschließlich der Pfadplanung sind im ROS-Metapaket cob_navigation enthalten.

Das vorhandene Pfadplanungsverfahren befindet sich im ROS-Paket eband_local_planner und ist Teil des ROS-Metapakets ipa_navigation_planning. Das Verfahren realisiert eine reaktive Pfadplanung mit einer zweidimensionalen Umgebungsrepräsentation. Die Repräsentation des Roboters basiert auf der vom Roboter eingenommenen, zweidimensionalen Grundfläche. Zur Vereinfachung der Pfadplanung wird der Roboter durch einen Kreis repräsentiert, dessen Radius dem Außenkreisradius der Grundfläche entspricht. Hindernisse entlang der z-Achse können ignoriert oder erfasst und in die Ebene projiziert werden. Eine präzise dreidimensionale Repräsentation ist nicht möglich. Details zu Umgebungsrepräsentationen, der Abbildung einer 3D-Umgebung und der Funktionsweise der verwendeten Umgebungsrepräsentation sind in den Abschnitten 2.4.3, 3.6 bzw. 4.1.2 zu finden.

2.4 Pfadplanung

Nachdem die Pfadplanung in Abschnitt 1.3 im Kontext der Steuerung eines mobilen Roboters eingeordnet wurde, werden in diesem Abschnitt das Problem der Pfadplanung konkretisiert und Grundlagen zur Lösung beschrieben. Dazu werden zunächst Definitionen für grundlegende Begriffe im Zusammenhang mit der Pfadplanung gegeben. Anschließend werden zwei Räume eingeführt, in denen Pfadplanung realisiert werden kann, und Grundlagen zur Repräsentation der Umgebung eines Roboters erläutert.

2.4.1 Begriffsdefinitionen

Pfadplanung bei mobilen Robotern beschreibt das Finden eines Pfades von einer Startpose zu einer Zielpose, ohne dabei mit Hindernissen zu kollidieren [SK08, S. 110]. Dafür wird eine Beschreibung des Roboters und eine vollständige oder partielle Beschreibung der Umgebung als gegeben vorausgesetzt. Eine Pose beschreibt Position und Orientierung des Roboters oder eines Objektes der Umgebung.

Als Pfad wird eine Sequenz von Konfigurationen verstanden, die Start- und Zielpose verbindet [vgl. SK08, S. 110]. Eine Konfiguration beschreibt den Zustand des Roboters. Ein Pfad ist nicht durch die zeitliche Dimension parametrisiert, d. h. er ist zeitunabhängig [vgl. Bro00, S. 13]. Ein Pfadplanungsverfahren ist ein Algorithmus, der als Eingabe eine Start- und Zielpose

erhält und unter Beachtung der Umgebung und der Robotereigenschaften als Ausgabe einen Pfad zwischen den Posen generiert.

An einen Pfad können verschiedene Optimalitätskriterien gebunden sein, z. B. minimale Wegstrecke, maximaler Abstand zu Hindernissen, minimale Richtungsänderungen während der Planausführung oder minimale Fahrzeit. Neben der Qualität des generierten Pfades gelten für ein Pfadplanungsverfahren weitere Kriterien. Ein Pfadplanungsverfahren ist vollständig, wenn es genau dann einen Pfad für eine Eingabe generiert, wenn ein gültiger Pfad existiert. Als robust gilt ein Pfadplanungsverfahren, wenn es Unsicherheiten in der Modellierung der Robotereigenschaften und der Umgebung berücksichtigt. Neben der Güte des erstellten Pfades ist Rechenaufwand zur Planung des Pfades relevant.

2.4.2 Arbeits- und Konfigurationsraum

Pfadplanung kann im Arbeitsraum oder im Konfigurationsraum erfolgen. Der Arbeitsraum W beschreibt alle vom Roboter erreichbaren Punkte im physischen Raum. Er wird geometrisch in einem kartesischen euklidischen Koordinatensystem dargestellt. Eine Konfiguration in W beschreibt durch eine Pose die Position und Orientierung des Roboters in einem absoluten oder relativen Koordinatensystem. Je nach Robotertyp und Einsatzgebiet kann der Arbeitsraum zwei- oder dreidimensional sein. Der Roboter wird im Arbeitsraum meist durch ein vereinfachtes geometrisches Modell repräsentiert [vgl. Gut94, S. 39].

Im Konfigurationsraum C wird der Roboter anhand seiner n Freiheitsgrade repräsentiert [LP83, S. 109]. Eine Konfiguration in C beschreibt die Zustände aller Gelenke und Aktoren des Roboters. Der Konfigurationsraum stellt einen n-dimensionalen euklidischen Raum dar. Der Roboter wird in ihm als ein Punkt repräsentiert. Die Geometrie des Roboters wird auf die Darstellung der Hindernisse übertragen. Ein Hindernis wird durch alle Konfigurationen beschrieben, die eine Kollision mit dem Roboter verursachen [LP83, S. 110]. Abbildung 2.4 stellt ein Beispiel für den Arbeitsraum eines einfachen Manipulators mit zwei rotatorischen Freiheitsgraden und den zugehörigen Konfigurationsraum dar.

In beiden Räumen wird zwischen belegten und freien Zuständen unterschieden. Ein belegter Zustand repräsentiert ein Hindernis, mit dem der Roboter kollidieren kann. In W sind die belegten Zustände die Menge der Raumkoordinaten, die ein Hindernis beschreiben, in C sind sie die Menge der Konfigurationen, bei denen der Roboter mit einem Hindernis kollidiert [vgl. Bro00, S. 12]. Der Freiraum bildet die Menge der nicht belegten Zustände.

Im Konfigurationsraum wird der Roboter als Punkt und ein Pfad als eindimensionale Kurve dargestellt. Diese einfache Repräsentation ist bei der Pfadplanung von Vorteil [SK08, S.

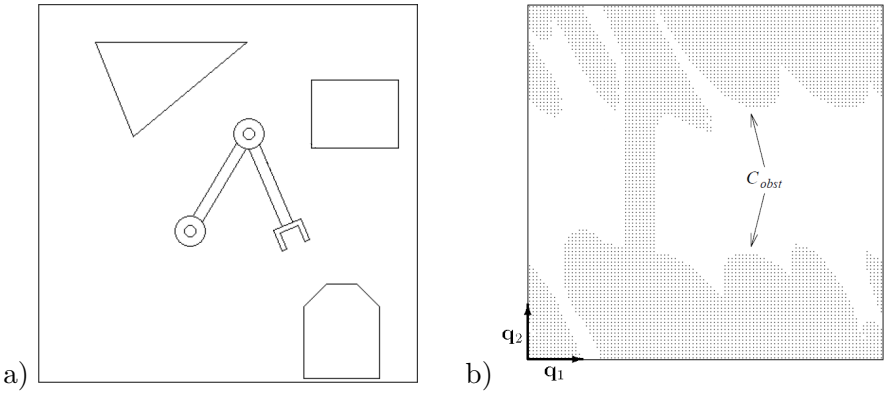


Abbildung 2.4: Arbeits- und Konfigurationsraum: Teilabbildung a) zeigt den zweidimensionalen Arbeitsraum eines Manipulators mit zwei Gelenken. Die umrandeten Bereiche bilden Hindernisse. In b) ist der zugehörige Konfigurationsraum dargestellt. Der Konfigurationsraum ist ebenfalls zweidimensional, da der Manipulator zwei DOF besitzt (Rotationen q1 und q2 um die beiden Gelenke). Der grau schraffierte Bereich beschreibt die transformierten Hindernisse. Quelle: [vgl. Qui94, S. 40]

110]. So kann z. B. der Abstand zwischen dem Roboter und dem nächsten Hindernis durch den Abstand zwischen zwei Punkten bestimmt werden. In W wird der Roboter durch eine geometrische Struktur repräsentiert. Die Komplexität der Abstandsberechnung zum nächsten Hindernis ist von der Struktur abhängig. Die Anzahl der Dimensionen von ist unabhängig von den Freiheitsgraden des Roboters und auf maximal drei begrenzt [Bro00, S. 13].

Da in C die Dimensionalität an die DOF geknüpft ist, kann der Raum hochdimensional und Berechnungen in ihm entsprechend komplex werden. In W gegebene Hindernisse und Posen, wie Start- und Zielposen, müssen für eine Pfadplanung in C transformiert werden. Diese Transformation ist rechenintensiv, insbesondere bei der Pfadplanung für eine 3D-Umgebung. Die Transformation kann approximiert werden, wodurch die Genauigkeit der Abbildung sinkt [Gut94, S. 40]. Objekte und Posen sind im physischen Raum anschaulicher, daher wird für die manuelle Programmierung eines Roboters meist W verwendet [Spa92, S. 26]. Pfadplanungsverfahren können in der Regel in beiden Räumen angewendet werden. Der Rechenaufwand steigt in höherdimensionalen Räumen unterschiedlich stark, wodurch einige Pfadplanungsverfahren in C ineffizient sein können.

2.4.3 Umgebungsrepräsentation

Pfadplanungsverfahren agieren auf einer Umgebungsrepräsentation, die relevante Ausschnitte der Umgebung abbildet. Vorrangig ist dies der Freiraum, bzw. Hindernisse und unbekannte Bereiche, die den Freiraum einschränken. Die Wahl einer geeigneten Umgebungsrepräsentati-

on ist entscheidend für den effizienten Einsatz eines Planungsverfahrens.

Die Umgebung des Roboters kann auf verschiedene Arten repräsentiert werden. Sie kann aus einer oder mehreren Karten bestehen und zusätzliche Informationen beinhalten. Karten werden in drei Kategorien eingeteilt. Topologische Karten bilden die Umgebung anhand eines Graphen ab, dessen Knoten relevante Objekte der Umgebung sind. Der Graph repräsentiert die geometrischen Beziehungen zwischen Objekten, nicht ihre absoluten Positionen [BFE96, S. 170]. Distanzen zwischen Objekten sind nicht maßstabsgetreu.

Geometrische Karten stellen maßstabsgetreue Distanzen sicher, indem sie die Umgebung in einem absoluten Koordinatensystem repräsentieren. Sie sind meist feiner strukturiert und bieten eine höhere Auflösung als topologische Karten [Thr02, S. 1]. Häufig verwendete geometrischen Karten sind Gitterkarten [ME85; Elf89]. Bei einer 2D-Gitterkarte wird die Umgebung z. B. durch ein zweidimensionales Gitter abgebildet. Für jede Gitterzelle des Gitters wird die Belegung bzw. Nichtbelegung durch ein Hindernis kodiert. Alternativ kann die Wahrscheinlichkeit der Belegung kodiert werden. Gitterkarten können schnell aufgebaut werden und ermöglichen eine einfache Integration der Daten verschiedener Sensorquellen [BFE96, S. 164]. Da nicht nur die Objekte, sondern alle Gitterzellen abgebildet werden, benötigt die Karte mehr Speicher als eine topologische.

Hybride Karten kombinieren beide Ansätze, indem sie z. B. einen Graphen bilden, dessen Knoten Ausschnitte der Umgebung in Form von lokalen geometrischen Karten sind [Nie13, S. 57].

2.4.4 Unsicherheit

Die Steuerung eines mobilen Roboters ist durch Unsicherheiten geprägt. Die aktuelle Pose des Roboters in der Umgebung ist nicht exakt bestimmt, sondern wird anhand der Sensoren approximiert. Durch die Approximation und ungenaue Sensordaten entsteht Positionsunsicherheit. Geplante Bewegungen des Roboters werden durch die Hardware fehlerbehaftet umgesetzt. Die resultierende Abweichung zwischen approximierter und realer Pose des Roboters steigt daher mit zunehmender Bewegung und vergrößert die Positionsunsicherheit.

Mit der Lokalisierung wird die Pose des Roboters approximiert und versucht, die Positionsunsicherheit zu minimieren. Dazu werden unter anderem die Eigenbewegung des Roboters durch interne Sensoren ausgewertet und die geschätzte Pose über Entfernungsmessungen mit bekannten Objekten der Umgebung verglichen. Die Sensoren liefern mit Messfehlern behaftete Sensordaten, haben einen begrenzten Erfassungsbereich und erfassen teilweise

nicht alle Objekte der Umgebung. Die daraus resultierende Unsicherheit kann durch Sensordatenfusion reduziert werden. Sensordatenfusion wird oft mithilfe von statistischen Daten über die Messfehler durch einen Kalman-Filter realisiert.

Bei der Pfadplanung wird Unsicherheit meist über die Umgebungsrepräsentation berücksichtigt. Einige Umgebungsrepräsentationen erlauben, die Unsicherheit der zugrunde liegenden Sensordaten explizit zu modellieren. Beispielsweise kann bei einer Gitterkarte jede Gitterzelle die Belegungswahrscheinlichkeit durch ein Hindernis angeben. Zur impliziten Repräsentation von Unsicherheit können Schwellwerte verwendet werden. Liegt die Belegungswahrscheinlichkeit einer Gitterzelle unter bzw. über einem definierten Schwellwert, wird sie als frei bzw. belegt markiert.

2.5 Grundlegende Pfadplanungsverfahren

In diesem Abschnitt werden drei grundlegende Verfahren zur Pfadplanung vorgestellt, um prinzipielle Vorgehensweisen zur Lösung des Pfadplanungsproblems zu verdeutlichen. In praktischen Anwendungen werden weiterentwickelte Verfahren eingesetzt, die auf den in diesem Abschnitt vorgestellten grundlegenden Pfadplanungsverfahren basieren. Ein umfassenderer Überblick über Pfadplanungsverfahren ist in [SK08] und [Lat91] zu finden.

2.5.1 Roadmap

Eine Roadmap ist ein Graph, der Verbindungen zwischen Teilmengen des Freiraumes des n-dimensionalen Raums darstellt. Die Kanten des Graphen bilden eindimensionale Kurven, die die Knoten verbinden. Zur Pfadplanung werden Start- und Zielposen in den Graphen integriert und eine Graphsuche zwischen beiden Posen durchgeführt. Der Aufbau des Graphen ist der rechenintensivste Schritt der Pfadplanung [Bro00, S. 14; Gut94, S. 35].

Beim Sichtbarkeitsgraphen [LPW79] werden die Hindernisse als konvexe Polygone abgebildet. Die Knoten des Sichtbarkeitsgraphen bilden Eckpunkte der Polygone sowie die Start- und Zielposen. Zwischen zwei Knoten, die durch eine Gerade ohne Kontakt zu einem Hindernis verbunden werden können, wird eine Kante angelegt. Ein Beispiel ist in Abbildung 2.5 dargestellt. In einem zweidimensionalen Raum wird bei diesem Verfahren stets der kürzeste Pfad gefunden, in höherdimensionalen Räumen ist dies nicht garantiert [Gut94, S. 35]. Da als Knoten Eckpunkte von Hindernissen verwendet werden, führt der gefundene Pfad dicht an Hindernissen vorbei, was das Kollisionsrisiko durch Unsicherheiten bei der Wahrnehmung

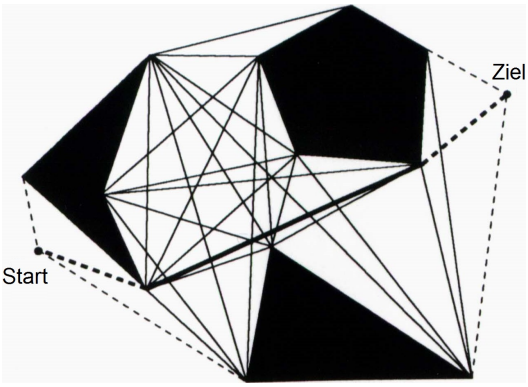


Abbildung 2.5: Sichtbarkeitsgraph: Der Graph besteht aus den Eckpunkten der Hindernisse als Knoten und schwarzen Geraden als Kanten. Die fett markierte Sequenz von Kanten bildet den gefundenen Pfad von der Start- zur Zielkonfiguration. Quelle: [vgl. Lat91, S. 156]

und Pfadplanung erhöht. Um einen größeren Abstand zu den Hindernissen zu gewährleisten, können die Hindernisse zuvor vergrößert werden.

Ein häufig verwendeter Ansatz für Roadmaps bilden Voronoi-Diagramme [SH75; Aur91]. Im zweidimensionalen Raum bestehen die Kanten eines Voronoi-Diagramms aus Punkten, die zu zwei Hinderniszentren den gleichen Abstand haben. Die Knoten sind Schnittpunkte der Kanten und besitzen den gleichen Abstand zu drei Hinderniszentren. Die Kanten bilden Pfade mit maximalem Abstand zu Hindernissen. Nach der Integration der Start- und Zielposen wird eine Graphsuche entlang der Kanten durchgeführt. Dies ist in Abbildung 2.6 illustriert. Wird ein gültiger Pfad gefunden, hat dieser in jeder Konfiguration den maximalen Abstand zu den nächsten Hindernissen. Die Komplexität zur Berechnung des Voronoi-Diagramms steigt mit der Dimensionalität des Raumes, in dem geplant wird. Das Verfahren erstellt sichere, aber unter Umständen unnötig lange Pfade. In [GMBJ11] wurde ein auf einem Voronoi-Diagramm basierendes Pfadplanungsverfahren für den Einsatz in 3D-Umgebungen erweitert.

2.5.2 Zellzerlegung

Zellzerlegungsverfahren teilen den Freiraum in kleine Einheiten, sogenannte Zellen, ein. Nach der Einteilung wird ein Graph zwischen benachbarten Zellen erstellt, der die Verbindungen zwischen Teilmengen des Freiraumes repräsentiert. Nach der Integration von Start- und Zielposen wird anschließend mittels einer Graphsuche ein Pfad zwischen beiden Posen erstellt.

Bei der exakten Zellzerlegung [SS83] bilden die Zellen den Freiraum genau ab. Im zweidimensionalen Fall wird der Raum in eine Menge konvexer Polygone zerlegt, die sich nicht

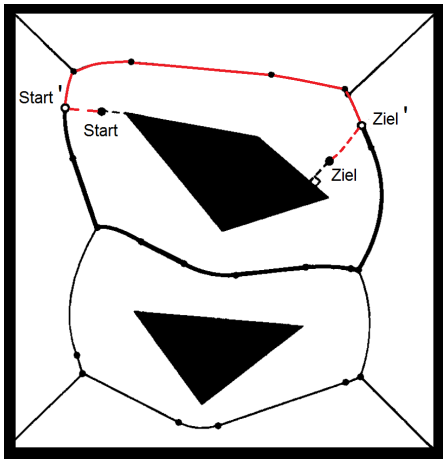


Abbildung 2.6: Voronoi-Diagramm: Die schwarz markierten Kanten des Graphen führen mit maximalem Abstand an Hindernissen vorbei. Zur Pfadplanung werden die zusätzlichen Konfigurationen Start ' und Ziel ' nahe den realen Start- und Zielkonfigurationen Start und Ziel in den Graph eingefügt, falls diese außerhalb des Graphen liegen. Der rot dargestellt Pfad wird über eine Graphsuche entlang der Kanten erstellt.
Quelle: [vgl. Lat91, S. 14]

überlappen und entweder nur Freiraum oder nur belegte Bereiche abbilden. Für die Zellzerlegung existieren verschiedene Algorithmen. Um einen Graphen zwischen den Zellen zu erstellen, wird im Zentrum jeder freien Zelle ein Knoten angelegt. Benachbarte Knoten werden mit einer Kante verbunden. Da jeder Pfad innerhalb einer freien Zelle kollisionsfrei ist, sind auch Kanten als Pfade zwischen zwei benachbarten Zellen kollisionsfrei. In dem so entstandenen Graphen wird ein Pfad zwischen der Start- und der Zielpose gesucht. Ein Beispiel für einen mit exakter Zellzerlegung gefundenen Pfad ist in Abbildung 2.7 dargestellt. Das Verfahren kann mit Polyedern als Zellen auf höherdimensionale Räume erweitert werden, in dem die Hindernisse als Polyeder modelliert sind [Gut94, S. 36]. Pfadplanung mittels exakter Zellzerlegung ist vollständig. Das Verfahren ist allerdings für höhere Dimensionen rechenintensiv und aufwendig zu implementieren.

Bei der approximativen Zellzerlegung [BLP85] wird der Raum mit n Dimensionen in gleichförmige, sich nicht überschneidende n-dimensionale Zellen zerlegt. Jede Zelle, die ein Hindernis teilweise oder ganz beinhaltet, wird als belegt markiert, alle anderen Zellen als frei. Die Zellen sind nicht deckungsgleich mit den Hindernissen, daher wird der Freiraum nicht vollständig abgebildet. Ausgehend von der Zelle, die die Startpose enthält, wird als Pfad eine Folge benachbarter freier Zellen gesucht, die zur Zelle führt, die der Zielpose entspricht.

Zusammenhängende freie Zellen können im 2D-Raum mit dem Prinzip des Quadtrees komprimiert abgebildet werden. Wenn neben freien und belegten Zellen zusätzlich zwischen

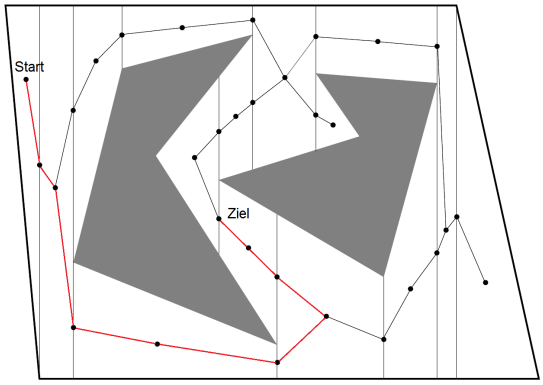


Abbildung 2.7: Exakte Zellzerlegung: Die schwarzen Kreise bilden die Knoten des Graphen. Sie basieren auf den Zentren der Teilmengen des Freiraumes und der Trennlinien zwischen den Teilmengen. In der Menge der schwarz markierten Kanten zwischen benachbarten Knoten wird ein Pfad von der Start- zur Zielkonfiguration gesucht. Der geplante Pfad setzt sich aus den rot markierten Kanten zusammen.
Quelle: [vgl. LaV06, S. 256]

teilweise belegten Zellen unterschieden wird, kann die Suche mit verschiedenen Auflösungen durchgeführt werden. Wird bei einer ersten Suche mit grober Auflösung kein Pfad gefunden, können teilweise belegte Zellen feiner zerlegt werden. Die resultierenden Zellen repräsentieren den real vorhandenen Freiraum mit höherer Genauigkeit. In der verfeinerten Struktur kann rekursiv nach einem Pfad gesucht werden, bis keine teilweise belegten Zellen vorhanden sind oder eine bestimmte Rekursionstiefe erreicht wurde [Gut94, S. 37]. Das Prinzip ist in Abbildung 2.8 visualisiert. Die approximative Zellzerlegung ist nicht vollständig, da nicht der gesamte Freiraum abgebildet wird. Dem gegenüber steht der geringere Rechenaufwand im Vergleich zur exakten Zellzerlegung.

2.5.3 Potentialfeld

Bei Potentialfeldmethoden [Kha86] wird ein virtuelles Potentialfeld im Raum aufgespannt, aus dem sich an jedem Punkt im Raum die Richtung zum Ziel berechnen lässt. Das Potentialfeld setzt sich aus einem anziehenden Potential und mehreren abstoßenden Potentialen zusammen. Das anziehende Potential geht von der Zielpose aus und weist mit zunehmender Entfernung einer Pose im Raum zur Zielpose höhere Werte auf. Abstoßende Potentiale erzeugen an Hindernissen und ihrer unmittelbaren Umgebung hohe Werte. Alle Potentiale werden in eine Potentialfunktion integriert. Die Konstruktion eines Potentialfeldes ist in Abbildung 2.9 illustriert. Die Wegplanung differenziert diese Potentialfunktion während der Fahrt an jeder aktuellen Konfiguration des Roboters und kann z.B. Gradientenabstieg nutzen, um die nächste Konfiguration zu ermitteln.

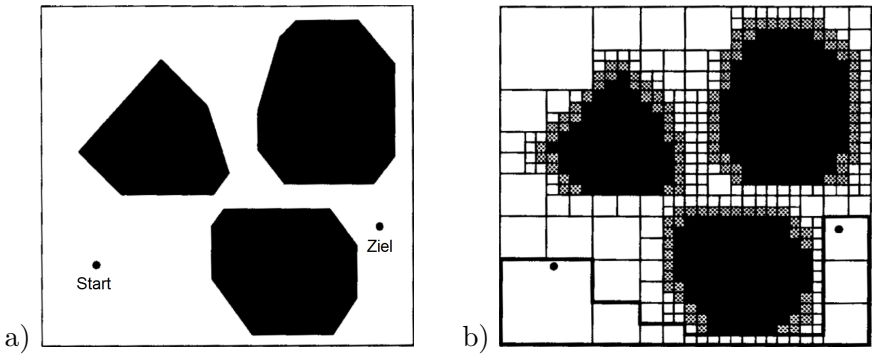


Abbildung 2.8: Approximative Zellzerlegung: In a) ist ein zweidimensionaler Raum mit einer Start- und einer Zielkonfiguration dargestellt. Die weiße Fläche repräsentiert Freiraum, Hindernisse sind schwarz visualisiert. Der Raum wird in b) in unterschiedlich große Zellen zerlegt. Einige der grau dargestellten, teilweise belegten Zellen wurden stufenweise mit einer höheren Auflösung zerlegt. Die schwarz umrandeten Zellen bilden einen gültigen Pfad, da sie die Start- und Zielkonfiguration durch zusammenhängenden Freiraum verbinden.
Quelle: [vgl. Lat91, S. 18]

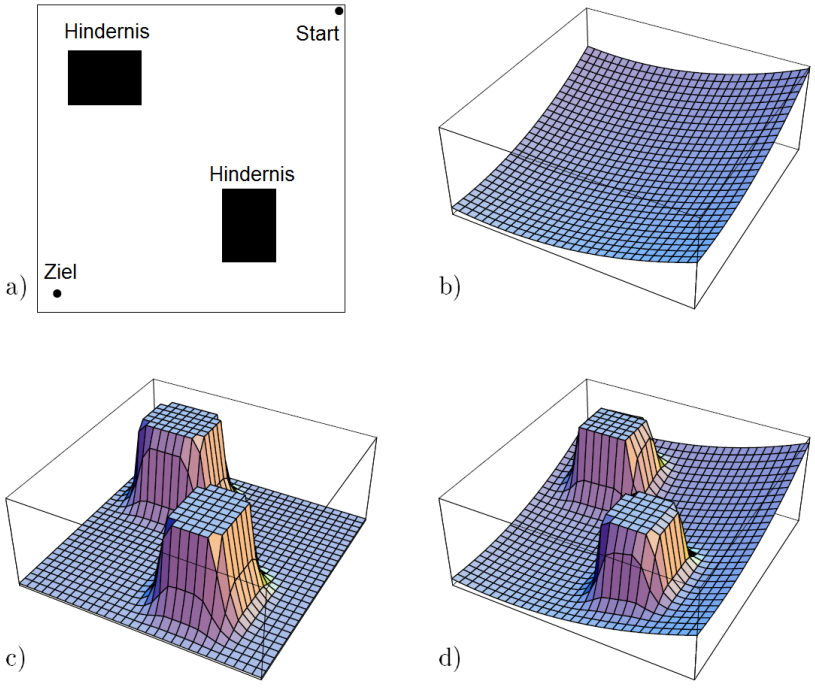


Abbildung 2.9: Konstruktion eines Potentialfeldes: Teilabbildung a) zeigt die Ausgangssituation inkl. Hindernisse sowie Start- und Zielkonfigurationen. In b) ist das anziehende Potential des Ziels dargestellt. Teilabbildung c) zeigt die abstoßenden Potentiale der Hindernisse. In d) werden alle Potentiale zusammengefasst.
Quelle: [vgl. Bro00, S. 17]

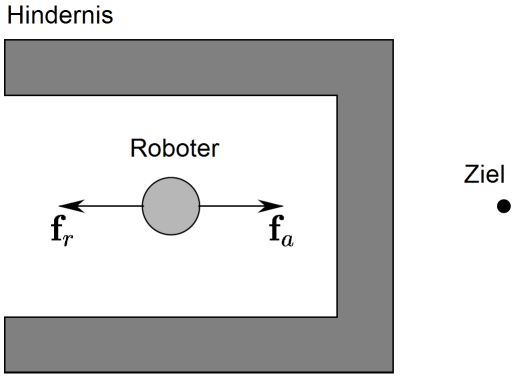


Abbildung 2.10: Lokales Minimum: Der Roboter ist teilweise von einem Hindernis umgeben. Das anziehende Potential des Ziels und die abstoßenden Potentiale des Hindernisses gleichen sich aus. Der Roboter kann sich nicht mehr bewegen, er befindet sich in einem lokalen Minimum.
Quelle: [vgl. Bro00, S. 18]

Die beschriebene Potentialfeldmethode ist nicht vollständig. Gleichen sich an einer Pose ungleich der Zielpose anziehende und abstoßende Potentiale aus, befindet sich das Verfahren in einem lokalen Minimum und die Pose kann nicht verlassen werden. Eine solche Situation ist in Abbildung 2.10 dargestellt. Zur Überwindung lokaler Minima existieren verschiedene Ansätze, die jedoch Einschränkungen und einen erhöhten Rechenaufwand mit sich bringen [BL90; Kod87].

Im Gegensatz zu den bisher vorgestellten Pfadplanungsverfahren wird der Pfad nicht zuvor vollständig erstellt, sondern ist implizit repräsentiert und wird dynamisch berechnet [Bro00, S. 16]. Dies geschieht effizient durch Auswertung der integrierten Potentialfunktion. Dem Vorteil des geringen Rechenaufwands steht das Problem der lokalen Minima gegenüber.

3 Pfadplanung in dynamischer Umgebung

Bei der Beschreibung der grundlegenden Pfadplanungsverfahren in Kapitel 2 wurde implizit angenommen, dass die Umgebung vollständig bekannt ist. Wie in Kapitel 1 erwähnt, kann dies in vielen Anwendungsfällen nicht vorausgesetzt werden. Unvollständige Informationen über die Umgebung, sich bewegende und unbekannte Hindernisse haben einen wesentlichen Einfluss auf die Pfadplanung. In Abschnitt 3.1 wird zunächst der Begriff der dynamischen Umgebung definiert. Das Problem der Pfadplanung für eine solche Umgebung wird in Abschnitt 3.2 in zwei Teilprobleme aufgegliedert, der globalen und der lokalen Pfadplanung. Die Integration beider Teilprobleme in ein Pfadplanungsverfahren ist Thema von Abschnitt 3.3. In dieser Arbeit wird die lokale Pfadplanung fokussiert, auf die in Abschnitt 3.4 genauer eingegangen wird. Das Elastic-Band-Pfadplanungsverfahren und zwei Erweiterungen werden in Abschnitt 3.5 im Detail erläutert. Abschließend wird in Abschnitt 3.6 das Pfadplanungsproblem einer dreidimensionalen dynamischen Umgebung diskutiert.

3.1 Dynamische Umgebung

Pfadplanungsverfahren können in einer statischen oder einer dynamischen Umgebung agieren. In einer statischen Umgebung sind die Posen aller Hindernisse bekannt [KJIF06, S. 538]. Es gilt die Annahme, dass keine sich bewegenden Hindernisse vorhanden sind. Umgebungsrepräsentationen können vor der Laufzeit aufgebaut und müssen während der Laufzeit nicht aktualisiert werden.

In einer dynamischen Umgebung können sich bewegende Hindernisse vorhanden sein. Unterschieden wird, ob die Bewegungsabläufe der Hindernisse bekannt oder unbekannt sind. Im letzteren Fall wird die Umgebung als unbekannte dynamische Umgebung bezeichnet [Kru98, S. 16]. Insbesondere in einer solchen Umgebung muss das Pfadplanungsverfahren den aktuellen Zustand der Umgebung wahrnehmen und berücksichtigen. Eine unbekannte dynamische Umgebung liegt meist im Bereich der autonomen mobilen Roboter vor [Kru98, S. 16] und wird in dieser Arbeit als gegeben angenommen.

3.2 Globale und lokale Pfadplanung

Eine naive Möglichkeit, Pfadplanung in dynamischen Umgebungen zu realisieren, ist stetige Neuplanung. Änderungen in der Umgebung werden über die Sensoren erfasst und die Umgebungsrepräsentation aktualisiert. Das Pfadplanungsverfahren erstellt anschließend anhand der aktualisierten Daten einen neuen Pfad. In der Praxis ist dieses Vorgehen nicht realisierbar, da das Planen eines Pfades zu rechenintensiv ist [FBT97, S. 3f.]. Um Kollisionsfreiheit gewährleisten zu können, muss die Umgebungsrepräsentation häufiger aktualisiert werden, als ein Pfad komplett neu geplant werden kann.

Um diesem Problem zu begegnen, kann die Pfadplanung in zwei Teilprobleme aufgespalten werden. Globale Pfadplanung umfasst die gesamte, für den Roboter relevante Umgebung und plant für diese einen globalen Pfad [FBT97, S. 3]. Dabei werden vorrangig statische Informationen über unveränderliche Hindernisse verwendet, z. B. aus abstrakten Karten wie Raumplänen. Dadurch kann globale Pfadplanung komplett oder teilweise vor der Ausführung der Roboterbewegung durchgeführt werden [Bro00, S. 20].

Lokale Pfadplanungsverfahren agieren nur auf einem Ausschnitt der Umgebung, in dem sich der Roboter aktuell befindet bzw. nur mit den aktuellen Sensordaten [FBT97, S. 4]. Sie reagieren auf die Änderungen der Umgebung und realisieren somit reaktive Pfadplanung. Da sie in einer begrenzten Umgebung agieren, sind lokale Pfadplanungsverfahren anfällig für lokale Minima, der Rechenaufwand ist aber geringer als bei globaler Pfadplanung. Lokale Pfadplanung kann daher gekoppelt mit der Ausführung der Roboterbewegung durchgeführt werden. Durch den geringeren Rechenaufwand als bei der globalen Pfadplanung, können lokale Pfadplanungsverfahren in Echtzeit auf eine dynamische Umgebung reagieren. Echtzeitfähigkeit im Kontext der reaktiven Pfadplanung fordert eine rechtzeitige Reaktion auf die dynamische Umgebung, um Kollisionen zu vermeiden [Hop92, S. 39f.].

Die in Abschnitt 2.5 vorgestellten Pfadplanungsverfahren bilden die Grundlage für aktuell eingesetzte Verfahren. Roadmap- und Zellzerlegungsverfahren sind globale Pfadplanungsverfahren. Potentialfeldmethoden können global und lokal eingesetzt werden. Anziehende und abstoßende Potentiale können unter Nutzung statischer Informationen für die gesamte Umgebung oder mithilfe der aktuellen Sensordaten für einen Ausschnitt erstellt werden. Durch Erweiterungen können sie lokale Minima vermeiden und global agieren, wodurch die Komplexität steigt. Alternativ zu den Begriffen globaler und lokaler Pfadplanung wird in der Literatur teilweise zwischen Pfadplanung und Hindernisvermeidung unterschieden.

3.3 Integration globaler und lokaler Pfadplanung

Durch die Trennung zwischen globaler und lokaler Pfadplanung kann zunächst ein globaler Pfad von der Start- zur Zielpose erstellt werden, der dynamisch an die aktuelle Umgebung angepasst wird. Um dies zu realisieren, müssen beide Bestandteile in ein Gesamtsystem integriert werden. Der Aufbau des Gesamtsystems wird durch die Steuerarchitektur des Roboters definiert. Eine Steuerarchitektur beschreibt das grundlegende Paradigma zur Steuerung des Roboters. Sie definiert ein Modell, das den Ablauf der Steuerung beschreibt. Das Modell umfasst neben der Pfadplanung andere Komponenten, z. B. eine Repräsentation der Umgebung oder eine Komponente zur Vorverarbeitung von Sensordaten. Die Steuerung eines Roboters definiert Reaktionen auf eingehende Sensordaten, die in Ausgaben für Aktoren resultieren, um einen gewünschten Zustand des Roboters zu erreichen [vgl. MM08, S. 892].

Eine deliberative Steuerarchitektur zeichnet sich durch einen Zyklus aus Wahrnehmen, Planen und Handeln (sense, plan, act) aus [KS08, S. 189]. Sie wird durch in hierarchischen Schichten organisierte Komponenten realisiert, die Teilziele bearbeiten [Neh02, S. 14]. Diese Steuerarchitektur realisiert globale Pfadplanung. Aus den Sensordaten wird zur Repräsentation der Umgebung ein Weltmodell erstellt, auf dessen Grundlage ein globaler Pfad geplant wird. Dem Pfad entsprechende Steuerbefehle werden generiert und ausgeführt, ohne auf potenzielle Änderungen der Umgebung zu achten.

Eine reaktive Steuerarchitektur verfolgt das Paradigma der schnellen Reaktion auf Sensordaten durch direkte Kopplung von Sensoren und Aktoren [Bro86; Bek05, S. 104f.]. Hindernisvermeidung wird in einer Steuerschleife mit hoher Frequenz durchgeführt. Ein Weltmodell und globale Pfadplanung sind nicht vorhanden, daher können lokale Minima auftreten. Abbildung 3.1 stellt die deliberative und die reaktive Steuerarchitektur gegenüber.

Eine hybride Steuerarchitektur kombiniert Eigenschaften der deliberativen und der reaktiven Steuerarchitektur. Ein Weltmodell wird zur globalen Pfadplanung verwendet. Statt diesen Plan direkt auszuführen, wird eine Steuerschleife verwendet. Diese nutzt den initial erstellten globalen Pfad und setzt lokale Pfadplanung um. Dadurch werden die Planung eines global optimalen Pfades und gleichzeitig schnelle Reaktionen auf Änderungen der Umgebung ermöglicht. Unsicherheit kann im Weltmodell repräsentiert werden, das durch die reaktive Komponente hochfrequent aktualisiert wird. Als Beispiele zur Umsetzung sind die 3-T-Architektur [Gat92; RK03] oder der orthogonale Ansatz [Hop92; Bek05; TMD+06; DSL03] zu nennen. In Abbildung 3.2 ist der prinzipielle Aufbau einer hybriden Steuerarchitektur dargestellt.

In [Nou97] wird die Integration von lokaler und globaler Pfadplanung durch alternierende

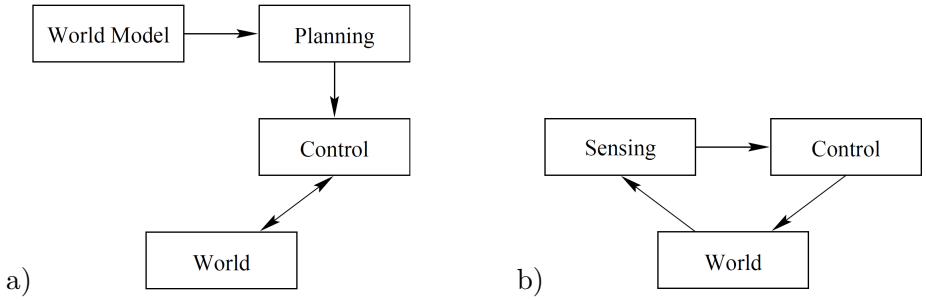


Abbildung 3.1: Vergleich von Steuerarchitekturen: Bei der deliberativen Steuerarchitektur in a) wird ein globaler Pfad anhand des Weltmodells erstellt. Änderungen in der Umgebung werden erst bei Aktualisierung des Weltmodells erfasst und bei Neuplanung des globalen Pfades berücksichtigt. Bei der reaktiven Steuerarchitektur in b) haben Änderungen in der Umgebung direkt Einfluss auf die Ausführung der Steuerbefehle. Da nur die Sensordaten der lokalen Umgebung verwendet werden, ist keine globale Pfadplanung möglich.
Quelle: [Bro00, S. 22]

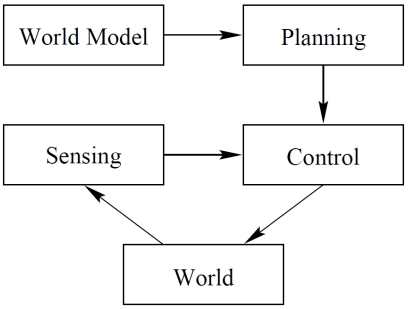


Abbildung 3.2: Hybride Steuerarchitektur: Wie bei der deliberativen Steuerarchitektur wird auf Grundlage des Weltmodells ein globaler Pfad geplant. Darunter liegende Steuerschichten haben wie bei der reaktiven Steuerarchitektur Zugriff auf aktuelle Sensordaten und können den Pfad bzw. daraus erstellte Steuerbefehle selbstständig modifizieren.
Quelle: [Bro00, S. 24]

Phasen aus Planung und Ausführung betrachtet. Der Suchraum für einen Pfadplan wird durch Aufteilung in Teilpläne, Überführung in abstraktere Pläne oder durch vereinfachende Annahmen reduziert. [Kru98] gibt einen Überblick über statistische Pfadplanung. Dabei werden statistische Daten über die Bewegungen von Hindernissen und über Kollisionswahrscheinlichkeiten automatisch gesammelt, mit denen die globale Pfadplanung genauere Pfadpläne erstellen kann. Rechenintensive Neuplanungen auf globaler Ebene und Ausweichmanöver auf lokaler Ebene können damit teilweise vermieden werden.

3.4 Lokale Pfadplanung

Lokale Pfadplanungsverfahren generieren Pfade für einen begrenzten Ausschnitt der Umgebung abhängig von der aktuellen Pose des Roboters. Das primäre Ziel besteht darin, Hindernisse in einer dynamischen Umgebung zu vermeiden. Der Aufgabenumfang der verschiedenen Verfahren variiert und reicht vom einfachen Ausweichen über Modifizierung eines globalen Pfades bis zur selbstständigen Generierung von lokalen Pfaden. In den nachfolgenden Abschnitten werden einige populäre Ansätze vorgestellt. [KJIF06, S. 539f.; SK08, S. 839ff.] geben einen Überblick über weitere lokale Pfadplanungsverfahren.

3.4.1 Dynamic Window Approach

Der Dynamic Window Approach [FBT96; FBT97] stellt bei der Pfadplanung die Dynamik des Roboters in den Vordergrund. Das Pfadplanungsverfahren arbeitet in einem zweidimensionalen Suchraum, bestehend aus eindeutigen Paaren von Translations- und Rotationsgeschwindigkeit des Roboters. Der Suchraum wird zunächst auf Geschwindigkeitspaare beschränkt, die dem Roboter ermöglichen, vor Hindernissen anzuhalten. Anschließend folgt eine zweite Einschränkung auf Geschwindigkeitspaare, die innerhalb eines Zeitintervalls erreicht werden können. Im reduzierten Suchraum wird eine Zielfunktion maximiert. Ziel ist es, den Winkel zwischen Fahrtrichtung und Zielpose zu minimieren, sowie den Abstand zu Hindernissen und die Geschwindigkeit in Fahrtrichtung zu maximieren.

Der Dynamic Window Approach ermöglicht kollisionsfreie Navigation in einer dynamischen Umgebung mit unbekannten, sich bewegenden Hindernissen bei hohen Geschwindigkeiten [FBT96, S. 8; SK08, S. 843]. Das ursprüngliche, für einen nicht-holonomen Roboter entwickelte Pfadplanungsverfahren wurde in [Bro00, S. 44ff.] für holonome Roboter erweitert.

3.4.2 Nearness Diagram

Beim Nearness-Diagram-Verfahren [MM00; MM04] werden anhand der lokalen Umgebung zwei Diagramme berechnet, die Diskontinuitäten in der Verteilung von freien und belegten Bereichen repräsentieren. Das PND (Nearness Diagram from the central point) beschreibt die Abstände von Hindernissen zum Zentrum des Roboters; das RND (Nearness Diagram from the robot) beschreibt die Abstände von Hindernissen zum äußeren Umfang des Roboters [MM00, S. 2095]. Aus dem PND werden Sektoren als Teilmengen des Freiraumes identifiziert. Der Sektor, der die Zielpose enthält, wird als Zielsektor markiert. Aus dem RND wird die Sicherheit der aktuellen Pose kategorisiert.

Im nächsten Schritt wird eine Situation aus der Menge von drei Situationen mit hoher und zwei Situationen mit geringer Sicherheit ausgewählt. Geringe Sicherheit liegt z. B. vor, wenn sich ein Hindernis auf einer Seite des aktuellen Sektors im Sicherheitsbereich des Roboters befindet oder wenn der Roboter auf beiden Seiten von nahen Hindernissen umgeben ist. Für jede Situation ist eine einfache Navigationsvorschrift definiert, mit der die Bewegung des Roboters in Zielrichtung berechnet wird. Nach [MM00, S. 2100] ist das Nearness-Diagram-Verfahren vor allem in hochgradig dynamischen, unstrukturierten Umgebungen von Vorteil und zeichnet sich durch geringen Aufwand zur Parametrisierung aus.

3.4.3 Virtual Force Field

Das Virtual-Force-Field-Verfahren [BK89] kombiniert Potentialfelder mit einer Gitterkarte zur Abbildung von Belegungswahrscheinlichkeiten der Gitterzellen durch Hindernisse. Die Gitterkarte bildet die gesamte Umgebung ab, wobei ein Ausschnitt fester Größe mit der aktuellen Roboterpose als Zentrum kontinuierlich aktualisiert wird. Signalisieren die Sensordaten zu einem Zeitpunkt ein Hindernis in einer Gitterzelle, wird der Zähler der Gitterzelle erhöht und die Wahrscheinlichkeit einer realen Belegung steigt. Durch die fortlaufende Aktualisierung wird der Einfluss von fehlerhaften Sensordaten verringert.

Belegte Gitterzellen erzeugen eine abstoßende Kraft auf den Roboter, die mit abnehmender Distanz zum Roboter und zunehmender Wahrscheinlichkeit der Belegung steigt. Eine anziehende Kraft mit konstanter Intensität wirkt von der Zielpose auf den Roboter. Alle Kräfte werden aufsummiert. Die Richtung der resultierenden Gesamtkraft bestimmt die Fahrtrichtung des Roboters. Das Virtual-Force-Field-Verfahren eignet sich zur Integration unterschiedlicher Sensoren unter expliziter Berücksichtigung von Unsicherheit der Sensordaten [BK89, S. 1181].

3.4.4 Vector Field Histogram

Das Vector-Field-Histogram-Verfahren [BK90; BK91] ist eine Erweiterung des Virtual-Force-Field-Verfahrens. Problematisches Verhalten, wie das Oszillieren des Pfades bei der Durchquerung eines schmalen Korridors, wird durch einen zusätzlichen Zwischenschritt bei der Berechnung der Belegungswahrscheinlichkeit einer Gitterzelle vermieden. Die Sensordaten des kontinuierlich aktualisierten Ausschnitts des Gitters werden in ein Polarkoordinatensystem übertragen.

Das Polarkoordinatensystem teilt aktuell erreichbare Fahrtwinkel in Sektoren ein. Aus dem Polarkoordinatensystem wird ein Histogramm erzeugt, das die Belegungswahrscheinlichkei-

ten der einzelnen Sektoren beschreibt. Sektoren, deren Belegungswahrscheinlichkeit unter einem definierten Schwellwert liegt, sind Kandidaten für den anzusteuernden Fahrtwinkel. Aus den Kandidaten wird der beste Sektor gewählt. Dies erfolgt anhand der Abweichung des aktuellen Fahrtwinkels vom Fahrtwinkel zur Zielpose, der Nähe zu Hindernissen und einer Mindestanzahl an zusammenhängenden Sektoren, d. h. der Größe des repräsentierten Freiraumes. Das Vector-Field-Histogram-Verfahren ist in unstrukturierten Umgebungen robuster als das Virtual-Force-Field-Verfahren und berücksichtigt explizit Unsicherheit bei der Modellierung der Umgebung.

3.4.5 Elastic Band

Elastic Band ist ein lokales Pfadplanungsverfahren, das einen gegebenen globalen Pfad modifiziert und dabei an Änderungen in der dynamischen Umgebung anpasst. In Analogie zu einem elastischen Band, das zwischen Start- und Zielpose gespannt ist, wirken zwei Kräfte auf den Pfad und deformieren ihn. Das Verfahren und zwei Erweiterungen werden im folgenden Abschnitt 3.5.1 erläutert.

3.5 Elastic Band Framework

Das Elastic Band Framework ist ein Framework zur Steuerung eines Roboters, das globale Pfadplanung mit lokaler Pfadplanung verbindet. Ein globaler Pfad wird erstellt und vom Elastic-Band-Verfahren in Echtzeit modifiziert. Das Elastic Band Framework setzt sich aus einem globalen und einem lokalen Pfadplanungsverfahren zusammen sowie weiteren Komponenten wie einem Weltmodell inkl. einer Umgebungsrepräsentation und Komponenten zur Ausführung eines erstellten Pfades. Das Elastic Band Framework kann als hybride Steuerarchitektur aufgefasst werden. In Abbildung 3.3 ist die hybride Steuerarchitektur dem strukturellen Aufbau des Elastic Band Frameworks gegenüber gestellt. Im Folgenden werden das Elastic-Band-Verfahren als lokales Pfadplanungsverfahren und seine Erweiterungen erläutert.

3.5.1 Elastic Band

Das Elastic-Band-Verfahren (elastisches Band) [QK93; Qui94] agiert nach dem Prinzip eines elastisches Bandes, z. B. eines Gummibandes, das zwischen der aktuellen Roboterpose und der Zielpose gespannt wird. Es folgt dabei initial dem globalen Pfad, passt ihn kontinuierlich an und führt um Hindernisse herum. Die interne Spannung des Bandes sorgt für einen

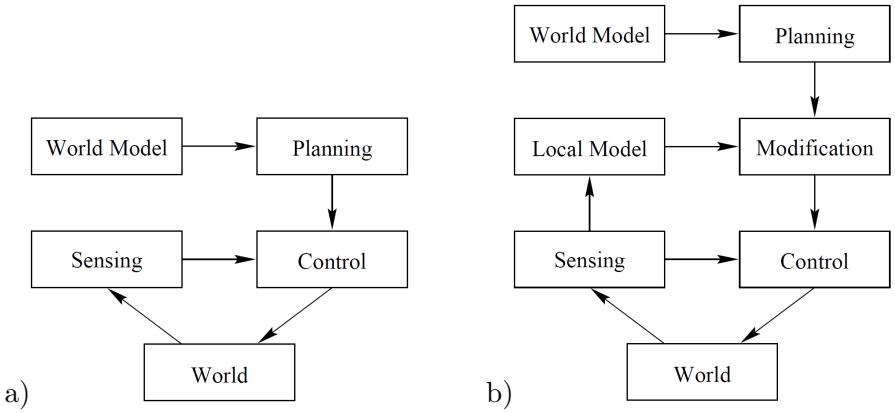


Abbildung 3.3: Steuerarchitektur des Elastic Band Framework: Teilabbildung a) zeigt die hybride Steuerarchitektur. Wie in b) zu erkennen, setzt das Elastic Band Framework diese Steuerarchitektur um. Der globale Pfad wird anhand des Weltmodells initial geplant. Durch ein Modell der lokalen Umgebung, wie z. B. eine Karte, wird der initiale Plan kontinuierlich modifiziert. Auf Basis des modifizierten, lokalen Plans werden Steuerbefehle für den Roboter erstellt. Quellen: [Bro00, S. 24, S. 77]

kurzen Pfad zur Zielpose. Hindernisse deformieren das Band, sodass Kollisionen vermieden werden. Dies wird durch externe Kräfte modelliert, die ausgehend von den Hindernissen abstoßend auf das Band einwirken. Ändern Hindernisse ihre Pose oder treten neue Hindernisse auf, deformieren die externen Kräfte das Band und ändern seine Lage. Dadurch reagiert das Verfahren auf Änderungen in der Umgebung, ohne dass eine Neuplanung des Pfades notwendig wird.

Ein elastisches Band repräsentiert den relevanten Ausschnitt des kollisionsfreien Bereichs im Konfigurationsraum des Roboters. Anstatt den gesamten Freiraum zu modellieren, werden dynamisch lokale Teilbereiche berechnet [QK93, S. 3]. Die Teilbereiche werden durch konvexe geometrische Strukturen modelliert, sogenannte Blasen (Bubbles). Meist werden n-dimensionale Sphären verwendet, wobei n der Anzahl der Dimensionen des Raumes entspricht, in dem geplant wird. Für den zweidimensionalen Raum werden z. B. Kreise verwendet, für den dreidimensionalen Raum Kugeln. Abbildung 3.4 illustriert ein elastisches Band in einem 2D-Konfigurationsraum.

Jede Blase repräsentiert eine Teilmenge des Freiraumes um den Roboter in einer spezifischen Konfiguration. Das Zentrum einer Blase bildet die angenommene Konfiguration des Roboters auf dem Pfad. Die Größe einer Blase wird so gewählt, dass jede Konfiguration innerhalb kollisionsfrei ist. Eine Blase b ist somit als eine Teilmenge des Freiraumes B um eine Konfiguration $q_b \in C$ definiert:

$$b = B(q_b) = \{q : D(q_b - q) < d(q_b)\}, \tag{3.1}$$

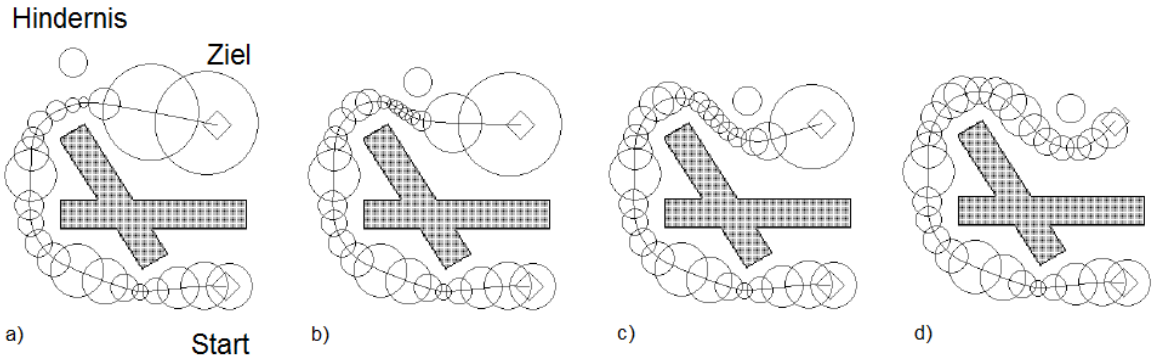


Abbildung 3.4: Elastisches Band: Ein Pfad führt in a) von einer Start- zu einer Zielpose um ein statisches Hindernis herum. Die überlappenden Blasen bilden eine Teilmenge des Freiraumes um den Roboters entlang des Pfades ab. Durch ein sich bewegendes Hindernis wird der Pfad in b) bis d) deformiert. Die Größen und die Anzahl der Blasen im betroffenen Abschnitt des Pfades ändern sich dabei.
Quelle: [vgl. Qui94, S. 66]

mit $q \in C$. Die Funktion $d : C \to \mathbb{R}$ berechnet die minimale Distanz zwischen einer Konfiguration und den Hindernissen der Umgebung. Die Funktion $D : C \to \mathbb{R}$ definiert die maximale Distanz, die ein Roboter zurücklegen kann, um von einer Konfiguration q zu einer Konfiguration $q^{'}$zu gelangen. Für einen Roboter mit zwei translatorischen Freiheitsgraden entlang der x- und y-Achse sowie einem rotatorischem Freiheitsgrad um die z-Achse gilt:

$$D(\triangle q) = \sqrt{\triangle q_x^2 + \triangle q_y^2} + r_{max}|\triangle q_\theta|, \tag{3.2}$$

mit $q = (x, y, \theta)$, $\triangle q = q - q^{'}$ und r_{max} als einer Konstanten, die den maximalen Abstand des Zentrums des Roboters zu jedem anderen Punkt des Roboters angibt. [QK93, S. 3f.]

Da jede Konfiguration innerhalb einer Blase ohne Kollision erreicht werden kann, ist zwischen zwei Blasen eine kollisionsfreie Verbindung vorhanden, wenn die Blasen überlappen [QK93, S. 4]. Wenn eine geordnete Liste von sich paarweise überlappenden Blasen erstellt werden kann, die die Start- und Zielposen beinhalten, existiert ein kollisionsfreier Pfad zwischen Start und Ziel. Ein elastisches Band $e = \{b_0, b_1, \ldots, b_n\}$ ist eine solche Liste, deren sich überlappende Blasen b_i an Zwischenpunkten entlang des globalen Pfades erstellt werden.

Im Zuge der Deformation des Bandes ändern sich Pose und Größe der Blasen. Um stets eine Überlappung zu gewährleisten, müssen dem Band neue Blasen hinzugefügt werden können. Blasen werden entfernt, sobald sie redundant werden, um unnötige Berechnungen zu vermeiden. Eine Blase ist redundant, sobald sich ihr Vorgänger und ihr Nachfolger überlappen. Dies kann geschehen, wenn sich z. B. ein dynamisches Hindernis vom Band

wegbewegt und eine benachbarte Blase eine größere Teilmenge des Freiraumes repräsentieren kann und dadurch vergrößert wird.

Die Blasen werden zwei Kräften ausgesetzt. Die interne Kraft bewirkt eine Kontraktion des Bandes, die externe Kraft verhindert eine Kollision mit Hindernissen. Die interne Kraft f_c für eine Blase b_i an Position i im Band wird wie folgt ermittelt:

$$f_c = k_c \left(\frac{b_{i-1} - b_i}{|b_{i-1} - b_i|} + \frac{b_{i+1} - b_i}{|b_{i+1} - b_i|} \right), \tag{3.3}$$

mit k_c als einen Gewichtungsparameter [QK93, S. 5]. Berechnungen mit einer Blase b werden mit der Konfiguration ihres Zentrums q_b ausgeführt. Die externe Kraft f_e für b kann durch eine Potentialfunktion ermittelt werden. In [Qui94, S. 62] wird diese Potentialfunktion $V_e(b)$ folgendermaßen definiert:

$$V_e(b) = \begin{cases} \frac{1}{2} k_e \left(d_0 - d(b) \right)^2 & \text{wenn } d(b) < d_0 \\ 0 & \text{sonst} \end{cases}, \tag{3.4}$$

wobei k_e einen Gewichtungsparameter bezeichnet, $d(b)$ die minimale Distanz zwischen b und den Hindernissen berechnet und d_0 die maximale Distanz angibt, bis zu der abstoßende Kräfte wirken sollen. Die Gesamtkraft f_g für b ergibt sich durch Addieren der externen und internen Kraft:

$$f_g = f_c + f_e. \tag{3.5}$$

Das Hinzufügen und Entfernen von Blasen kann zu Instabilitäten im Band führen. Wenn eine Blase wiederholt an einer Pose im Band hinzugefügt, durch die Optimierung verschoben und an einer anderen Pose entfernt wird, kann das Band oszillieren. Das Problem wird durch Eliminierung der tangentialen Anteile der Kräfte verhindert. Die resultierende Kraft f_r ergibt sich wie folgt [QK93, S. 5]:

$$f_r = f_g - \frac{f_g \cdot (b_{i-1} - b_{i+1})(b_{i-1} - b_{i+1})}{|b_{i-1} - b_i|^2}. \tag{3.6}$$

Durch die einwirkenden Kräfte wird der vorgegebene Pfad dynamisch an die Umgebung angepasst. Dadurch können lokale Minima vermieden werden, ohne dass eine Neuplanung des globalen Plans erforderlich wird. Bei der Deformation wird die Topologie des globalen Plans nicht verändert. Dadurch können suboptimale Pfade generiert bzw. ein Pfad ungültig werden, obwohl ein veränderter gültiger Pfad existiert. Dies ist in Abbildung 3.5 demonstriert. Eine solche Situation wird durch Neuplanen des globalen Plans aufgelöst.

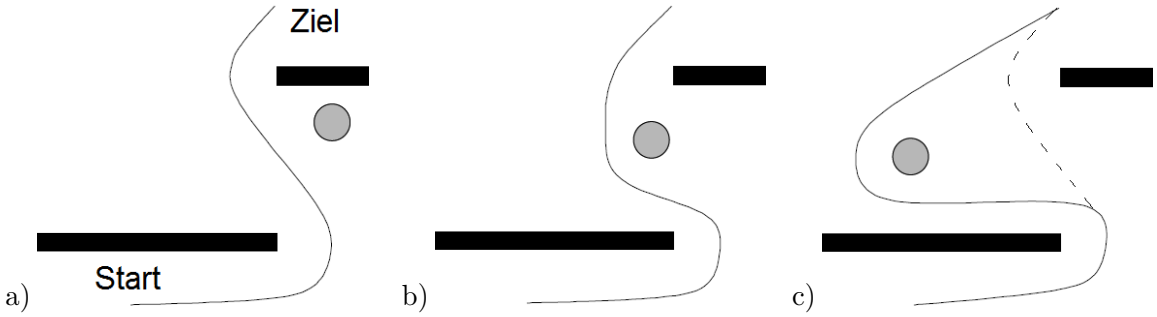


Abbildung 3.5: Suboptimaler Pfad: Der initiale Pfad in a) wird durch ein sich bewegendes Hindernis in b) deformiert. Durch die fortgesetzte Bewegung des Hindernisses wird der Pfad in c) weiter deformiert, obwohl der gestrichelte optimale Pfad gültig wäre.
Quelle: [vgl. Bro00, S. 79]

Das Elastic-Band-Verfahren wurde für verschiedene autonome mobile Roboter in 3D-Umgebungen [WSD03; LT11] und in 2D-Umgebungen [PS03] umgesetzt, darunter für Care-O-bot II [GHS04]. Das Verfahren wurde für nicht-holonome Roboter erweitert [GWS01; WG02] und für spezielle Anwendungsfelder wie autonome Fahrzeuge verwendet [KJCL97; GS00; SB05; FBB09].

3.5.2 Elastic Strip

Das Elastic Strip Framework (elastischer Streifen) [BK98a; Bro00] ist eine Erweiterung des Elastic Band Framework. Ein gegebener Pfad wird als elastischer Streifen interpretiert und mittels interner und externer Kräfte dynamisch der Umgebung angepasst. Beim Elastic-Band-Verfahren stellt das elastische Band eine eindimensionale Kurve aus Konfigurationen im Konfigurationsraum dar. Im Gegensatz dazu bildet der elastische Streifen beim Elastic-Strip-Verfahren einen mehrdimensionalen Pfad aus Posen im Arbeitsraum ab.

Durch die Pfadplanung im Arbeitsraum des Roboters wird der Einfluss der Freiheitsgrade auf das Elastic-Strip-Verfahren minimiert [BK98a, S. 3]. Der Arbeitsraum ist zwei- oder dreidimensional, unabhängig von der Anzahl der Freiheitsgrade und der damit verbundenen Dimensionalität des Konfigurationsraumes. Eine rechenintensive Transformation der erfassten Hindernisse in den Konfigurationsraum ist nicht notwendig. Weitere Objekte in der Umgebung, wie z. B. die Zielpose, werden oft im Arbeitsraum spezifiziert und müssen ebenfalls nicht transformiert werden.

Statt direkt aus Blasen, besteht der elastische Streifen s aus mehreren, den Roboter umschließenden Hüllen (Protective Hull), $s = \{h_0, h_1, \ldots, h_n\}$. Eine Hülle repräsentiert eine Teilmenge des Freiraumes um den Roboter im Arbeitsraum. Jede Hülle $h_i = \{b_{i0}, b_{i1}, \ldots, b_{im}\}$

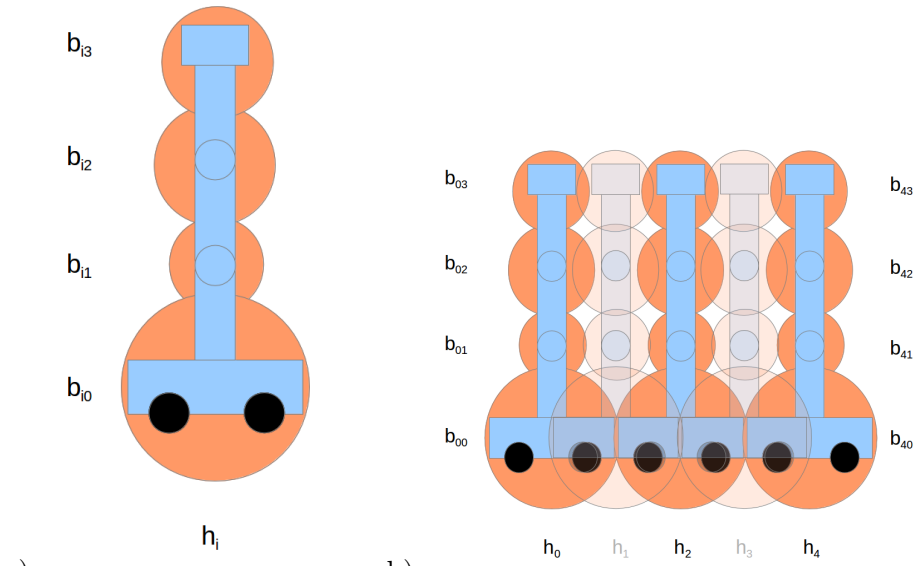


Abbildung 3.6: Elastischer Streifen: Eine einzelne zweidimensionale Hülle h_i für einen einfachen Roboter (blau), bestehend aus vier Blasen (orange) b_{ij}, ist in a) dargestellt. Der Aufbau eines Streifens, der sich aus mehreren überlappenden Hüllen zusammensetzt, wird in b) illustriert. Zur besseren Anschaulichkeit sind die Hüllen h_1 und h_3 mit Transparenz dargestellt.

setzt sich aus Blasen b_j zusammen, die jeweils eine Teilmenge des Freiraumes um einen Abschnitt des Roboters repräsentieren. Im zweidimensionalen Arbeitsraum kann eine Blase durch einen Kreis dargestellt werden, im dreidimensionalen Arbeitsraum durch eine Kugel. Abbildung 3.6 zeigt den Aufbau eines exemplarischen elastischen Streifens.

Eine einzige Blase, die den Roboter komplett umschließt, approximiert die Form des Roboters in der Regel nicht präzise. Da eine Hülle aus mehreren Blasen besteht, kann die Teilmenge des Freiraumes um den Roboter mit beliebiger Genauigkeit g repräsentiert werden, mit $0 \leq g < 1$. Abbildung 3.7 zeigt, wie der von einer Hülle eingenommene Bereich um eine rechteckige Basisfläche eines Roboters reduziert werden kann. Dazu wird die Hülle mit einer steigenden Anzahl an Blasen repräsentiert. Wird die Hülle dynamisch erzeugt, kann z. B. eine kleine Teilmenge des Freiraumes in der Nähe von Hindernissen genutzt werden, während der benötigte Rechenaufwand durch eine geringere Anzahl an Blasen sinkt, wenn der Roboter von einer großen Teilmenge des Freiraumes umgeben ist.

Physikalisch gesehen entspricht ein elastischer Streifen einer Sequenz von statischen Bindegliedern, die durch Federn miteinander verbunden sind. Jede den Roboter umschließende Hülle h_i entspricht einem Bindeglied, jede Blase b_{ij} einer Hülle entspricht einer Feder, siehe Abbildung 3.8. Die internen Kräfte auf eine Blase b_{ij} ergeben sich wie folgt aus den Distanzen

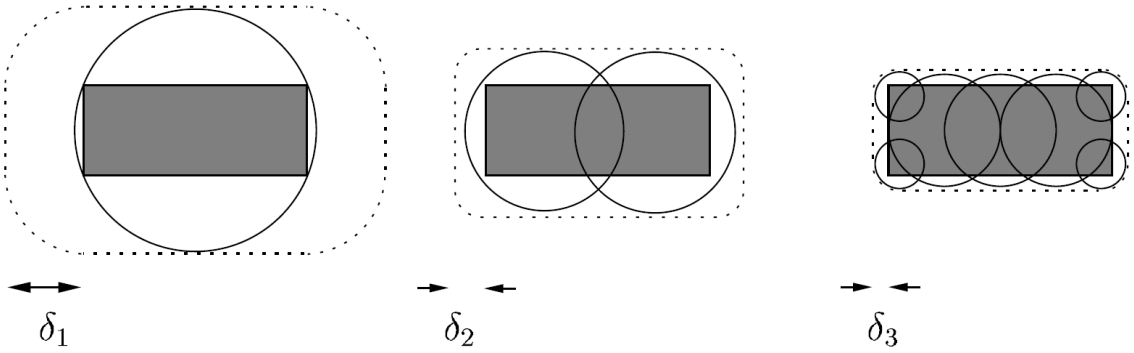


Abbildung 3.7: Approximation eines Körpers durch Blasen: Je mehr Blasen zur Repräsentation verwendet werden, desto genauer wird der Körper approximiert. Die Abstände $\delta_1$bis δ_3 geben den maximalen Abstand der Hülle vom Körper an. Der Abstand fällt stetig, sodass die Genauigkeit der Repräsentation stetig steigt.
Quelle: [Bro00, S. 132]

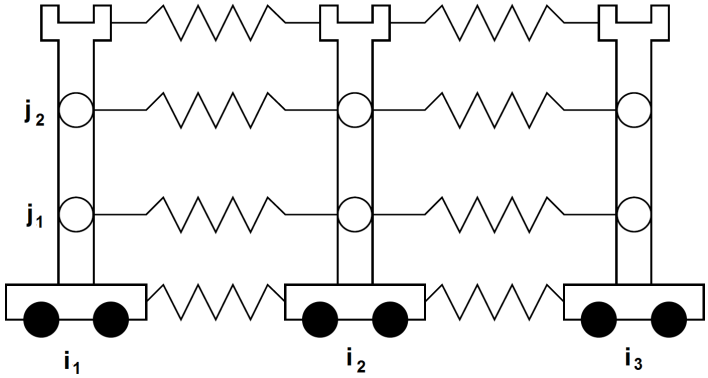


Abbildung 3.8: Elastischer Streifen: Physikalische Interpretation eines elastischen Streifens, bestehend aus Bindegliedern i, die durch Federn j verbunden sind.
Quelle: [vgl. BK98b, S. 4]

zu den Blasen b_{i-1j} und b_{i+1j} aus den benachbarten Hüllen h_{i-1} und h_{i+1}:

$$f_c = k_c \left(\frac{d_{i-1j}}{d_{i-1j} + d_{ij}} \left(b_{i+1j} - b_{i-1j} \right) - \left(b_{ij} - b_{i-1j} \right) \right), \tag{3.7}$$

mit $d_{ij} = |b_{ij} - b_{i+1j}|$ als Distanz zweier Blasen und k_c als einem Gewichtungsparameter. Die externen Kräfte auf eine Blase b_{ij} gehen wie beim Elastic-Band-Verfahren von den Hindernissen aus. Ihre Stärke ist durch den Gradienten der Potentialfunktion $V_e(b)$ in Formel 3.4 bestimmt:

$$f_e = k_e \left(d_0 - d\left(b_{ij} \right) \right) \frac{d}{|d|}, \tag{3.8}$$

wobei d den Vektor des Zentrums von b_{ij} zum nächsten Punkt des Hindernisses beschreibt, das den kleinsten Abstand zu b_{ij} einnimmt. [Bro00, S. 94ff.]

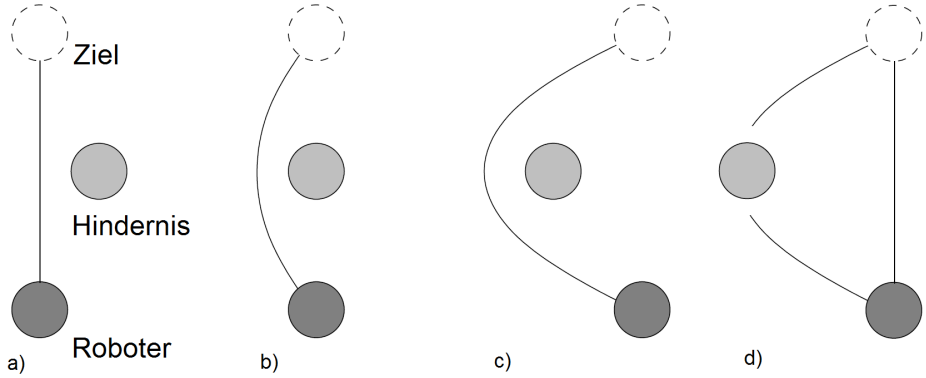


Abbildung 3.9: Suboptimalen Pfad auflösen: Das Hindernis in a) bewegt sich auf den Mittelpunkt zwischen Roboter und Ziel hinzu. Der Pfad des Roboters wird in b) angepasst, um dem Hindernis auszuweichen. Durch die fortgesetzte Bewegung des Hindernisses wird der Pfad in c) weiter verformt, obwohl der ursprünglich gerade Pfad zum Ziel erneut gültig ist. In d) wird der Pfad temporär unterbrochen und der optimale Pfad neu aufgebaut.
Quelle: [vgl. BK02, S. 1045]

Das Elastic Strip Framework ermöglicht die Integration von aufgabenspezifischem Verhalten in die dynamische Pfadanpassung [BK00, S. 5; BKV02, S. 3f]. Redundante Freiheitsgrade, z. B. durch einen Manipulator, können bei der Generierung der Gelenkgeschwindigkeiten ausgenutzt werden, um neben dem Vermeiden von Hindernissen andere Aufgaben zu verfolgen. So kann bspw. ein Manipulator parallel zur Planausführung zu einem Objekt ausgerichtet werden. Da in dieser Arbeit der Manipulator als statisch angenommen wird, wird dieser Vorteil nicht ausgenutzt. Dennoch wird somit eine einfache Erweiterung des Verfahrens um aufgabenspezifisches Verhalten oder eine Erweiterung für whole-body motion planning ermöglicht.

Wie beim Elastic-Band-Verfahren kann ein Streifen durch Änderungen in der Umgebung suboptimal oder ungültig werden. In Abbildung 3.9 ist eine quantitative Änderung in der Umgebung dargestellt, bei der der Streifen durch ein bewegliches Hindernis stark deformiert wird. Bei einer quantitativen Änderung werden die Größenverhältnisse zwischen Freiraum und belegten Bereichen verändert, die Topologie bleibt erhalten. Eine Rückkehr zum optimalen Pfad ist ohne Eingriff des globalen Pfadplanungsverfahrens möglich. Der Streifen wird temporär unterbrochen und das Hindernis kann passieren. Durch die internen Kräfte baut sich der Streifen anschließend neu auf und entspricht dem optimalen Pfad. In [Bro00, S. 113ff.] sind alternative Verfahren beschrieben.

Qualitative Änderungen in der Umgebung bewirken eine Umstrukturierung des Freiraumes, z. B. beim Öffnen einer zuvor geschlossenen Tür zu einem Raum. Sich dadurch ergebene, bessere Pfade zum Ziel werden durch das Elastic-Strip-Verfahren nicht erkannt, da es lokal agiert

[vgl. Bro00, S. 110]. Um ihnen folgen zu können, muss das globale Pfadplanungsverfahren einen neuen Pfad für das Elastic-Strip-Verfahren erstellen.

Das Elastic-Strip-Verfahren wurde für mobile Roboter mit vielen Freiheitsgraden umgesetzt, z. B. mobile Manipulatoren [BK98c] und humanoide Roboter [KYK12]. Vorteile ergeben sich bei diesen Robotern aus dem geringen Einfluss der DOF auf die Pfadplanung im Arbeitsraum. Das Elastic-Strip-Verfahren ermöglicht zudem die Integration aufgabenspezifischer Bewegungen und Kontrolle der Körperhaltung bei humanoiden Robotern [BK02, S. 1037ff.; KYB+99, S. 687].

3.5.3 Elastic Roadmap

Das Prinzip der kontinuierlichen Anpassung eines Pfades an die dynamische Umgebung lässt sich auf Pfadplanungsverfahren basierend auf Roadmaps übertragen. Statt wie beim Elastic-Band-Verfahren einen Pfad zu modifizieren, werden beim Elastic-Roadmap-Verfahren [YB06; YB10] Knoten einer Roadmap dynamisch aktualisiert. Das Elastic-Roadmap-Verfahren realisiert damit eine globale reaktive Pfadplanung. Die Roadmap bildet wie bei normalen Roadmaps den Freiraum global ab, ist allerdings im Arbeitsraum definiert. Die Repräsentation des Freiraumes ist approximativ, daher ist das Verfahren nicht vollständig. Das Elastic-Roadmap-Verfahren ist für Roboter ausgelegt, die über einen Manipulator inkl. Endeffektor verfügen [YB06, S. 1].

Die Knoten m_i der Roadmap werden Meilensteine genannt und bilden virtuelle, kollisionsfreie Posen des Roboters ab. Kanten bilden kollisionsfreie Pfade zwischen den Knoten und werden indirekt über eine geordnete Relation $C : \Phi \times M \times M \rightarrow \{true,\ false\}$ repräsentiert, mit $M = \{m_1, m_2, \ldots, m_n\}$ als einer Menge von Meilensteinen m_i. Die Menge $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ bezeichnet eine Menge von Controllern ϕ_i, die den Roboter zwischen zwei Meilensteinen bewegen können. Eine Kante existiert zwischen zwei Meilensteinen m_i und m_j, wenn die Relation $C(\phi_k, m_i, m_j)$ für ein ϕ_k *true* ergibt. Ein Controller besteht aus einer hierarchisch verschachtelten Folge von aufgabenspezifischen Verhalten V_i, die jeweils durch eine Kraft F auf den Endeffektor und eine Menge von Gelenkgeschwindigkeiten Γ definiert werden. Durch die Verschachtelung kann ein Verhalten V_i unabhängig von allen Verhalten V_j spezifiziert werden, für die gilt $i > j$. Im Gegensatz zu normalen Roadmaps wird der Graph durch Verschieben, Hinzufügen oder Entfernen von Knoten dynamisch an die Umgebung angepasst. [YB10, S. 117]

Meilensteine werden in der Nähe von Hindernissen erstellt. Dazu werden die Hindernisse durch Quader approximiert und deren Eckpunkte und Mittelpunkte von Kanten als Referenzpunkte

verwendet. Ein Meilenstein m_i wird nahe dem Hindernis erstellt und durch einen Controller ϕ_k an die Referenzpunkte angenähert. Der Controller ϕ_k behält seine Pose während der Bewegung des Roboters relativ zum Hindernis bei und passt m_i an Änderungen des Hindernisses an. Ein Meilenstein kann drei Status besitzen: ungültig, wenn er nicht kollisionsfrei ist; gültig, wenn er kollisionsfrei ist; und aufgabenkonsistent, wenn er gültig ist und zusätzlich Einschränkungen der Roboterpose durch aufgabenspezifisches Verhalten berücksichtigt.

Die Relation $C(\phi_k, m_i, m_j)$ kann auf mehrere Arten umgesetzt werden. In [YB10] wird die Relation auf die Sichtbarkeit zwischen m_i und m_j im Arbeitsraum beschränkt. Ein Pfad besteht zwischen m_i und m_j, wenn die Meilensteine durch eine Folge kollisionsfreier Geraden verbunden werden können.

Die Pfadplanung wird durch eine Graphsuche zwischen aufgabenkonsistenten Meilensteinen realisiert. Der Pfad besteht aus einer Sequenz von aufgabenkonsistenten Meilensteinen m_1 bis m_n. Die Bewegung des Roboters von einem Meilenstein m_i des Pfades zu m_{i+1} wird durch einen Controller ϕ_k realisiert, der die Relation $C(\phi_k, m_i, m_{i+1})$ erfüllt. Wird ein Pfad während der Planausführung ungültig oder tritt ein Fehler auf, wird der betroffene Meilenstein als ungültig klassifiziert und ein neuer Pfad in der Roadmap gesucht.

3.6 3D-Umgebung

In dieser Arbeit wird die lokale Pfadplanung in einer dreidimensionalen Umgebung durchgeführt. Dafür ist eine geeignete Umgebungsrepräsentation notwendig. Nachfolgend werden mit 3D-Karten und 2,5D-Karten zwei Möglichkeiten zur Umsetzung einer 3D-Umgebungsrepräsentation erläutert.

3.6.1 3D-Pfadplanung

Pfadplanung in einer 3D-Umgebung ist für autonome mobile Roboter notwendig, die sich entlang aller drei Raumdimensionen bewegen können, z. B. Laufroboter [BS11; BRSM+09], humanoide Roboter [KYK12], Unterwasserroboter [EP00] oder Flugroboter [LT11]. Eine Übersicht über solche Roboter bietet [SK08]. Roboter, deren Bewegungsmöglichkeiten sich auf zwei Raumdimensionen beschränken, sind nicht auf eine Pfadplanung in drei Dimensionen angewiesen. Die dreidimensionale Umgebung, inkl. der Hindernisse sowie der Start- und Zielpose, kann teilweise zur Vereinfachung der Pfadplanung auf eine zweidimensionale Repräsentation abgebildet werden. Bei dieser Abbildung wird der Suchraum beschränkt, was den Aufwand zur Pfadplanung reduziert, aber den Wegfall optimaler Lösungen bzw.

aller gültigen Lösungen zur Folge haben kann [BRSM+09, S. 844; HPJ+12, S. 1]. Indem die Pfadplanung die dritte Dimension berücksichtigt, wird im Austausch gegen erhöhte Komplexität die Genauigkeit der Pfadplanung gesteigert und die Lösungsqualität erhöht.

Anstatt in drei Raumdimensionen zu planen, wird in [AM12] die Zeit als dritte Dimension verwendet. Bekannte oder voraussichtliche Bewegungen dynamischer Hindernisse werden entlang der Zeitachse modelliert. Dafür kann der gesamte durchfahrene Bereich abgebildet werden, den ein Hindernis im Zuge seiner Bewegung einnimmt. Pfadplanungsverfahren für statische Umgebungen können somit für eine dynamische Umgebung verwendet werden. Dabei gilt die Einschränkung, Pfade nur in positiver Richtung entlang der Zeitachse zu planen [Kru98, S. 15]. In dieser Arbeit wird die Zeit bei der Pfadplanung nicht explizit berücksichtigt.

3.6.2 3D-Karten

Für die Pfadplanung von mobilen Robotern werden meist geometrische Karten verwendet, insbesondere Gitterkarten [FACS03, S. 2986]. Bei der lokalen Pfadplanung muss der Roboter in einem begrenzten Ausschnitt der Umgebung präzise an Hindernissen vorbei zum Ziel geleitet werden. Die höhere Genauigkeit der geometrischen Karten ist daher von Vorteil. Der Nachteil der speicherplatzintensiveren Repräsentation wird reduziert, da nur ein begrenzter Ausschnitt der Umgebung abgebildet wird. Aus diesen Gründen wird in dieser Arbeit eine geometrische Karte vorgezogen.

Zur Umsetzung geometrischer Karten für dreidimensionale Umgebungen gibt es mehrere Ansätze. Eine 3D-Punktwolke bildet Hindernisse durch eine Menge von Punkten ab. Ein Punkt repräsentiert die Koordinaten eines erfassten Hindernisses. Da Laserscanner und 3D-Kameras in der Regel selbst Punktwolken liefern, lässt sich leicht eine Karte erstellen, indem die verschiedenen Sensordaten in eine Punktwolke integriert werden. Eine 3D-Punktwolke bietet eine hohe Auflösung durch eine große Anzahl an Datenpunkten. Dadurch ist die Karte speicherintensiv und die Verarbeitung rechenintensiv [TPB06, S. 2277]. Für Planungsverfahren ist von Nachteil, dass der Freiraum nicht dargestellt wird [HWB+13, S. 3].

Bei einer 3D-Gitterkarte wird der zu kartierende Bereich in einem dreidimensionalen Gitter dargestellt. Jede quaderförmige Gitterzelle, auch als Voxel bezeichnet, ist gleich groß und kodiert den Zustand im Raum, z. B. ob der Raum frei oder durch ein Hindernis belegt ist. Im Gegensatz zu Punktwolken werden die Daten diskretisiert. Dadurch ist der benötigte Speicherbedarf unabhängig von der Anzahl der aktuell erfassten Hindernisse. Über die Größe der Voxel kann die Auflösung gesteuert werden und damit der Speicherbedarf und der

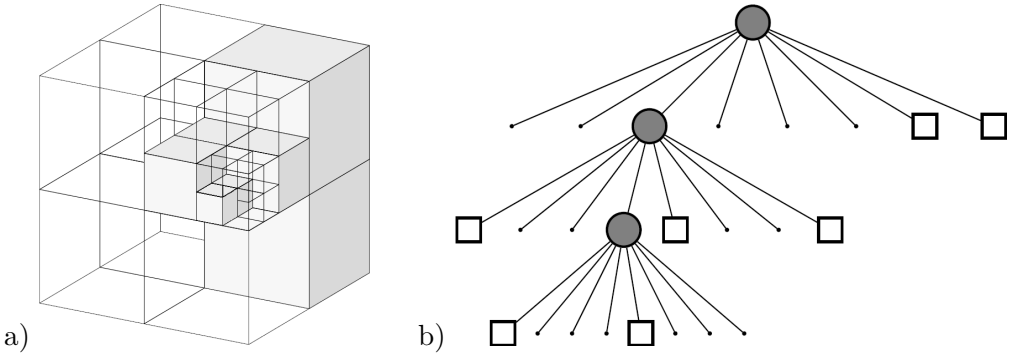


Abbildung 3.10: Octree: Teilabbildung a) zeigt einen Ausschnitt einer dreidimensionalen Umgebung, repräsentiert als Würfel. Transparente Würfel stellen eine Teilmenge des Freiraumes dar, weiß ausgefüllte Würfel Hindernisse. Der vordere obere Würfel enthält unterschiedliche Bereiche und ist daher in kleinere Würfel unterteilt. Teilabbildung b) zeigt den zugehörigen Octree in Baumstruktur. Weiße Knoten sind belegt, graue Knoten sind gemischt und punktförmige Knoten frei.
Quelle: [HWB+13, S. 4]

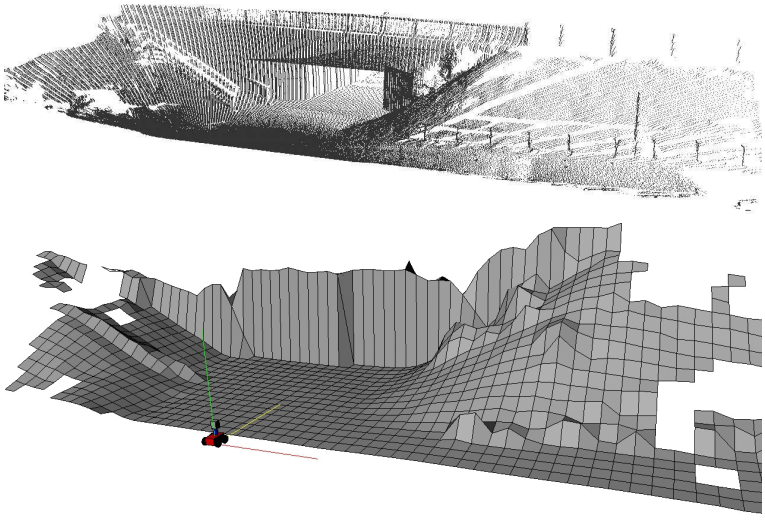
Verarbeitungsaufwand.

Octrees [Mea82; JT80] erlauben zusammenhängende Hindernisse oder Teilmengen des Freiraumes mit geringerem Speicherbedarf zu kodieren. Ein Octree ist eine Baumstruktur, die einen Quadtree für dreidimensionale Anwendungsfälle erweitert. Jeder Knoten im Baum enthält keinen oder acht Kindknoten und repräsentiert einen Würfel oder Quader im dreidimensionalen Raum. Der Raum wird hierarchisch abgebildet: Enthält ein Würfel verschiedene zu repräsentierende Werte, werden für den zugehörigen Knoten acht Kindknoten erstellt. Die Kindknoten unterteilen den Würfel in acht kleinere Würfel. Enthält ein Würfel nur gleiche zu repräsentierende Werte, muss er nicht unterteilt werden. Ein Beispiel eines Octrees ist in Abbildung 3.10 dargestellt.

Da sich in einer realen Umgebung oft zusammenhängende Hindernisse oder freie Bereiche befinden, kann die Umgebung im Vergleich zu einem Gitter bei gleicher Genauigkeit mit geringerem Speicherbedarf repräsentiert werden. Der Zugriff auf einzelne Werte im Raum ist mit höherem Aufwand verbunden, da der Baum durchsucht werden muss und kein Direktzugriff wie beim Gitter möglich ist [FACS03, S. 2987].

3.6.3 2,5D-Karten

Der Speicherbedarf zur Umgebungsrepräsentation kann durch eine 2,5D-Karte weiter reduziert werden. Eine 2,5D-Karte stellt Informationen der dreidimensionalen Umgebung in einer zweidimensionalen Struktur dar. Sie bildet die dreidimensionale Umgebung nicht vollständig



a)

b)

Abbildung 3.11: 2,5D-Karte: In a) ist ein Ausschnitt einer Umgebung mit einer 3D-Punktwolke abgebildet. Gut erkennbar ist die Brücke und die sich darunter befindliche Teilmenge des Freiraumes. Teilabbildung b) zeigt denselben Ausschnitt durch eine 2,5D-Höhenkarte. Die Teilmenge des Freiraumes unter der Brücke wird als belegt repräsentiert und ist daher für den Roboter nicht passierbar.
Quelle: [PTB07, S. 2]

ab, enthält aber mehr Informationen als eine 2D-Karte [FACS03, S. 2986].

Die am häufigsten eingesetzte 2,5D-Karte ist die Höhenkarte (elevation map). Eine Höhenkarte erweitert eine 2D-Gitterkarte um Höheninformationen über Hindernisse. Dabei wird ein zweidimensionales Gitter verwendet, bei dem jede Gitterzelle in der x-y-Ebene die z-Koordinate des höchsten gemessenen Hindernispunktes enthält. In [FACS03] wird zusätzlich die Wahrscheinlichkeit solcher Hindernispunkte gespeichert. Der Bereich unterhalb wird als Hindernis interpretiert. Teilmengen des Freiraumes unter dem höchsten Hindernispunkt, wie bei einer Brücke, werden nicht abgebildet. Bei der Planung kann daher nicht der gesamte Freiraum genutzt werden. Eine solche Situation ist in Abbildung 3.11 dargestellt.

Erweiterungen wie in [PTB07] umgehen den Nachteil der nicht abgebildeten Teilmengen des Freiraumes durch eine zusätzliche Klassifizierung der Gitterzellen als vertikale Gitterzelle oder Lückenzelle. In [TPB06] wird statt der z-Koordinate des höchsten Hindernispunktes pro Gitterzelle eine Liste von Durchschnitt und Varianz der Höheninformationen gespeichert.

4 Konzeption

In diesem Kapitel werden grundsätzliche konzeptionelle Entscheidungen für das zu entwickelnde Pfadplanungsverfahren getroffen. Zunächst wird in Abschnitt 4.1 eine geeignete 3D-Karte ausgewählt. In Abschnitt 4.2 wird ein Verfahren des Elastic Band Framework zur Umsetzung gewählt und an die gegebene Aufgabenstellung angepasst.

4.1 3D-Karte

Für ein 3D-Pfadplanungsverfahren wird eine 3D-Umgebungsrepräsentation benötigt. Für ROS existieren mehrere Karten, die für die Aufgabenstellung dieser Arbeit erweitert oder angepasst werden können. Die Karten werden in diesem Abschnitt erläutert und hinsichtlich ihrer Eignung für die gegebene Aufgabenstellung diskutiert.

4.1.1 Point Cloud Library

Die Point Cloud Library[1] (PCL) ist ein ROS-Paket zur Darstellung und Verarbeitung von Punktwolken. Beim Care-O-bot® 3 wird PCL für die Sensordaten der Laserscanner und der 3D-Kamera verwendet. Das ROS-Paket ermöglicht die Integration mehrerer Punktwolken von unterschiedlichen Sensoren in eine Repräsentation.

Aufgrund der hohen Anzahl an Datenpunkten ist eine direkte Verwendung von Punktwolken für die Pfadplanung ungeeignet, insbesondere für reaktive Pfadplanung unter Echtzeitanforderungen. Der notwendige Speicher- und Rechenaufwand kann für eine effiziente Verarbeitung auf unterschiedliche Arten reduziert werden, z.B. indem geometrische Objekte durch ein Polygonnetz approximieren werden. Bei einem Dreiecksnetz als Polygonnetz werden die Datenpunkte der Punktwolke als Eckpunkte einer Menge von Dreiecken aufgefasst. Die Anzahl der Dreiecke wird iterativ reduziert, indem mehrere Dreiecke zusammengefasst und durch ein größeres Dreieck ersetzt werden.

[1] http://wiki.ros.org/pcl

Eine Punktwolke kann durch eine Gitterkarte diskretisiert werden. Dazu wird die Gitterkarte über die Punktwolke gelegt und jede Gitterzelle als belegt markiert, die einen oder mehrere Datenpunkte der Punktwolke enthält. Die Größe der Gitterzellen bestimmt die Präzision und den Speicherbedarf der diskretisierten Gitterkarte. Analog kann ein Octree aufgebaut werden, indem eine Gitterkarte erstellt und durch eine entsprechende Baumstruktur repräsentiert wird.

4.1.2 Costmap2D

Das ROS-Paket costmap_2d[2] stellt eine zweidimensionale Umgebungsrepräsentation in Form einer Gitterkarte zur Verfügung. Das ROS-Paket costmap_2d wird vom ROS-Paket move_base und somit vom vorhandenen Planungsverfahren eingesetzt. Die verwendete Gitterkarte Costmap2D setzt sich aus zwei zweidimensionalen Gittern zusammen. Ein Gitter dient der Repräsentation der Hindernisse. Der Wert jeder Gitterzelle zeigt, ob die Zelle durch ein Hindernis belegt ist. Unsicherheit wird nicht explizit repräsentiert. Das zweite Gitter wird zur Speicherung von Kosten verwendet, die die räumliche Nähe einer Gitterzelle zu einem Hindernis abbilden.

Die Kosten vereinfachen die Kollisionsvermeidung. Ohne Kosten würde bei jedem Navigationsschritt für mehrere Gitterzellen in bestimmter Entfernung zur aktuellen Roboterpose geprüft, ob sie ein Hindernis enthalten. Durch die Kosten kann die minimale Entfernung einer Gitterzelle zum nächsten Hindernis direkt ausgelesen werden, wodurch explizite Kollisionsprüfungen entfallen. Die Kosten werden bei der Aktualisierung der Karte berechnet. Durch die Vorberechnung und Speicherung der Kosten werden wiederholte Berechnungen während der Navigation vermieden.

Die Kosten werden mittels einer Abstandsfunktion zwischen der aktuellen Gitterzelle und den Hindernissen berechnet. Gitterzellen, die ein Hindernis beinhalten, werden dabei maximale Kosten zugewiesen. Die Hindernisse werden gleichmäßig um einen Radius aufgebläht, der dem Innenkreisradius der Grundfläche des Roboters entspricht. Gitterzellen innerhalb dieses Radius erhalten einen hohen, konstanten Wert. Außerhalb des Radius werden die Kosten mit zunehmender Distanz zum Roboter logarithmisch abfallend berechnet. Ein Parameter gibt die maximale Distanz zum Roboter an, bis zu der Kosten berechnet werden. Gitterzellen außerhalb dieser Distanz gelten als frei und erhalten einen minimalen Kostenwert. Das Prinzip kann als Berechnung eines abstoßendes Potentials für jedes Hindernis aufgefasst werden.

[2] http://wiki.ros.org/costmap_2d

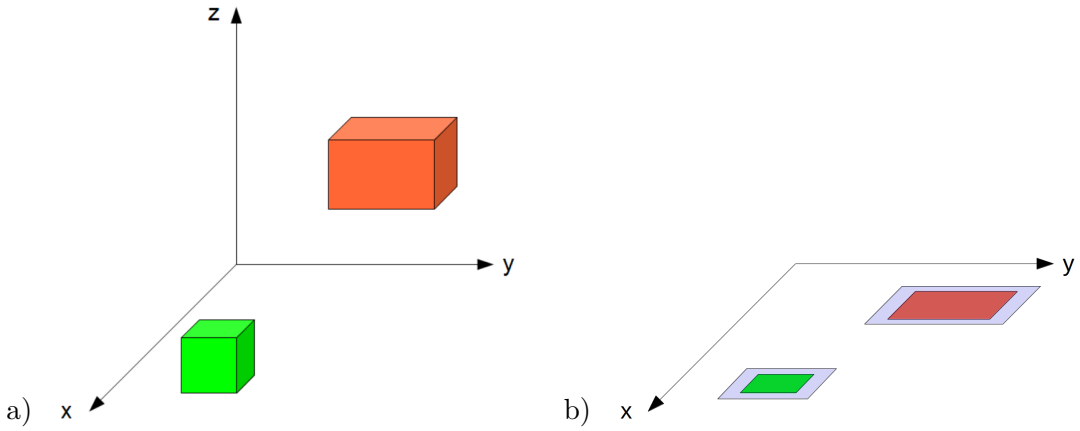


Abbildung 4.1: Hindernisdarstellung der erweiterten Costmap2D: Die zwei farbigen Quader in a) markieren von den Sensoren erfasste Hindernisse. In b) ist die Repräsentation der Hindernisse in der erweiterten Costmap2D dargestellt. Die Hindernisse wurden in die x-y-Ebene projiziert. Der Höhenunterschied zwischen ihnen wird nicht repräsentiert. Die blauen Bereiche visualisieren die aufgeblähten Hindernisse.

Im selben ROS-Paket ist eine Erweiterung des Prinzips auf eine 2,5D-Karte vorhanden. Die Sensordaten werden in einem dreidimensionalen Gitter mit begrenzter Auflösung temporär erfasst. Die erfassten Hindernisse werden anschließend auf die x-y-Ebene projiziert und in das zweidimensionale Gitter zur Speicherung der Kosten integriert. Abbildung 4.1 zeigt die resultierende Repräsentation für eine Beispielumgebung.

4.1.3 3D-Erweiterung Costmap2D

Eine 3D-Karte auf Basis der Costmap2D kann auf verschiedene Arten erstellt werden. Eine Möglichkeit bildet eine mehrschichtige Karte. Eine mehrschichtige Karte setzt sich aus mehreren 2D- bzw. 2,5D-Teilkarten zusammen, die entlang der z-Achse angeordnet werden. Für jede Teilkarte werden die einfließenden Sensordaten durch eine Mindest- und eine Maximalhöhe für erfasste Objekte begrenzt. Anstatt Berechnungen wie Kollisionsabfragen dreidimensional durchzuführen, genügen mehrere zweidimensionale Berechnungen. Entlang der z-Achse zusammenhängende Hindernisse werden nicht als zusammenhängend modelliert. Das Planungsverfahren muss für Kollisionsabfragen einzelnen Roboterabschnitten die zuständigen Teilkarten zuordnen. Auflösung und Speicherbedarf sind durch eine parametrierbare Anzahl von Teilkarten skalierbar. Das Verfahren kann schlecht durch eine Schnittstelle für 3D-Karten gekapselt werden, um ein flexibles Austauschen der Kartenimplementierung zu ermöglichen. Abbildung 4.2 zeigt das Prinzip der mehrschichtigen Karte für die Beispielumgebung aus Abbildung 4.1.

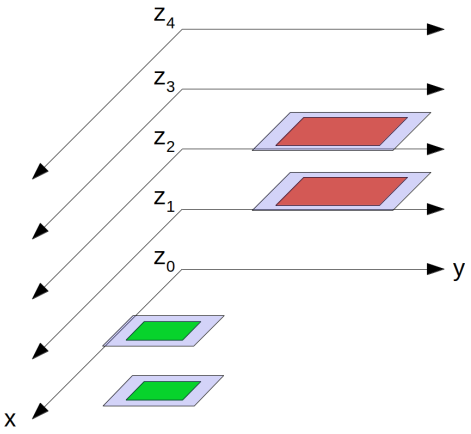


Abbildung 4.2: Mehrschichtige Erweiterung der Costmap2D: In einer mehrschichtigen Karte findet das Aufblähen der Hindernisse und die Kostenberechnung in jeder Teilkarte statt. Jeder Teilkarte ist ein disjunkter Abschnitt des Raumes entlang der z-Achse zugewiesen.

Eine Alternative ist die Nutzung einer erweiterten 2,5D-Karte, bei der jedes Voxel Höheninformationen aus mehreren Abschnitten der z-Achse kodiert. In der vorhandenen 2,5D-Karte wird die z-Achse in bis zu 16 Bereiche unterteilt und für jeden Bereich kodiert, ob er frei, belegt oder unbekannt ist. Die Informationen werden über Bitoperationen in einem numerischen Datentypen zusammengefasst. Die einfachste Erweiterung besteht darin, die Projektion der Hindernisse in die x-y-Ebene zu unterbinden und direkt auf die kodierten Höhendaten zuzugreifen. Das Aufblähen der Hindernisse und die Kostenfunktion müssen auf drei Dimensionen erweitert werden. Die Kosten können im Gegensatz zur Costmap2D nicht gespeichert werden, sondern müssen pro Kollisionsabfrage erneut berechnet werden. Dadurch sinkt der Rechenaufwand zur Erstellung der Karte, wogegen der Rechenaufwand bei der Nutzung der Karte steigt.

Eine andere Erweiterungsmöglichkeit ist die Verwendung eines dreidimensionalen Gitters statt eines zweidimensionalen. Die Höheninformationen der vorhandenen 2,5D-Karte werden in ein separates dreidimensionales Gitter übertragen. In diesem kann das Prinzip der Vorberechnung der Kosten erhalten bleiben. Dafür müssen das Aufblähen der Hindernisse und die Kostenfunktion auf den 3D-Fall erweitert werden. Durch die Auslagerung in eine separate Karte entsteht eine Schnittstelle, über die die vorhandene 2,5D-Karte ausgetauscht werden kann oder zusätzliche Datenquellen integriert werden können. Nachteilig ist bei dieser Variante der erhöhte Speicherbedarf für das separate Gitter. Abbildung 4.3 zeigt den Unterschied in der Repräsentation der Hindernisse der beiden Erweiterungsmöglichkeiten.

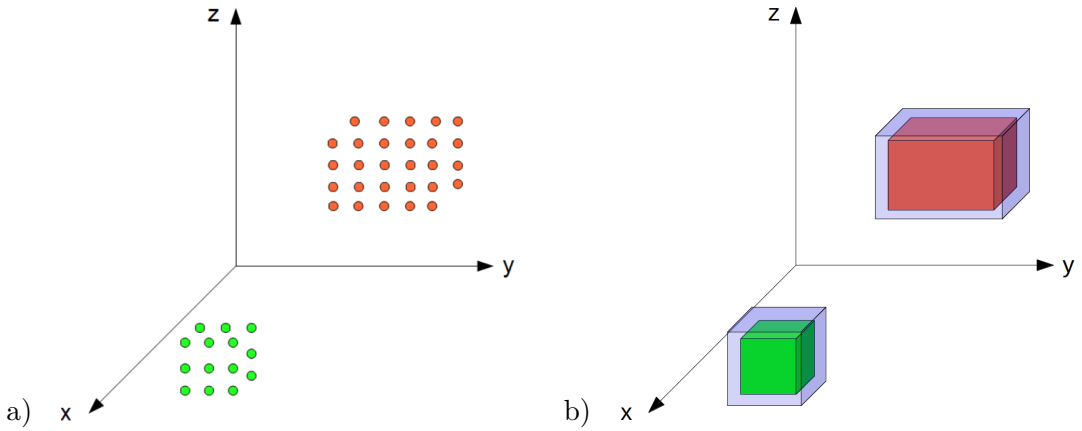


Abbildung 4.3: Auslesen der Höheninformationen: Beim Direktzugriff in a) werden die erfassten Höhendaten der erweiterten Costmap2D bei jeder Kollisionsabfrage direkt ausgelesen. Da diese Erweiterung ohne eigene Repräsentation agiert, werden die Kosten nicht explizit repräsentiert, sondern müssen pro Kollisionsabfrage berechnet werden. In b) werden die Höhendaten in ein separates dreidimensionales Gitter übertragen. Das Aufblähen der Hindernisse und die Berechnung der Kosten erfolgt analog zur Costmap2D.

4.1.4 Weitere Karten

Im Bereich von 3D-Karten wurden mehrere ROS-Pakete entwickelt. OctoMap[3] [HWB+13] ist eine Implementierung einer 3D-Karte auf Basis eines Octrees. Die Karte unterstützt RGB-D-Kameras und Laserscanner. Bei der Integration von Sensordaten wird Unsicherheit berücksichtigt. Die Größe des repräsentierten Bereiches ist zur Laufzeit variabel. OctoMap ist in mehreren ROS-Paketen integriert, unter anderem in einem ROS-Paket zur 3D-Navigation [HPJ+12], welches lokale und globale Pfadplanung beinhaltet. Die in dieser Arbeit verwendete ROS-Distribution Hydro Medusa wird unterstützt.

Das ROS-Paket ccny_rgdb[4] [DVX13] nutzt die 3D-Sensordaten der Microsoft Kinect, um visuelle Odometrie und SLAM durchzuführen. Durch die ausschließliche Nutzung der Microsoft Kinect als Sensor können die Laserscanner des Care-O-bot® 3 nicht verwendet werden. Die intern verwendeten 3D-Punktwolken können über ROS veröffentlicht werden und müssen für die Pfadplanung in eine einzelne Kartendarstellung integriert werden. Alternativ können die 3D-Punktwolken über einen Dienst als OctoMap in eine Datei exportiert werden. Das Schreiben in eine Datei und anschließende Auslesen genügt der Echtzeitanforderung der lokalen Pfadplanung nicht. Die aktuellste, unterstützte ROS-Distribution ist Groovy Galapagos.

[3] http://wiki.ros.org/octomap
[4] http://wiki.ros.org/ccny_rgbd/

Ähnlich wie das ROS-Paket ccny_rgdb agiert das ROS-Paket rgbdslam[5]. 3D-Punktwolken von RGB-D-Kameras werden für ein SLAM-Verfahren in eine 3D-Karte integriert. Laserscanner werden nicht unterstützt. Die erstellte 3D-Punktwolke kann über ROS veröffentlicht und zur Aktualisierung einer OctoMap verwendet werden. Das ROS-Paket rgbdslam ist für die ROS-Distribution Fuerte verfügbar. Eine aktualisierte Version unter dem Namen RGBDSLAMv2[6] [EHS+14] unterstützt ROS Hydro Medusa, liegt aber zum Zeitpunkt der Erstellung dieser Arbeit ausschließlich als Beta-Version vor.

RTAB-Map[7] [LM13] nutzt graphbasiertes SLAM zur Erstellung einer 3D-Karte unter Verwendung von Loop Closure Detection. Sensordaten von RGB-D-Kameras, Laserscannern und der Odometrie können in die Karte integriert werden. Das Verfahren kann in einem Mapping- und in einem Lokalisierungsmodus verwendet werden. Für den Lokalisierungsmodus muss eine Karte mit einer Mindestanzahl an Referenzpunkten für die Lokalisierung bereitgestellt werden. Die ROS-Distribution Hydro Medusa wird unterstützt.

4.1.5 Diskussion 3D-Karte

Für die Auswahl einer geeigneten 3D-Umgebungsrepräsentation sind verschiedene Kriterien zu beachten. Neben der Genauigkeit der Karte ist der benötigte Speicherbedarf und der Aufwand zur Berechnung von Kollisionen von Bedeutung. Zusätzlich muss eine einfache Integration in das Gesamtsystem und die Erweiterbarkeit der Karte gewährleistet sein.

Die ROS-Pakete ccny_rgdb und rgbdslam wurden für 3D-SLAM statt zur Pfadplanung entwickelt, unterstützen keine Laserscanner und sind nicht mit der verwendeten ROS-Distribution kompatibel. Sie sind daher nicht für die gegebene Aufgabenstellung geeignet. RTAB-Map und OctoMap unterliegen diesen Einschränkungen nicht. Das ROS-Paket RTAB-Map wurde ebenfalls für SLAM entwickelt, verfügt aber über einen eigenständigen Lokalisierungsmodus. Dafür notwendige statische Karten sind allerdings nicht vorhanden und lassen sich nicht aus den bereits modellierten statischen Karten für die Simulationsumgebung Gazebo generieren. Das ROS-Paket OctoMap erfüllt die Voraussetzungen für die gegebene Aufgabenstellung. Die Integration des ROS-Pakets der OctoMap und des zugehörigen ROS-Pakets zur 3D-Navigation ist mit umfangreichen Umstrukturierungen des Gesamtsystems und hohem Umsetzungsaufwand verbunden.

Eine Lösung auf Basis der Costmap2D lässt sich leichter in das Gesamtsystem integrieren. Der ROS-Knoten move_base als Basis der vorhandenen Pfadplanung nutzt Costmap2D. Durch

[5] http://wiki.ros.org/rgbdslam
[6] http://felixendres.github.io/rgbdslam_v2/
[7] https://code.google.com/p/rtabmap/

eine Erweiterung der Costmap2D lassen sich Änderungen an Schnittstellen außerhalb des ROS-Pakets eband_local_planner vermeiden. Als weiterer Vorteil ist die Vorberechnung der Kosten anzusehen, die eine effiziente Berechnung von Distanzen und Prüfung von Kollisionen ermöglicht.

Bei der Erweiterung der Costmap2D hat die Nutzung eines dreidimensionalen Gitters für die lokale Pfadplanung die größten Vorteile. Eine 3D-Karte bringt die höchste Genauigkeit für das Planen von kurzen Pfaden, die dicht an Hindernissen vorbei führen. Durch Beibehaltung der Arbeitsweise der vorhandenen Karte lässt sich die Erweiterung leicht integrieren und kann für andere vorhandene Pfadplanungsverfahren verwendet werden. Für die lokale Pfadplanung genügt ein kleiner Umgebungsausschnitt, wodurch der Nachteil des hohen Speicherbedarfs kompensiert wird. Da sich der Roboter in dieser Arbeit nur in der x-y-Ebene bewegt und kein whole-body motion planning notwendig ist, kann die Auflösung der z-Achse zur weiteren Verringerung des Speicherbedarfs niedriger als die Auflösung der x- und y-Achse gewählt werden.

4.2 Elastic Band Framework

Neben dem Elastic-Band-Verfahren existieren zwei Erweiterungen des Prinzips, das Elastic-Strip-Verfahren und das Elastic-Roadmap-Verfahren. Zunächst wird die Eignung der Verfahren für die gegebene Aufgabenstellung diskutiert und eines der Verfahren zur Umsetzung ausgewählt. In den folgenden Abschnitten werden Ansätze diskutiert, wie das gewählte Verfahren an die gegebene Aufgabenstellung und deren Einschränkungen angepasst werden kann.

4.2.1 Diskussion Elastic-Band-Verfahren

Da das Elastic-Band-Verfahren im Konfigurationsraum agiert, sind für eine Erweiterung auf eine 3D-Umgebung keine direkten Änderungen notwendig. Dagegen erhöht sich die Komplexität der Transformation der Hindernisse in den Konfigurationsraum. Die Transformation ist die rechenintensivste Operation im Planungsprozess [Bro00, S. 20], wodurch die Erweiterung auf eine 3D-Umgebung mit hohem zusätzlichen Rechenaufwand verbunden ist. Durch die hohe Anzahl der Freiheitsgrade des Roboters ist die Planung im Konfigurationsraum komplex; insgesamt sind zehn Dimensionen notwendig.

Das Elastic-Strip-Verfahren agiert im Arbeitsraum. Für eine Erweiterung auf eine 3D-Umgebung muss der Arbeitsraum auf drei Dimensionen erweitert werden. Die Planung im

3D-Arbeitsraum ist dennoch mit weniger Verarbeitungsaufwand verbunden als die Planung im 10D-Konfigurationsraum. Das Verfahren kann ohne größeren zusätzlichen Rechenaufwand auf whole-body motion planning erweitert werden, da die Planung im Arbeitsraum nur gering durch zusätzliche DOF beeinträchtigt wird. Dies würde auch die Integration von aufgabenspezifischem Verhalten in die lokale Pfadplanung ermöglichen. Da die vorhandene Implementierung der lokalen Pfadplanung bereits im Arbeitsraum agiert, lässt sich eine dreidimensionale Planung im Arbeitsraum besser in das Gesamtsystem integrieren.

Das Elastic-Roadmap-Verfahren realisiert eine globale reaktive Pfadplanung und vermeidet daher trotz Unvollständigkeit lokale Minima. Da das Verfahren wie das Elastic-Strip-Verfahren im Arbeitsraum agiert, verfügt es über dieselben Vorteile. Das Verfahren ist für einen mobilen Roboter mit Manipulator und die Integration aufgabenspezifischen Verhaltens des Manipulators ausgelegt. Care-O-bot® 3 verfügt über einen Manipulator, der in dieser Arbeit jedoch als statisch angenommen wird. Die Pfadplanung wird ausschließlich für die mobile Basis realisiert. Die Controller und die Relation zur Verbindung von Meilensteinen können unabhängig von aufgabenbasierten Verhalten durch eine einfache Navigationslogik implementiert werden. Dies widerspricht jedoch einem Grundgedanken des Verfahrens und verursacht zusätzlichen Umsetzungsaufwand. Die Aktualisierung aller Meilensteine bedeutet vor allem in einer 3D-Umgebung einen Mehraufwand gegenüber dem Elastic-Strip-Verfahren. Die Pfadplanung in einer einzigen globalen Umgebungsrepräsentation widerspricht dem Paradigma des Gesamtsystems und lässt sich daher schwer integrieren.

Zusammenfassend stellt das Elastic-Strip-Verfahren das geeignetste lokale Pfadplanungsverfahren für diese Arbeit dar. Die Planung im Arbeitsraum verringert den Rechenaufwand gegenüber dem Elastic-Band-Verfahren und bringt weitere Vorteile. Durch die dynamische Pfadanpassung und die Möglichkeit, den Streifen temporär zu unterbrechen, ist ein Neuplanen des globalen Pfades selten nötig. Der Vorteil der globalen Pfadplanung beim Elastic-Roadmap-Verfahren überwiegt den zusätzlichen Mehraufwand bei der Berechnung und der Integration in das Gesamtsystem nicht.

4.2.2 Einschränkungen der Aufgabenstellung

In der gegebenen Aufgabenstellung liegen Einschränkungen vor, die eine Anpassung des originalen Elastic-Strip-Verfahrens nötig machen:

- globaler 2D-Pfad: Das globale Pfadplanungsverfahren stellt einen Pfad für eine 2D-Umgebung zur Verfügung. Aus diesem muss ein 3D-Pfad erstellt werden, der fortlaufend modifiziert werden kann.

- 2D-Steuerung: Der Roboter kann nur Translationsbewegungen in der Ebene und Rotationen um die z-Achse durchführen. Andere Bewegungsbefehle können nicht umgesetzt und müssen daher vermieden werden.
- Abhängigkeiten zwischen Blasen: In dieser Arbeit kann nur die Basis des Roboters bewegt werden. Daraus ergeben sich statische Abhängigkeiten zwischen den Blasen an verschiedenen Komponenten des Roboters. Beispielsweise können Kräfte, die auf den Sensorkopf des Roboters wirken, nicht direkt in Bewegungen umgesetzt werden. Sie müssen auf die Blase der Basis übertragen werden und zu entsprechenden Bewegungsbefehlen führen.

Neben diesen Einschränkungen müssen bei der Anpassung weitere Kriterien berücksichtigt werden. Der gewählte Ansatz soll den Rechenaufwand minimieren, eine einfache Implementierung und Wartung ermöglichen, sich leicht in das vorhandene Gesamtsystem zur Steuerung integrieren lassen und zukünftige Erweiterungen ermöglichen.

4.2.3 Originaler Ansatz

Zunächst wird die Umsetzung des originalen Ansatzes für das Elastic-Strip-Verfahren skizziert. Er dient als Grundlage für angepasste Ansätze und soll das Grundprinzip verdeutlichen. Gegeben ist ein globaler 3D-Pfad. Aus dem 3D-Pfad werden einzelne, unabhängige Bänder erstellt. Anhand der Umgebungsrepräsentation werden die Teilmengen des Freiraumes entlang des Pfades um den Roboter ermittelt und die den Roboter umgebenden Hüllen erstellt. Alle Bänder werden anschließend einzeln optimiert.

Die Optimierung eines Bandes e_j findet iterativ in zwei Phasen statt. In jeder Iteration werden zunächst die Kräfte f_g für die einzelnen Blasen von e_j berechnet, auf die Blasen angewendet und die Blasen entsprechend neu positioniert. Anschließend wird in der zweiten Phase das Band als Ganzes optimiert. Redundante Blasen werden aus dem Band entfernt sowie Lücken im Band erkannt und durch neue Blasen aufgefüllt.

Der originale Ansatz realisiert eine vollständige 3D-Pfadplanung und ermöglicht whole-body motion planning sowie 3D-Navigation, z. B. Rotationen um die x- und die y-Achse. Da keine Einschränkungen für die Berechnung der Kräfte und die Umsetzung in Bewegungsbefehle vorhanden sind, werden optimale Pfade im Sinne der lokalen Pfadplanung erstellt. Für diese Arbeit ist das Verfahren nicht direkt umsetzbar, da kein globaler 3D-Pfad gegeben ist. Weiterhin bestehen Abhängigkeiten zwischen den einzelnen Bändern. Andere Ansätze können den notwendigen Rechenaufwand reduzieren, da aufgrund dieser Abhängigkeiten nicht alle Bänder einzeln optimiert werden müssen.

4.2.4 3D-Pfad simulieren

Ein abgewandelter Ansatz ermöglicht das Elastic-Strip-Verfahren umzusetzen, indem ein globaler 3D-Pfad simuliert wird. Dafür wird zunächst die Hülle h_0 für die Startpose erstellt. Der gegebene 2D-Pfad wird in ein elastisches Band $e_0 = \{b_0, b_1, \ldots, b_n\}$ transformiert, das so genannte Basisband. Die Hülle h_0 wird auf jede Blase des Basisbandes übertragen, d. h. kopiert und auf das Zentrum der Blase verschoben:

$$h_i = h_0 + (b_i - b_0)\,. \tag{4.1}$$

Die Blasen an der jeweils gleichen Position in den Hüllen h_i bilden ein elastisches Band e_j entlang des Pfades: $e_j = \{h_{0j}, h_{1j}, \ldots, h_{nj}\}$, wobei h_{ij} die Blase b_j der Hülle h_i bezeichnet. Dadurch entsteht eine Menge unabhängiger elastischer Bänder, die zusammen einen elastischen Streifen $s = \{e_0, e_1, \ldots, e_n\}$ bilden.

Die entstandenen Bänder werden einzeln optimiert. Um die Änderungen an den einzelnen Bändern auf das Basisband zu übertragen, müssen die berechneten Kräfte zusammengefasst werden. Dazu werden die optimierten Blasen wieder einer Hülle zugeordnet. Alle Kräfte, die auf Blasen einer Hülle wirken, werden kombiniert und auf die entsprechende Blase des Basisbandes übertragen:

$$f_{g0j} = \sum f_{gij}. \tag{4.2}$$

Dieser Ansatz weicht gering vom originalen Ansatz ab und kann einfach auf whole-body motion planning und 3D-Navigation erweitert werden. Da die Bänder unabhängig von einander optimiert werden, werden die Freiheiten in der Planausführung nicht wesentlich eingeschränkt. Daher sind weiterhin nahezu optimale Pfade möglich. Der Rechenaufwand kann gegenüber dem originalen Ansatz reduziert werden, da zur Optimierung der Bänder für die gegebene Aufgabenstellung 2D-Berechnungen ausreichend sind.

Das Kombinieren der Kräfte ist nicht trivial. Eine einfache Mittelwertbildung kann zu ungültigem Verhalten führen. Wenn z. B. die Kräfte einer Blase ein Ausweichen auf die linke Seite eines zentralen Hindernisses bewirken und die Kräfte einer anderen Blase zu einem Ausweichen der Blase auf die rechte Seite führen, kann eine Mittelwertbildung eine Kollision verursachen. Da nur die Kräfte von Blasen einer Hülle kombiniert werden, wird das häufige Auftreten einer solchen Konstellation vermieden. Die Zuordnung der optimierten Blasen zu einer Hülle kann zu komplizierten Sonderfällen führen, da in jedem Band Blasen entfernt und hinzugefügt werden können.

4.2.5 Redundanz prüfen

Wenn sich eine Hülle aus vielen Blasen zusammensetzt, muss zur Optimierung des elastischen Streifens eine große Anzahl an Blasen bearbeitet werden. Die Blasen unterschiedlicher Hüllen überlappen sich teilweise. Bei großen Überlappungen können Blasen redundant werden und unnötige Berechnungen verursachen. Beim zuvor vorgestellten Ansatz werden redundante Blasen nur entlang eines Bandes erkannt, nicht zwischen einzelnen Bändern.

Analog zum vorherigen Ansatz wird eine Hülle h_0 für die Startpose erstellt und der gegebene 2D-Pfad in einen elastischen Streifen $s = \{e_0, e_1, \ldots, e_n\}$ transformiert. Bei der ersten Phase der Optimierung der einzelnen Bänder e_j werden die Kräfte und die Radien der Blasen für drei Dimensionen ermittelt. In der zweiten Phase der Optimierung werden zur Überprüfung auf redundante Blasen b_{ij} neben den benachbarten Blasen $\{b_{oj}, b_{1j}, \ldots, b_{nj}\}$ innerhalb des Bandes e_j benachbarte Blasen von anderen Bändern $\{b_{i0}, b_{i1}, \ldots, b_{im}\}$ berücksichtigt. Anschließend werden die optimierten Blasen einer Hülle zugeordnet und die resultierenden Kräfte auf das Basisband übertragen.

Dieser Ansatz wandelt den originalen Ansatz geringfügig ab und lässt sich einfach auf whole-body motion planning erweitern. Durch die Verwendung von 3D-Berechnungen sind nahezu optimale Pfade und eine einfache Erweiterung auf 3D-Navigation möglich. Das Entfernen redundanter Blasen verringert den Rechenaufwand zur Optimierung der zu berücksichtigenden Blasen beim Kombinieren der Kräfte. Gleichzeitig wird das Kombinieren und die Zuordnung der Blasen zu einer Hülle komplizierter, da die Blasen stärker verschoben und öfter entfernt werden.

4.2.6 Kräfte früh kombinieren

In dieser Arbeit kann die statische Pose des Roboters ausgenutzt werden, um Berechnungen zu sparen und ungünstige Effekte beim Kombinieren der Blasen zu vermeiden. Anstatt jedes Band e_j vollständig zu optimieren, können die Kräfte f_g für die einzelnen Hüllen h_i in der ersten Phase jeder Optimierungsiteration kombiniert werden. Die resultierenden Kräfte werden durch Formel 4.2 auf das Basisband e_0 übertragen. e_0 wird anschließend in der zweiten Phase als einziges Band optimiert. Nach der Optimierung wird das verformte Basisband abschließend auf Kollisionen geprüft.

Bei diesem Ansatz werden die Abhängigkeiten zwischen den Blasen durch die statische Pose des Roboters implizit beachtet und benutzt, um den Rechenaufwand zur Optimierung zu verringern. Die nachträgliche Zuordnung der optimierten Blasen zur entsprechenden Hülle

entfällt, da nur für ein Band Blasen entfernt und Lücken aufgefüllt werden. Die Kombination der Kräfte wird im Vergleich zu den anderen Ansätzen vereinfacht, da der Positionsunterschied der zu kombinierenden Blasen nicht durch die zweite Phase der Optimierung vergrößert wird. Ungünstige Konstellationen beim Kombinieren treten mit geringer Wahrscheinlichkeit auf.

Im Gegensatz zu den zuvor beschriebenen Ansätzen weicht dieser Ansatz stärker vom originalen Verfahren ab und ist nicht direkt auf whole-body motion planning sowie nur beschränkt für 3D-Navigation erweiterbar. Durch die eingeschränkte Optimierung wird der Suchraum stärker als bei den anderen Ansätzen begrenzt, sodass die erstellten Pfade nicht immer optimal sind.

4.2.7 Diskussion Ansätze

Der originale Ansatz für das Elastic-Strip-Verfahren ist für die gegebene Aufgabenstellung nicht direkt anwendbar und muss modifiziert werden. Die Ansätze „3D-Pfad simulieren“ und „Redundanz prüfen“ weichen gering vom originalen Ansatz ab und ermöglichen eine einfache Erweiterung auf whole-body motion planning und Pfadplanung außerhalb der x-y-Ebene. Der Ansatz „Kräfte früh kombinieren“ ist weniger komplex und einfacher zu realisieren, da die Abhängigkeiten zwischen den Blasen implizit berücksichtigt sind. Die Zuordnung von Blasen zu einer Hülle nach der Optimierung entfällt. Die Kombination der Kräfte wird vereinfacht, da keine Änderungen der Blasen durch die Optimierung der Bänder zu berücksichtigen sind. Der Rechenaufwand sinkt, da nur ein Band optimiert wird und keine Zuordnung von Blasen zu Hüllen notwendig ist. In dieser Arbeit wird der Ansatz „Kräfte früh kombinieren“ verwendet. Die Vorteile der anderen Ansätze, wie bessere Erweiterbarkeit und geringere Einschränkung des Suchraumes, wiegen die Nachteile der höheren Komplexität und des höheren Rechnungsaufwandes nicht auf.

5 Implementierung

In diesem Kapitel wird die konkrete Umsetzung des entwickelten Pfadplanungsverfahrens beschrieben. Ein Überblick über die an der Pfadplanung beteiligten ROS-Pakete und Klassen wird in Abschnitt 5.1 gegeben. In Abschnitt 5.2 wird der Ablauf des Pfadplanungsprozesses gezeigt. Abschließend werden die wichtigsten Teilschritte zur Planung eines lokalen Pfades in Abschnitt 5.3 im Detail erläutert.

5.1 Struktureller Aufbau

In diesem Abschnitt wird der strukturelle Aufbau des Gesamtsystems erläutert. Ein Überblick über die Systemumgebung und die wichtigsten ROS-Pakete ist in Abbildung 2.3 dargestellt. Abbildung 5.1 zeigt ein Klassendiagramm für das ROS-Paket eband_local_planner und die Schnittstellen zu anderen ROS-Paketen.

Die Klasse MoveBase bildet den Einstiegspunkt der gesamten Pfadplanung. Mit ihr werden Instanzen der benötigten Klassen erstellt und verwaltet. Die Klasse steuert den Ablauf der globalen und lokalen Pfadplanung. Die abstrakten Klassen BaseLocalPlanner und BaseGlobalPlanner sind Basisklassen für unterschiedliche lokale bzw. globale Pfadplanungsverfahren. Bei der Initialisierung von MoveBase wird jeweils eine Realisierung über ein Plug-in-Verfahren erstellt.

Für beide Realisierungen wird eine Instanz von Costmap2DROS zur Verfügung gestellt. Die Klasse Costmap2DROS bildet die Schnittstelle der Costmap2D für andere ROS-Pakete. Eine Costmap2DROS-Instanz erstellt eine Instanz der Klasse Costmap2D, verwaltet sie und gewährt anderen Klassen Zugriff auf die erstellte Karte. Die Karten beider Pfadplanungsverfahren agieren unabhängig voneinander und können unterschiedlich konfiguriert werden.

Die Klasse EBandPlannerROS bildet die Schnittstelle des entwickelten Pfadplanungsverfahrens zu Klassen anderer ROS-Pakete. Sie ist eine Realisierung der abstrakten Basisklasse

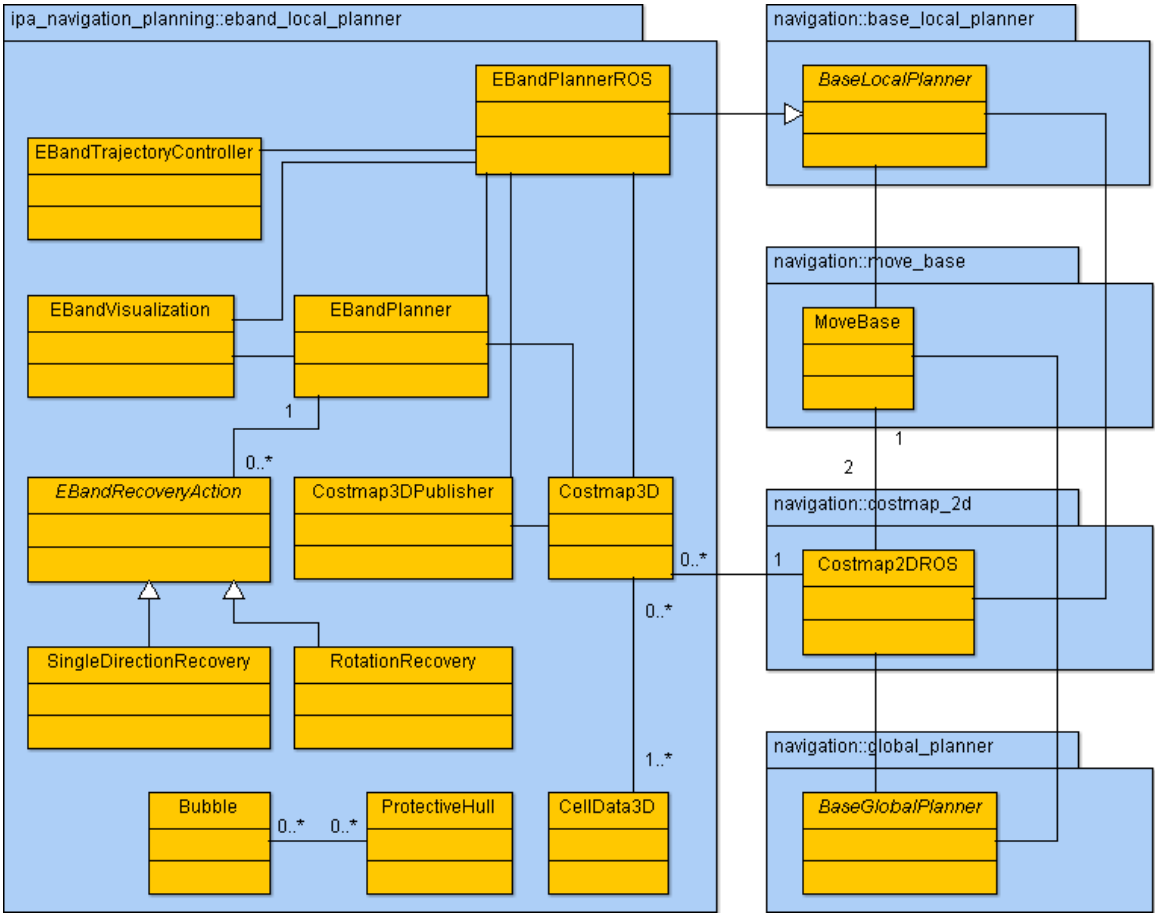


Abbildung 5.1: Klassendiagramm des Gesamtsystems: Das Klassendiagramm zeigt eine Übersicht der statischen Zusammenhänge der relevanten Klassen aus den ROS-Paketen, die am Gesamtsystem unmittelbar beteiligt sind. Aus Gründen der Übersichtlichkeit sind nicht alle Klassen aufgeführt und einige Zusammenhänge vereinfacht dargestellt. Sofern nicht anders gekennzeichnet, sind die Multiplizitäten der Assoziationen 1 zu 1.

BaseLocalPlanner und steht somit als ein Plug-in für MoveBase zur Verfügung. Die Kommunikation mit MoveBase umfasst unter anderem den Empfang von globalen Plänen und den Erhalt von Statusinformationen der lokalen Pfadplanung, wie dem Erreichen der Zielpose und dem Auftreten eines ungültigen Zustandes. Eine Instanz von EBandPlannerROS erstellt jeweils eine Instanz der Klassen EBandPlanner, EBandVisualization, TrajectoryController, Costmap3D und Costmap3DPublisher. Sie verwaltet die erstellten Instanzen, löst gegenseitige Abhängigkeiten auf und definiert den Kontrollfluss innerhalb des ROS-Pakets.

Die Klasse Costmap3D realisiert die interne 3D-Gitterkarte der lokalen Pfadplanung. Eine Instanz der Klasse greift auf veröffentlichte Hindernispositionen der Costmap2D-Instanz für die lokale Pfadplanung zurück. Zum dreidimensionalen Aufblähen der Hindernisse wird eine Vorrangwarteschlange für den Datentyp CellData3D verwendet. Eine Instanz der

Klasse Costmap3DPublisher veröffentlicht die erstellte 3D-Gitterkarte, um sie z. B. über das Visualisierungsprogramm Rviz darstellen zu lassen.

Die Klasse EBandPlanner ist für die Kernfunktionalität der lokalen Pfadplanung zuständig, d. h. für die reaktive Modifizierung des elastischen Streifens. Dabei werden Realisierungen der abstrakten Basisklasse EBandRecoveryAction zur Wiederherstellung eines gültigen Zustands genutzt. Eine Instanz der Klasse EBandVisualization wird von den Klassen EBandPlanner und EBandPlannerROS genutzt, um den modifizierten elastischen Streifen bzw. Teile dessen zu veröffentlichen. Die veröffentlichten Daten können über Rviz visualisiert werden. Nach einer erfolgreichen Modifizierung wird eine Instanz der Klasse TrajectoryController von EBandPlannerROS aufgerufen, um anhand des elastischen Streifens Steuerbefehle für die Roboterbasis zu generieren.

Die Datentypen Bubble und ProtectiveHull repräsentieren eine Blase bzw. eine den Roboter umschließende Hülle, die sich aus mehreren Blasen zusammensetzt. Beide Datentypen werden von mehreren Klassen des ROS-Pakets eband_local_planner verwendet. In Abbildung 5.1 sind zur besseren Übersichtlichkeit keine Assoziationen zu diesen Klassen aufgezeigt.

5.2 Ablauf der Pfadplanung

Der Ablauf des Pfadplanungsprozesses ist in Abbildung 5.2 in Form eines Aktivitätsdiagramms dargestellt. Ausgangspunkt der Pfadplanung ist der Start des Knotens move_base, durch den eine Instanz der Klasse MoveBase erstellt wird. Über die Schnittstellen Costmap2DROS und EBandPlannerROS werden Instanzen von Costmap2D und EBandPlanner erstellt und parallel zu MoveBase ausgeführt.

Costmap2D greift mit einer Frequenz von 4 Hz auf die Sensordaten der Laserscanner und der 3D-Kamera zu und aktualisiert einen Kartenausschnitt von $4 \times 4 \times 2$ m, dessen Zentrum die aktuelle Pose des Roboters ist. Voxel, die erfasste Hindernisse repräsentieren, werden über ROS veröffentlicht. Die Schleife wird mit dem Beenden des Knotens move_base beendet.

Costmap3D empfängt die veröffentlichten Voxel und integriert sie in eine separate 3D-Gitterkarte der lokalen Pfadplanung. Die Hindernisse werden dreidimensional aufgebläht und anschließend die Kosten für die Voxel berechnet. Die Kosten stehen für die lokale Pfadplanung zur Verfügung und werden zusätzlich über ROS veröffentlicht. Die Aktualisierung der Karte wird unabhängig vom Erreichen einer Zielpose beendet, wenn der Knoten move_base beendet wird.

Nachdem die Pfadplanung angestoßen und eine Zielpose an die Instanz von MoveBase

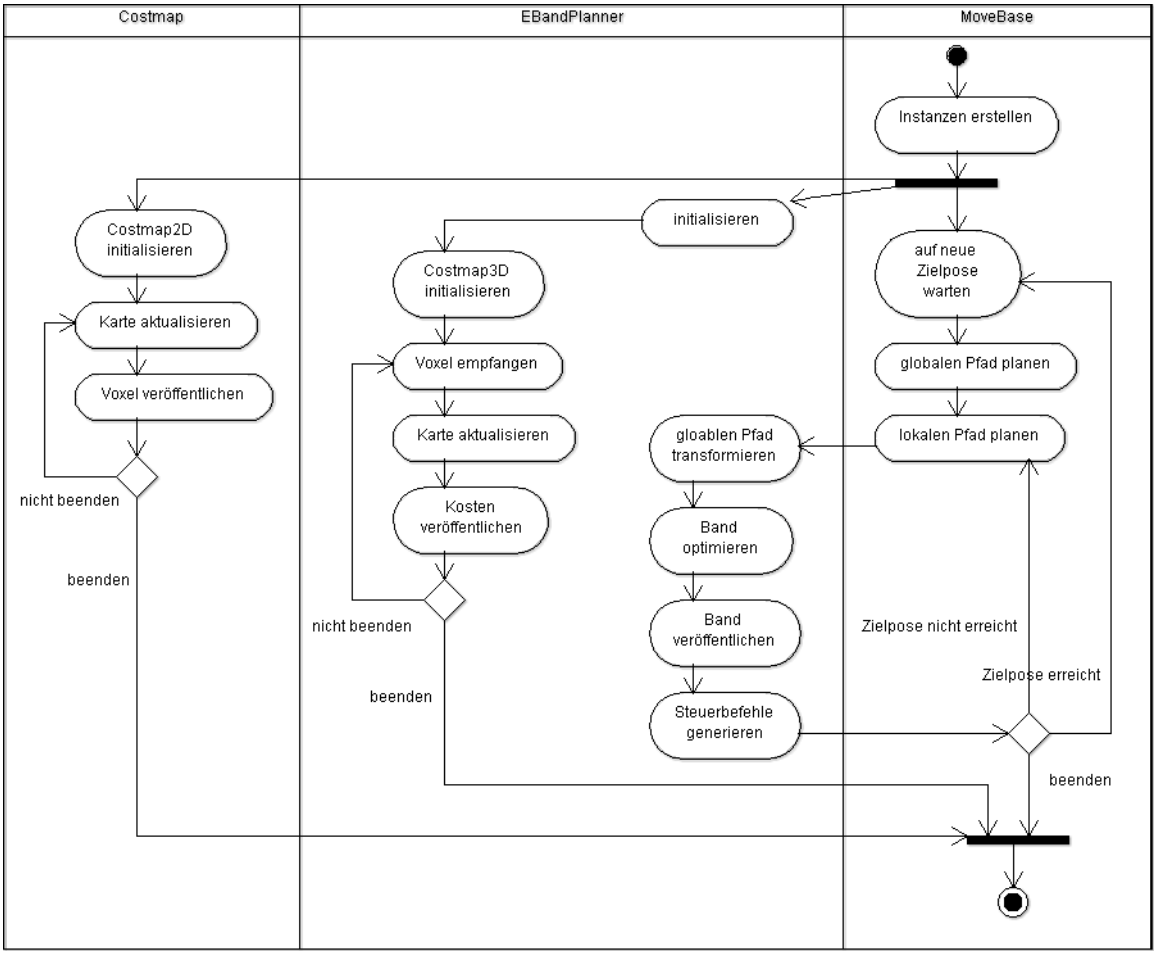


Abbildung 5.2: Aktivitätsdiagramm der Pfadplanung: Dargestellt ist eine Übersicht des Ablaufes des gesamten Pfadplanungsprozesses mit Fokus auf der lokalen Pfadplanung. Teilschritte des Pfadplanungsprozesses sind in Aktivitäten zusammengefasst und der Kontrollfluss vereinfacht dargestellt.

übergeben wurde, wird mit der Instanz von BaseGlobalPlanner ein globaler Pfad zur Zielpose geplant. Zur globalen Pfadplanung wird das ROS-Paket navfn[1] genutzt, das auf eine Costmap2D-Instanz zurückgreift. Die globale Pfadplanung approximiert den Roboter durch den Umkreis seiner Grundfläche und berechnet mithilfe des Algorithmus von Dijkstra einen Pfad mit minimalen Kosten von der Start- zur Zielpose.

Der globale Pfad wird über die EBandPlannerROS-Instanz an die EBandPlanner-Instanz übergeben und dient als initialer Pfad, der lokal modifiziert wird. Zur Optimierung wird auf die Kosten des 3D-Gitters der Costmap3D-Instanz zugegriffen. Nach der Optimierung wird das Band über die Instanz von EBandVisualization veröffentlicht. Anhand des optimierten Bandes werden Steuerbefehle durch die Instanz von TrajectoryController erstellt. Die Optimierung des Bandes wird in einer Schleife wiederholt, bis die Zielpose erreicht wurde, der

[1] http://wiki.ros.org/navfn

Knoten move_base beendet wurde oder kein gültiger lokaler Pfad erstellt werden konnte.

5.3 Umsetzung der Teilschritte

In diesem Abschnitt wird die Umsetzung wichtiger Teilschritte in der Arbeitsweise des implementierten Pfadplanungsverfahrens erläutert. Die Reihenfolge der Unterabschnitte spiegelt die zeitliche Abfolge der Arbeitsweise wider. Die Beschreibung fokussiert die in dieser Arbeit umgesetzten Erweiterungen.

5.3.1 Karte erstellen

Für das Pfadplanungsverfahren wird eine 3D-Erweiterung der Klasse Costmap2D eingesetzt. Eine direkte Erweiterung der Costmap2D durch Vererbung ist aufgrund der Struktur der Klassen nicht praktikabel. Stattdessen setzt sich die verwendete Karte aus zwei Komponenten zusammen. Eingehende Sensordaten werden durch die vorhandene 2,5D-Karte verarbeitet und in dem 3D-Gitter erfasst. Das 3D-Gitter wird durch ROS veröffentlicht. Die Klasse Costmap3D empfängt die verarbeiteten Sensordaten und integriert sie in ein eigenes 3D-Gitter. Die Beziehungen zwischen den Klassen sind in Abbildung 5.2 dargestellt.

Das Aufblähen der Hindernisse und die Berechnung der Kosten wird in der Klasse Costmap3D in drei Dimensionen durchgeführt. In der Klasse Costmap2D richtet sich der Radius zum Aufblähen der Hindernisse nach dem Innenkreisradius der Grundfläche des Roboters. In der dreidimensionalen Erweiterung wird der Roboter nicht durch seine zweidimensionale Grundfläche, sondern durch eine komplexere geometrische Struktur in Form einer Hülle repräsentiert, die aus mehreren Blasen mit unterschiedlichen Radien besteht. Als Radius zum Aufblähen der Hindernisse wird der minimale Radius dieser Blasen gewählt. Für Gitterzellen innerhalb dieses Radius sind die Kosten maximal, außerhalb sinken sie logarithmisch bis zum maximalen Radius der Kostenberechnung.

Costmap3D ist unabhängig von der Quelle der eingehenden Sensordaten und bildet eine Schnittstelle zwischen der Pfadplanung und der Verarbeitung der Sensordaten. Dadurch kann die Implementierung der Karte unabhängig von der Pfadplanung ausgetauscht werden. Die Übertragung der Sensordaten und Integration in das 3D-Gitter einer Instanz von Costmap3D verursacht einen zusätzlichen Mehraufwand beim Aufbau der Karte. Daher ist diese Erweiterung nur für die lokale Karte mit begrenzter Größe geeignet.

5.3.2 Globalen Plan transformieren

Der globale Plan wird als eine Sequenz aus Posen übergeben und muss initial in ein elastisches Band transformiert werden. Dazu wird die Sequenz auf einen Ausschnitt von der aktuellen Roboterpose bis zu einer festgelegten Maximallänge des elastischen Bandes begrenzt. Für jede Pose p_i in der Sequenz wird eine Blase b_i mit p_i als Zentrum erstellt. Die so erstellten Blasen überlappen einander stark, weshalb redundante Blasen nach dem Prinzip der zweiten Optimierungsphase entfernt werden. Anschließend liegt ein elastisches Band vor, das den globalen Pfad abbildet und als Basisband für die lokale Pfadplanung verwendet wird.

5.3.3 Elastischen Streifen erstellen

Eine Teilmenge des Freiraumes um den Roboter entlang des Pfades wird durch den elastischen Streifen repräsentiert. Dieser setzt sich aus einer Menge den Roboter umschließender Hüllen zusammen. Die Hüllen bilden jeweils die Teilmenge des Freiraumes um eine virtuelle Roboterpose ab. Die Hülle der virtuellen Roboterpose wird durch Projektion der Hülle der aktuellen Roboterpose auf eine Blase des Basisbandes erstellt. Bei der ersten Hülle sind virtuelle und aktuelle Roboterpose identisch, da die erste Blase des Basisbandes die Startpose des lokalen Pfades abbildet.

Die statische Hülle definiert eine minimale Teilmenge des Freiraumes um den Roboter, die für eine kollisionsfreie Bewegung des Roboters vorhanden sein muss. Sie wird mithilfe von Blasen erstellt, deren Zentren relativ zu Koordinatensystemen von Gelenken definiert werden. Die Zuordnung zu einem Gelenk, die Abweichung relativ zu dessen Koordinatensystem und der Radius einer Blase werden in einer Konfigurationsdatei spezifiziert. Die Anzahl der Blasen zur Repräsentation des Roboters ist beliebig und wird während der Pfadplanung nicht verändert. Die Posen aller Gelenke sind zur Laufzeit stets bekannt. Da die Blasen relativ zu den Gelenken definiert sind, werden ihre Zentren automatisch aktualisiert und bilden jederzeit die aktuelle Pose des Roboters ab. In dieser Implementierung wird die statische Hülle aus acht Blasen zusammengesetzt: drei für den Roboterkörper, vier für den Manipulator und eine für das Tablett.

Zur Projektion der Hülle einer virtuellen Roboterpose wird Formel 4.1 umgesetzt, indem eine statische Hülle für die Startpose erstellt und auf die restlichen Blasen b_{i0} des Basisbandes kopiert wird. Die relative Abweichung zwischen der Blase der Startpose b_{00} und b_{i0} wird auf alle Blasen b_{ij} der Hülle h_i übertragen, sodass h_i die Hülle der virtuellen Roboterpose an

dem geplanten Wegpunkt i darstellt.

$$b_{ij} = b_{0j} + (b_{i0} - b_{00}), \tag{5.1}$$

wobei i den Index des Basisbandes und j den Index der Blase in einer Hülle angibt.

5.3.4 Kräfte berechnen

In der ersten Phase der Optimierung werden Kräfte für die Blasen $b_{i0} \in \{b_{10}, b_{20}, \ldots, b_{n-10}\}$ des Basisbandes berechnet und auf die Blase angewendet. Dafür werden die externen Kräfte für alle Blasen der jeweiligen Hülle $h_i \in \{h_1, h_2, \ldots, h_{n-1}\}$ berechnet. Die externen Kräfte der Blasen von h_i werden kombiniert und auf die Blase b_{i0} des Basisbandes übertragen. Neben den kombinierten externen Kräften werden die internen Kräfte für die Blasen des Basisbandes berechnet. Die kombinierten externen Kräfte und die internen Kräfte werden addiert und daraus abschließend die resultierende Kraft für die Blase b_{i0} berechnet.

Die Berechnung der externen Kräfte für eine Blase b erfolgt anhand der folgenden Realisierung der Formel 3.8. Für jeden Freiheitsgrad der mobilen Basis wird eine Kraft berechnet. f_{xe} und f_{ye} bezeichnen die Kräfte für die Translation entlang der x- bzw. der y-Achse und $f_{\theta e}$ das Drehmoment um die z-Achse:

$$f_{xe} = -k_e \frac{d\left(b_x - b_r, b_y, b_z, b_\theta\right) - d\left(b_x + b_r, b_y, b_z, b_\theta\right)}{2b_r} \tag{5.2}$$

$$f_{ye} = -k_e \frac{d\left(b_x, b_y - b_r, b_z, b_\theta\right) - d\left(b_x, b_y + b_r, b_z, b_\theta\right)}{2b_r} \tag{5.3}$$

$$f_{\theta e} = -k_e \frac{d\left(b_x, b_y, b_z, b_\theta - \frac{b_r}{r_c}\right) - d\left(b_x, b_y, b_z, , b_\theta + \frac{b_r}{r_c}\right)}{2b_r}. \tag{5.4}$$

Eine Blase ist dabei durch die kartesischen Koordinaten b_x, b_y und b_z ihres Zentrums, ihren Drehwinkel um die z-Achse b_θ sowie ihren Radius b_r charakterisiert. Die Funktion $d : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ berechnet anhand der 3D-Gitterkarte den minimalen Abstand zu Hindernissen. Die Konstanten k_e und r_c repräsentieren einen Gewichtungsparameter bzw. den Außenradius der Roboterbasis. Es werden nur Kräfte für die Freiheitsgrade der Roboterbasis berechnet.

Die folgende Kombination der externen Kräfte bzw. Drehmomente f_{ej} der einzelnen Blasen b_j einer Hülle realisiert Formel 4.2. Dazu werden die Kräfte f_{ej} mit dem Radius b_{rj} der jeweiligen Blase b_j multipliziert und aufsummiert. Die Gewichtung mit dem Radius der

Blase mindert Schwankungen der kombinierten Kraft zwischen zwei Pfadplanungsschritten.

$$f_{xs} = \sum f_{xej} b_{rj} \tag{5.5}$$

$$f_{ys} = \sum f_{yej} b_{rj} \tag{5.6}$$

$$f_{\theta s} = f_{\theta z} + \sum \left(f_{\theta ej} + f_{yej} \left(b_{xj} - b_{x0} \right) - f_{xej} \left(b_{yj} - b_{y0} \right) \right) b_{rj}. \tag{5.7}$$

Das Drehmoment $f_{\theta z}$ bewirkt eine konstante Drehung in Richtung der Zielpose und wird wie folgt ermittelt:

$$f_{\theta z} = k_z \left(z_\theta - b_{\theta 0} \right) \sqrt{\left(z_x - b_{x0} \right)^2 + \left(z_y - b_{y0} \right)^2}, \tag{5.8}$$

wobei k_z einen Gewichtungsparameter und $z = (x, y, \theta)$ die Zielpose darstellt. Dieses zusätzliche Drehmoment wurde eingeführt, um den Roboter während der Planausführung möglichst in Richtung der Zielorientierung zu drehen. Damit wird versucht, den Sichtbereich der 3D-Kamera entlang der Fahrtrichtung auszurichten. Für die aktuelle Planausführung relevante Hindernisse werden somit mit höherer Wahrscheinlichkeit von der 3D-Kamera erfasst. Die Stärke des Drehmoments sinkt mit abnehmender Distanz zur Zielpose, um in der Nähe der Zielpose nicht mit der Rotation des Roboters durch die Planausführung zu interferieren.

Die Formel 3.7 zur Berechnung der internen Kräfte auf eine Blase $b_{i0} \in \{b_{10}, b_{20}, \ldots, b_{n-10}\}$ wird wie folgt umgesetzt:

$$f_{xc} = k_c \left(\frac{b_{xi-1} - b_{xi}}{|b_{i-1} - b_i|} + \frac{b_{xi+1} - b_{xi}}{|b_{i+1} - b_i|} \right) \tag{5.9}$$

$$f_{yc} = k_c \left(\frac{b_{yi-1} - b_{yi}}{|b_{i-1} - b_i|} + \frac{b_{yi+1} - b_{yi}}{|b_{i+1} - b_i|} \right) \tag{5.10}$$

$$f_{\theta c} = k_c \left(\frac{\left(b_{\theta i-1} - b_{\theta i} \right) r_c}{|b_{i-1} - b_i|} + \frac{\left(b_{\theta i+1} - b_{\theta i} \right) r_c}{|b_{i+1} - b_i|} \right). \tag{5.11}$$

Der Operator $|b| = \sqrt{b_x^2 + b_y^2 + b_z^2}$ bezeichnet den Betrag der als Vektor aufgefassten kartesischen Koordinaten des Zentrums einer Blase b. Die Konstante k_c bezeichnet einen Gewichtungsparameter.

Anschließend folgt die Berechnung der Gesamtkraft f_g und der resultierenden Kraft f_r über direkte Umsetzung der Formeln 3.5 und 3.6.

5.3.5 Elastisches Band optimieren

Nachdem die Blasen durch die Anwendung der Kräfte neu positioniert wurden, wird das Basisband als Ganzes optimiert, indem redundante Blasen entfernt und Lücken im Basisband mit neuen Blasen gefüllt werden. Eine Blase gilt als redundant, wenn sie mit einer Blase überlappt, die nicht ihr direkter Nachbar ist. Nachdem eine redundante Blase erkannt und entfernt wurde, werden die Nachbarblasen rekursiv auf Redundanz geprüft.

Wird eine Lücke zwischen zwei Blasen erkannt, wird durch Interpolation eine neue Blase mittig zwischen den beiden Blasenzentren erstellt. Ihr Radius wird auf den minimalen Abstand zum nächsten Hindernis gesetzt. Übersteigt dieser Radius ein definiertes Minimum, wird die Blase in das Basisband eingefügt. Anschließend werden die Abschnitte zwischen der neuen Blase und den ursprünglichen Blasen rekursiv auf Lücken geprüft und ggf. neue Blasen erstellt.

Da in der zweiten Phase der Optimierung nur das Basisband berücksichtigt wird, müssen die zugehörigen Hüllen an Änderungen des Basisbandes angepasst werden. Wurde das Basisband durch die Optimierung verändert, wird der elastische Streifen wie in Abschnitt 5.3.3 beschrieben neu aufgebaut. In der ersten Phase der Optimierung werden die Hüllen direkt berücksichtigt, sodass keine nachträgliche Anpassung notwendig ist.

Durch die Kombination der Kräfte ist die Gültigkeit des optimierten Streifens nicht garantiert. Um Kollisionsfreiheit sicherzustellen, wird der Streifen abschließend auf Kollisionen geprüft. Zur Kollisionsprüfung werden die dynamisch erstellten Hüllen mit der statischen Hülle verglichen, die den minimalen benötigten Freiraum um den Roboter definiert. Dazu wird der aktuelle Radius r_{ij} aller Blasen b_{ij} einer dynamisch erstellten Hülle h_i mit dem Radius r_{mj} der Blasen der statischen Hülle verglichen. Ist der Radius r_{ij} kleiner als der minimal notwendige Radius r_{mj}, liegt eine potenzielle Kollision vor:

$$k_{ij} = \begin{cases} 0 & \text{wenn } r_{mj} \leq r_{ij} \\ 1 & \text{sonst} \end{cases} \tag{5.12}$$

wobei k_{ij} eine potenzielle Kollision der Blase b_{ij} angibt.

5.3.6 Gültigen Zustand wiederherstellen

Das entwickelte Pfadplanungsverfahren verfügt über mehrere Mechanismen, bei Auftreten eines ungültigen Zustandes einen gültigen Zustand wiederherzustellen. Ein wiederherstellba-

Algorithmus 5.1 Wiederherstellung eines gültigen Zustandes bei einer potenziellen Kollision

```
1  boolean checkAndRecoverStrip(hulls) {
2    for each hull in hulls {
3      bubbleNumber = checkCollision(hull);
4      if bubbleNumber != NO_COLLISION {
5        for each recoveryAction in recoveryActions {
6          newForces = recoveryAction.startRecovery(hull, bubbleNumber);
7          applyForces(hull, bubbleNumber, newForces);
8          nextCollision = checkCollision(hull);
9
10         if nextCollision > bubbleNumber
11           break;
12         if recoveryAction == LAST_RECOVERY_ACTION
13           return false;
14       }
15     }
16   }
17   return true;
18 }
```

rer, ungültiger Zustand ist z. B. eine potenzielle Kollision oder ein nicht-kritischer Fehler bei der Ausführung des Algorithmus.

Nach der vollständigen Optimierung des Basisbandes wird eine iterative Kollisionsprüfung des elastischen Streifens durchgeführt. Wird eine potenzielle Kollision einer Blase erkannt, wird versucht, diese durch lokale Wiederherstellungsaktionen zu vermeiden. Über ein Plug-in-basiertes Verfahren kann der Benutzer Wiederherstellungsaktionen spezifizieren, die bei der Initialisierung einer Instanz der Klasse EbandLocalPlanner geladen werden. Die Wiederherstellungsaktionen werden sequentiell ausgeführt, bis die potenzielle Kollision behoben wurde oder alle Wiederherstellungsaktionen scheiterten. Jede Wiederherstellungsaktion kann die Kräfte und Drehmomente manipulieren, die auf die Blasen der betroffenen Hülle wirken. Nach jeder Ausführung wird die Kollisionsprüfung der betroffenen Blase wiederholt, um die Behebung der potenziellen Kollision zu prüfen. Der vereinfachte Ablauf der Kollisionsprüfung ist in Algorithmus 5.1 in Pseudocode skizziert.

Aktuell werden zwei Wiederherstellungsaktionen unterstützt. Eine Instanz der Klasse SingleDirectionRecoveryAction positioniert eine potenziell in Kollision befindliche Blase neu, indem Kräfte ignoriert werden, die die Blase zum Hindernis bewegen. Dazu werden nur die Kräfte in x- oder y-Richtung angewendet, die den größten absoluten Betrag besitzen. Eine Instanz der Klasse RotationRecoveryAction ignoriert alle Kräfte und wendet nur Drehmomente an, um eine Drehung des Roboters in einem beengten Bereich zu ermöglichen.

Kann der ungültige Zustand nicht direkt von der EBandLocalPlanner-Instanz behoben werden, wird ein Fehlerstatus an die EBandLocalPlannerROS-Instanz zurückgegeben. Diese

speichert den Fehlerstatus und gibt ihrerseits einen Fehlerstatus an die Instanz von MoveBase zurück. Daraufhin wird ein Neuplanen des globalen Pfades veranlasst und die lokale Pfadplanung erneut aufgerufen. Bei einer erfolgreichen lokalen Pfadplanung wird der Fehlerstatus aufgehoben und das Programm normal fortgesetzt.

Scheitert die lokale Pfadplanung erneut, kann die MoveBase-Instanz einen ungültigen Zustand über global agierende Wiederherstellungsverhalten auflösen. Dazu wird ein Plug-in-basiertes Verfahren analog zu dem zuvor beschriebenen verwendet. Global agierende Wiederherstellungsverhalten können z. B. die lokale Karte durch Rotation des Roboters um 360 Grad neu aufbauen oder ein globales Neuplanen veranlassen. Ist die Wiederherstellung auf globaler Ebene nicht erfolgreich, wird nach einer festgelegten Anzahl an Fehlversuchen ein Fehler ausgegeben, der Roboter gestoppt und die gesamte Pfadplanung abgebrochen.

5.3.7 Zielgeschwindigkeiten generieren

Beim entwickelten Pfadplanungsverfahren wird wie beim vorhandenen Pfadplanungsverfahren ein Pfad für die mobile Basis der Roboters mit zwei translatorischen und einem rotatorischen Freiheitsgrad geplant. Da diesbezüglich in der gegebenen Aufgabenstellung keine geänderten Anforderungen bestehen, wird die Funktionalität zur Generierung von Zielgeschwindigkeiten vom vorhandenen Pfadplanungsverfahren nicht modifiziert.

Anstatt anhand des lokalen Plans eine Trajektorie zum Zentrum der letzten Blase im Basisband zu generieren, werden nur Zielgeschwindigkeiten für die aktuelle Pose des Roboters berechnet. Ziel ist dabei, den Roboter von der aktuellen Pose zum Zentrum der nächsten Blase des optimierten Basisbandes zu bewegen. Damit wird sichergestellt, dass sich der Roboter stets innerhalb einer Blase und somit innerhalb des Freiraumes befindet. Auf die Berechnung einer vollständigen Trajektorie wird verzichtet, um unnötige Berechnungen zu vermeiden. Bevor eine Trajektorie vollständig ausgeführt werden könnte, würde die lokale Pfadplanung erneut ausgeführt und eine neue Trajektorie berechnet. Durch die Ausführung der lokalen Pfadplanung mit hoher Frequenz ist die Neuberechnung der Zielgeschwindigkeiten zum Erreichen der nächsten Blase ausreichend.

Anhand der nächsten Blase b_1 im optimierten Basisband werden Zielgeschwindigkeiten für die Translation entlang der x- und der y-Achse sowie für die Rotation um die z-Achse berechnet. Dafür werden zunächst die Differenzen der Freiheitsgrade zwischen der aktuellen Roboterpose bzw. dem Zentrum der ersten Blase des Basisbandes b_0 und dem Zentrum von b_1 ermittelt. Die Differenzen werden als Geschwindigkeiten interpretiert und anhand der euklidischen Distanz zwischen b_0 und b_1 skaliert. Um den Roboter in die Orientierung der Zielpose zu rotieren,

wird die Differenz zur Orientierung der letzten Blase im Basisband ermittelt. Ein PID-Regler modifiziert die Zielgeschwindigkeiten, um die Rotation zu realisieren. Abschließend werden die berechneten Zielgeschwindigkeiten anhand der minimalen und maximalen Geschwindigkeits- und Beschleunigungsgrenzwerte des Roboters skaliert.

Die berechneten Zielgeschwindigkeiten werden über die Instanz von EBandPlannerROS an die MoveBase-Instanz übergeben. Die Zielgeschwindigkeiten werden von der MoveBase-Instanz veröffentlicht und von der Hardwareabstraktionsschicht verarbeitet. Diese passt die Zielgeschwindigkeiten an die verwendete Hardware an und erstellt konkrete Steuerbefehle, die von der Hardware ausgeführt werden.

6 Evaluierung

Das entwickelte Pfadplanungsverfahren wird anhand von vier Testszenarien evaluiert. Dabei werden Leistungsmerkmale zur Beurteilung der Güte der erstellten Pfade erfasst und mit dem vorhandenen 2D-Pfadplanungsverfahren verglichen. Die Durchführung der Tests und die erfassten Leistungsmerkmale werden in Abschnitt 6.1 beschrieben. Auf die Ermittlung der Testergebnisse wird in Abschnitt 6.2 eingegangen. In den Abschnitten 6.3 bis 6.6 werden Aufbau und Zielstellung der einzelnen Testszenarien erläutert sowie die Ergebnisse der Durchführung präsentiert und analysiert. In Abschnitt 6.7 wird eine Gesamtauswertung der Tests vorgenommen.

6.1 Testablauf

Die Testszenarien definieren eine Start- und Zielpose, für die das Pfadplanungsverfahren einen geeigneten Pfad erstellen soll. Dabei wird jeweils der Fokus auf ein spezifisches Problem gelegt. Die Testszenarien wurden jeweils in vier unterschiedlichen Roboterposen absolviert, die Einfluss auf das erwartete Verhalten haben. Die verwendeten Roboterposen sind in Abbildung 6.1 dargestellt.

Die Tests werden automatisch über Skripte ausgeführt. Für jede Kombination von Testszenario und Roboterpose werden 50 Testdurchläufe ausgewertet. Der Roboter, die Start- und Zielpose und die erstellten Hindernisse werden vor jedem Testdurchlauf in eine definierte Ausgangsposition gebracht. Somit wird die Vergleichbarkeit der einzelnen Testdurchläufe gewährleistet. Jeder Testdurchlauf wird nach Erreichen eines Zeitlimits beendet, unabhängig davon, ob der Roboter die Zielpose erreicht hat.

Die Tests werden in der Simulationsumgebung Gazebo ausgeführt. Gazebo simuliert den Care-O-bot® 3, eine häusliche Umgebung sowie für jedes Testszenario unterschiedliche Hindernisse. Daten zum Status des Roboters und des geplanten sowie des ausgeführten Pfads werden über ROS ausgegeben und zur Analyse gespeichert. Die Tests werden über die graphische Benutzeroberfläche von Gazebo und über das Visualisierungsprogramm Rviz manuell überwacht.

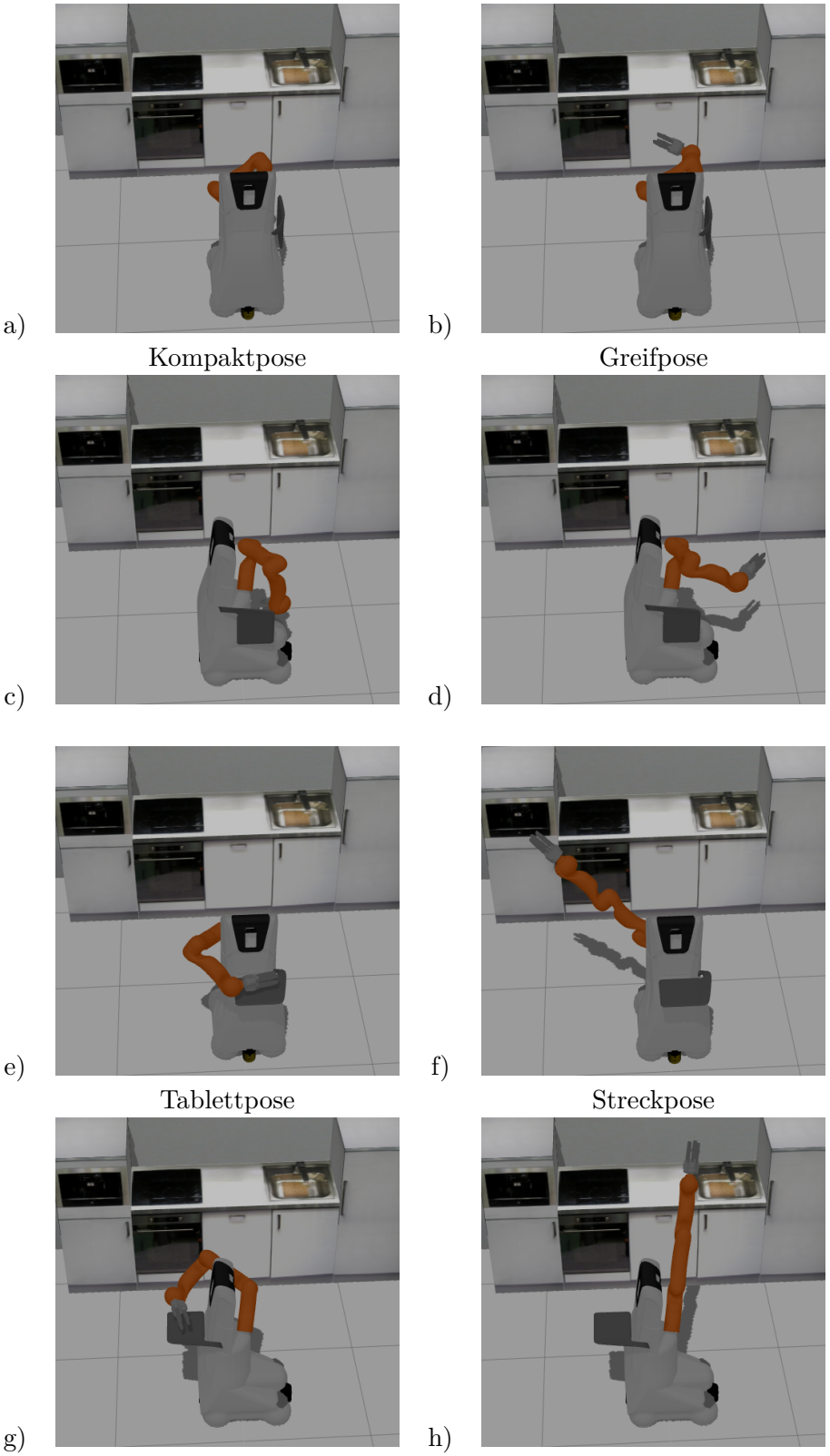


Abbildung 6.1: Roboterposen: Das geringste Volumen nimmt der Roboter in Kompaktpose in a) und c) ein. In Greifpose in b) und d) befindet sich der Manipulator in einer Pose zum einfachen Greifen von Objekten. In e) und g) ist der Roboter in Tablettpose mit ausgeklapptem Tablett und dem Manipulator über dem Tablett abgebildet. Durch den gestreckten, seitlich abstehenden Manipulator in f) und h) ist der Roboter in Streckpose für Kollisionen anfällig.

Zur Analyse der erstellten Pfade werden folgende Leistungsmerkmale betrachtet:

- Fahrzeit: Die benötigte Zeit zum Erreichen der Zielpose sollte minimal sein.
- Gesamtlänge: Zur Minimierung der Fahrzeit und der Positionsungenauigkeit sollte die zurückgelegte Wegstrecke minimiert werden.
- Gesamtrotation: Wenige Rotationen reduzieren die Positionsungenauigkeit der Lokalisierung.
- Abstand zu Hindernissen: Während kurze Pfade oft dicht an Hindernissen vorbei führen, verringert ein großer Abstand zu Hindernissen die Kollisionswahrscheinlichkeit.
- Geschwindigkeitsverlauf: Für ein stabiles Fahrverhalten sind abrupte Beschleunigungen zu vermeiden und gleichmäßige Geschwindigkeiten zu bevorzugen.
- Häufigkeit von Wiederherstellungsmaßnahmen: Wiederherstellungsmaßnahmen unterbrechen den normalen Fahrtverlauf und erhöhen den benötigten Rechenaufwand.
- Speicherbedarf: Um die Anforderungen an die Hardware zu minimieren, sollte der Speicherbedarf gering gehalten werden.

Weitere mögliche Leistungsmerkmale bzw. Anforderungen an ein Pfadplanungsverfahren sind in [Hop92, S. 37ff.] zu finden.

6.2 Testergebnisse

Die Ergebnisse der Testszenarien sind jeweils in zwei Tabellen zusammengefasst. Die ersten vier Reihen zeigen die Ergebnisse des entwickelten Pfadplanungsverfahrens in der angegebenen Roboterpose, die nachfolgenden vier Reihen die Ergebnisse des vorhandenen Pfadplanungsverfahrens. Jede Ergebniszelle zeigt für das jeweilige Leistungsmerkmal den Mittelwert der Ergebnisse aller Testdurchläufe und in Klammern die zugehörige Standardabweichung.

Tabelle 6.1 gibt Auskunft über den jeweiligen Wert aller Leistungsmerkmale, bezogen auf einen Testdurchlauf. Ein Testdurchlauf gilt als erfolgreich, wenn der Roboter die Zielpose erreicht hat und keine Kollisionen auftraten.

Tabelle 6.1: Beschreibung der Leistungsmerkmale

Leistungsmerkmal	Beschreibung
Kollision	1, wenn während eines Testdurchlaufs eine oder mehrere Kollisionen auftraten, andernfalls 0

Ziel erreicht	1, wenn der Roboter die vorgegebene Zielpose innerhalb einer Toleranz von 12 cm und 0,1 rad erreichte, andernfalls 0
Strecke	gefahrene Gesamtstrecke in m
Rotation	Summe der ausgeführten Rotationen in rad
Zeit	benötige Fahrzeit zum Erreichen der Zielpose bzw. bis zum Abbruch der Pfadplanung in s
max. Speicher	maximaler Arbeitsspeicherbedarf des Knotens move_base in KiB
min. Abst	minimaler Abstand des Roboters zu Hindernissen in m
mittl. Abst	mittlerer Abstand des Roboters zu Hindernissen in m
mittl. Beschl.	mittlere Beschleunigung bzw. Verzögerung in m/s^2
max. Beschl.	maximale Beschleunigung bzw. Verzögerung in m/s^2
lok. Wied.	Anzahl der ausgeführten lokalen Wiederherstellungsaktionen (nur entwickelte Pfadplanung)
glob. Wied.	Anzahl der ausgeführten globalen Wiederherstellungsverhalten

Die Ergebnisse wurden durch eine Analyse von aufgezeichneten Daten der Simulationsumgebung ermittelt. Die Ergebnisse sind aufgrund von Ungenauigkeiten der simulierten Daten und der nachträglichen Analyse nicht exakt. Aufgrund des hohen Datenvolumens beschränkte sich die Aufzeichnung für die Ermittlung von Abständen auf die Pose der Roboterbasis. Die Abstandsberechnung erfolgte anhand der statischen Hülle, die für die Analyse approximiert wurde. Die Abstände des Roboters zu Hindernissen und der Zielpose sowie die Kollisionsberechnung beziehen sich daher auf die statische Hülle und sind nur auf ca. 5 cm genau.

Das Zeitlimit jedes Testdurchlaufes wurde für jede Kombination von Testszenario und Roboterpose konstant gewählt. Aufgrund von Schwankungen des Verhältnisses zwischen realer und simulierter Zeit durch die Simulationsumgebung stand dennoch nicht jedem Testdurchlauf die exakt gleiche simulierte Zeit zur Verfügung. Zur Ermittlung des realen Arbeitsspeicherbedarfs der Umgebungsrepräsentation wurde auf Funktionalitäten des Betriebssystems zurückgegriffen. Dabei konnte nur der Arbeitsspeicherbedarf des Knotens move_base gemessen werden, nicht der Arbeitsspeicherbedarf der Umgebungsrepräsentation.

6.3 Pfadplanung in statischer Umgebung

Im ersten Testszenario ist eine statische Umgebung mit einem unbeweglichen Hindernis gegeben.

a) 

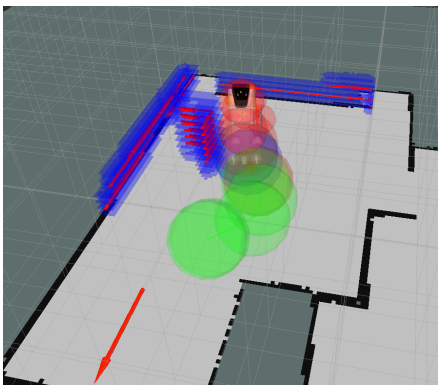
b) 

Abbildung 6.2: Testszenario statische Umgebung: Teilabbildung a) zeigt einen Überblick über den Aufbau des Testszenarios, visualisiert durch die Simulationsumgebung Gazebo. Der rote Pfeil markiert die Zielpose. Teilabbildung b) zeigt den Roboter und die Umgebungsrepräsentation bei der Umfahrung des Hindernisses auf dem linken Weg, dargestellt in dem Visualisierungsprogramm Rviz. Der elastische Streifen ist durch grüne Blasen und die statische Hülle durch rote Blasen repräsentiert.

6.3.1 Aufbau und Zielstellung

Ziel des Testszenarios ist die Durchquerung eines Raumes. Der direkte Weg wird durch ein statisches Hindernis blockiert, das es zu umfahren gilt. Das Hindernis „schwebt“, wodurch ein sich an der Decke befindliches Hindernis simuliert wird. Der Raum zwischen Boden und Hindernis kann befahren werden. Der Aufbau des Testszenarios ist in Abbildung 6.2 dargestellt. Das Hindernis kann auf zwei Wegen umfahren werden. Der rechte Weg ist kürzer als der linke Weg. Der zur Verfügung stehende Bereich wird durch eine Wand begrenzt und kann nicht in Streckpose passiert werden. Der linke Weg ist in allen Roboterposen befahrbar. Wenn die Roboterpose ein kollisionsfreies Erreichen der Zielposition verhindert, muss der Roboter anhalten, bevor eine Kollision auftritt.

Mit dem Testszenario wird das Verhalten des Pfadplanungsverfahrens in einer statischen Umgebung untersucht. Das Pfadplanungsverfahren soll wenn möglich einen Pfad für den kürzeren rechten Weg erstellen. Der Raum zwischen Hindernis und Wand ist eng bemessen und erfordert ein präzises Navigieren, um eine Kollision zu vermeiden. Weiterhin muss das Pfadplanungsverfahren in Streckpose erkennen, dass der rechte Weg nicht passierbar ist und ggf. eine Neuplanung des globalen Pfades anstoßen.

6.3.2 Ergebnisse

Die Tabellen 6.2 und 6.3 zeigen die Ergebnisse des Testszenarios Pfadplanung in statischer Umgebung.

Tabelle 6.2: Ergebnisse statische Umgebung

	Kollision	Ziel erreicht	Strecke	Rotation	Zeit	max. Speicher
Kompakt	0	1	3.807 (0.023)	0.318 (0.048)	14.374 (0.258)	105102 (2783)
Greif	0	0.98	3.894 (0.665)	0.545 (0.364)	15.545 (3.986)	110028 (3521)
Tablett	0	1	3.744 (0.018)	0.302 (0.035)	14.232 (0.307)	102178 (2796)
Streck	0.5	0.26	2.954 (1.848)	0.626 (0.444)	19.897 (8.690)	106334 (5315)
Kompakt 2D	0	0.80	4.190 (0.923)	1.032 (0.496)	20.208 (7.114)	101490 (3776)
Greif 2D	0.12	0.80	4.402 (1.014)	1.005 (0.469)	21.606 (6.782)	105163 (3006)
Tablett 2D	0	0.98	3.844 (0.544)	0.293 (0.236)	19.414 (2.817)	100656 (3296)
Streck 2D	1	0	1.554 (0.502)	0.262 (0.091)	6.929 (4.651)	110704 (4746)

Tabelle 6.3: Ergebnisse statische Umgebung (Fortsetzung)

	min. Abst.	mittl. Abst.	mittl. Beschl.	max. Beschl.	lok. Wied.	glo. Wied.
Kompakt	0.001 (0.003)	0.683 (0.009)	0.318 (0.008)	2.863 (2.459)	2.18 (3.60)	0.04 (0.28)
Greif	0.014 (0.050)	0.649 (0.038)	0.162 (0.059)	3.374 (3.411)	7.14 (8.01)	0.88 (1.61)
Tablett	0.002 (0.006)	0.573 (0.013)	0.199 (0.010)	2.158 (1.193)	0.00 (0.00)	0.00 (0.00)
Streck	0.084 (0.175)	0.553 (0.087)	0.167 (0.098)	4.515 (3.560)	296.64 (361.30)	80.20 (94.97)
Kompakt 2D	0.253 (0.149)	0.778 (0.084)	0.075 (0.066)	2.856 (2.463)	-	1.45 (3.71)
Greif 2D	0.168 (0.108)	0.752 (0.038)	0.054 (0.055)	3.288 (5.481)	-	1.40 (3.62)
Tablett 2D	0.010 (0.063)	0.518 (0.038)	0.197 (0.026)	2.510 (1.621)	-	0.08 (0.27)
Streck 2D	0.001 (0.001)	0.516 (0.049)	0.233 (0.062)	3.862 (3.239)	-	10.14 (3.31)

6.3.3 Analyse

In Kompakt-, Greif- und Tablettpose agierte das entwickelte Pfadplanungsverfahren zuverlässig. Die um ca. 28 % geringere benötigte Zeit als beim vorhandenen Pfadplanungsverfahren weist auf eine höhere Durchschnittsgeschwindigkeit hin. Beim entwickelten Pfadplanungsverfahren wurde stets der kürzere, rechte Weg genutzt. In Kompakt- und Greifpose fuhr der Roboter beim vorhandenen Pfadplanungsverfahren auf dem linken Weg um das Hindernis. Dies hatte größere Abstände zu Hindernissen und längere Gesamtstrecken zur Folge.

Schwierige Umstände und Defizite bestehen bei der Pfadplanung in Streckpose. Der globale Pfad führte initial fast ausschließlich rechts am Hindernis vorbei. Da dieser Pfad durch den ausgestreckten Manipulator nicht vollständig kollisionsfrei ist, wurde globales Neuplanen in direkter Nähe zum Hindernis notwendig. Beim neu geplanten globalen Pfad war der Manipulator zum Hindernis ausgerichtet, was eine erhöhte Kollisionswahrscheinlichkeit und eine komplizierte Umfahrung des Hindernisses mit häufigen Rotationen zur Folge hatte. Dies zeigt sich unter anderem an der hohen Anzahl lokaler und globaler Wiederherstellungsmaßnahmen im Vergleich zu den anderen Testszenarien. Weiteres Indiz ist die höhere Gesamtrotation von 0.626 rad, obwohl die Zielpose nur zu 26% erreicht wurde und die zurückgelegte Strecke entsprechend geringer ausfiel.

Die Gründe für das Fehlschlagen der Testdurchläufe in Streckpose sind vielfältig. Aufgrund des begrenzten Erfassungsbereiches des 3D-Sensors wurde das Hindernis an der Wand teilweise spät erkannt. Dadurch wurde die globale Neuplanung in einer Pose ausgeführt, in der das zentrale Hindernis aufwendig umfahren werden musste. In einigen Testdurchläufen nahm die lokale Wiederherstellung durch wiederholte Ausführung in Kombination mit globalem Neuplanen viel Zeit in Anspruch. Das Zeitlimit des Testdurchlaufs wurde im Zuge dessen z. T. überschritten und führte zum Fehlschlag des Testdurchlaufs. In anderen Testdurchläufen wurde ein globales Wiederherstellungsverhalten aufgerufen, das die lokale Karte neu aufbauen sollte. Dies führte zu einer hohen Positionsungenauigkeit der Lokalisierung und zum Fehlschlag des Testdurchlaufs. In seltenen Fällen wurde die Teilmenge des Freiraumes zwischen der Zielpose und dem Hindernis nicht korrekt durch Costmap2D dargestellt. Die Zielpose wurde dadurch nicht im Freiraum verortet und konnte deshalb nicht erreicht werden.

Das vorhandene Pfadplanungsverfahren verursachte in Streckpose in 100 % der Testdurchläufe eine Kollision. Da der ausgestreckte Manipulator über die Grundfläche hinausragte, wurde das Hindernis an der Wand trotz korrekter Erfassung nicht ausreichend berücksichtigt. Die durchschnittlich gefahrene Strecke ist beim vorhandenen Pfadplanungsverfahren geringer, da die Pfadplanung häufig vor Erreichen der Zielpose abgebrochen wurde.

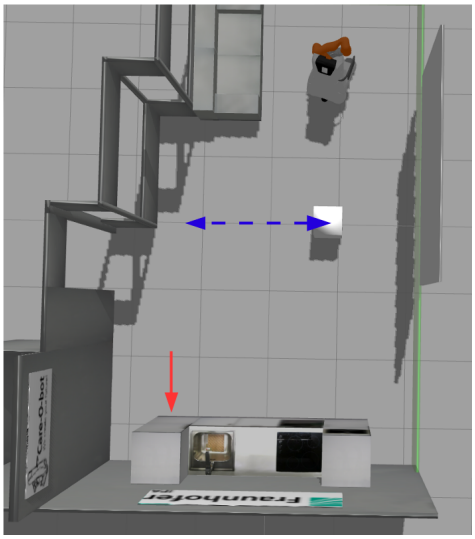
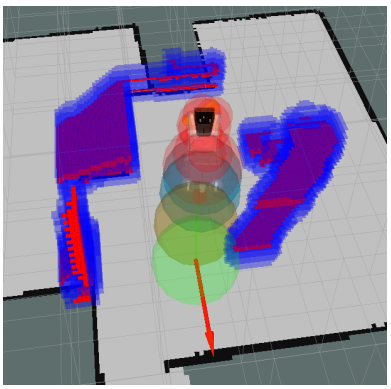
a)  b) 

Abbildung 6.3: Testszenario dynamische Umgebung: Die Ausgangssituation des Testszenarios ist in a) dargestellt. Der rote Pfeil repräsentiert die Zielpose des Roboters, der blaue Pfeil die Bewegung des Hindernisses. Die Umgebungsrepräsentation während eines Testdurchlaufes wird in b) gezeigt. Ein erfasstes Hindernis wird durch rote Voxel dargestellt, blaue Voxel zeigen den aufgeblähten Bereich um das Hindernis. Deutlich zu erkennen ist die fehlerhafte Erfassung des dynamischen Hindernisses.

6.4 Pfadplanung in dynamischer Umgebung

In diesem Testszenario wird eine dynamische Umgebung durch ein sich bewegendes Hindernis simuliert.

6.4.1 Aufbau und Zielstellung

Der Roboter soll einen Raum durchqueren, wobei die Zielpose initial auf direktem Weg durch eine Gerade erreichbar ist. Der Pfad des Roboters wird während der Planausführung durch die Bewegung des Hindernisses gekreuzt. Das Hindernis bewegt sich mit konstanter Geschwindigkeit auf einer Geraden quer zum Pfad des Roboters und pendelt zwischen zwei Positionen. Die Ausgangssituation und der Pfad des Hindernisses ist in Abbildung 6.3 dargestellt.

Mit diesem Testszenario wird das Verhalten des Pfadplanungsverfahrens auf sich bewegende Hindernisse untersucht. Der Roboter muss das Hindernis während der Planausführung erfassen und seinen Pfad dynamisch anpassen. Die Geschwindigkeit des Hindernisses erlaubt es dem Roboter, dem Hindernis in Fahrtrichtung des initialen Pfades auszuweichen. Im Fall einer zeitlichen Verzögerung wird der initiale Pfad unpassierbar und ein globales Neuplanen wird erforderlich.

6.4.2 Ergebnisse

Die Tabellen 6.4 und 6.5 zeigen die Ergebnisse des Testszenarios Pfadplanung in dynamischer Umgebung.

Tabelle 6.4: Ergebnisse dynamische Umgebung

	Kollision	Ziel erreicht	Strecke	Rotation	Zeit	max. Speicher
Kompakt	0.00	0.88	4.714 (0.732)	0.137 (0.388)	19.329 (3.622)	109327 (10298)
Greif	0.00	0.76	4.566 (1.020)	0.261 (0.588)	18.705 (4.924)	117512 (5177)
Tablett	0.04	0.78	4.347 (1.058)	0.150 (0.409)	17.581 (4.510)	211632 (115174)
Streck	0.86	0.20	4.534 (0.737)	0.189 (0.386)	21.553 (6.428)	116115 (5278)
Kompakt 2D	0.20	0.02	4.104 (1.348)	0.450 (0.438)	31.552 (10.124)	108335 (6265)
Greif 2D	0.14	0.06	3.728 (1.270)	0.426 (0.346)	29.373 (9.345)	105908 (4962)
Tablett 2D	0.24	0.08	4.068 (1.420)	0.667 (0.661)	17.297 (12.729)	106691 (6318)
Streck 2D	0.52	0.00	4.094 (1.160)	0.409 (0.119)	28.705 (8.102)	109932 (6198)

Tabelle 6.5: Ergebnisse dynamische Umgebung (Fortsetzung)

	min. Abst.	mittl. Abst.	mittl. Beschl.	max. Beschl.	lok. Wied.	glo. Wied.
Kompakt	0.202 (0.056)	0.894 (0.077)	0.093 (0.091)	4.572 (6.836)	13.06 (58.09)	4.74 (15.62)
Greif	0.172 (0.065)	0.871 (0.172)	0.128 (0.096)	5.856 (9.434)	15.16 (61.00)	7.3 (19.98)
Tablett	0.138 (0.093)	0.745 (0.081)	0.171 (0.078)	7.585 (17.060)	0.44 (1.33)	2.66 (5.33)
Streck	0.011 (0.033)	0.606 (0.121)	0.226 (0.085)	8.019 (10.928)	5.38 (28.41)	3.78 (11.09)
Kompakt 2D	0.054 (0.077)	0.618 (0.105)	0.188 (0.091)	12.744 (22.744)	-	7.62 (5.64)
Greif 2D	0.056 (0.064)	0.606 (0.101)	0.426 (0.069)	10.423 (12.614)	-	9.32 (5.93)
Tablett 2D	0.102 (0.132)	0.614 (0.167)	0.191 (0.098)	17.297 (37.837)	-	7.2 (5.28)
Streck 2D	0.028 (0.048)	0.551 (0.069)	1.592 (3.011)	324 (988)	-	6.46 (6.09)

6.4.3 Analyse

Im Großteil der Testdurchläufe wurde das Hindernis rechts umfahren und die Zielpose erreicht. Konnte der Roboter das Hindernis aufgrund von Verzögerungen nicht rechtzeitig rechts umfahren, wurde die Pfadplanung oft abgebrochen. Verzögerungen entstanden, da der hinter dem Hindernis liegende Freiraum durch die erweiterte Costmap2D wegen fehlerhaftem Raytracing nicht erfasst wurde. Das Hindernis wurde daher verzerrt und der Pfad aufgrund der Verzerrung unnötig korrigiert bzw. die Pfadplanung abgebrochen, weil die Zielpose nicht mehr innerhalb des erfassten Freiraumes lag. Die Verzerrung des Hindernisses ist in Abbildung 6.3 dargestellt. Globales Neuplanen war durch die Verzerrung nicht erfolgreich.

Wenn sich der Roboter aufgrund des Hindernisses nah entlang den Wänden bewegte, ragte der Manipulator in Streckpose teilweise durch die Öffnung der Wände. Dies führte zu einer Kollision, wenn sich der Roboter weiter bewegte und die durchgängige Wandecke nicht erkannt wurde.

Mit dem vorhandenen Pfadplanungsverfahren konnte das Hindernis selten umfahren werden, bevor der Pfad rechts um das Hindernis blockiert wurde. In Hindernisnähe verringerte der Roboter seine Geschwindigkeit. Teilweise kreuzte der Roboter den Pfad des Hindernisses, wies zum Ausweichen eine zu geringe Geschwindigkeit auf und kollidierte seitlich mit dem Hindernis. In anderen Testdurchläufen stoppte der Roboter vor dem Hindernis und führte ein globales Neuplanen aus. Das globale Neuplanen scheiterte in einer kurzen Zeitspanne und die Pfadplanung wurde abgebrochen.

Das vorhandene Pfadplanungsverfahren zeigt im Vergleich zum entwickelten Pfadplanungsverfahren höhere durchschnittliche und maximale Beschleunigungen. Die stark erhöhten Beschleunigungswerte in Streckpose lassen aufgrund der hohen Standardabweichung auf einen Messfehler schließen.

6.5 Pfadplanung in unstrukturierter Umgebung

Dieses Testszenario simuliert durch mehrere Hindernisse in verschiedenen Ausrichtungen und Höhen eine unstrukturierte Umgebung.

6.5.1 Aufbau und Zielstellung

In der Ausgangssituation des Testszenarios verdecken sich die Hindernisse für die Sensoren des Roboters teilweise gegenseitig. Somit können sie von der lokalen Pfadplanung z. T.

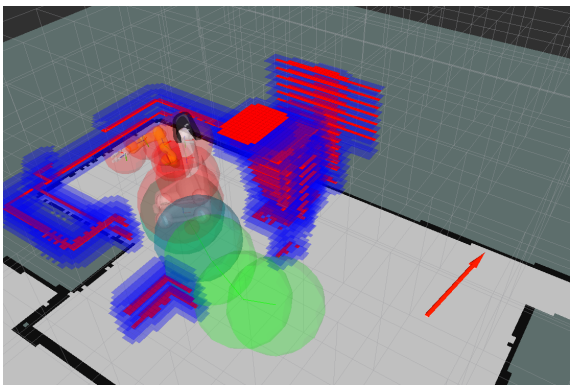
a)  b) 

Abbildung 6.4: Testszenario unstrukturierte Umgebung: Die Anordnung der Hindernisse in der Umgebung ist in a) dargestellt. In b) ist der begrenzte Umfang der lokalen Umgebungsrepräsentation zu erkennen. Die hinteren, zunächst verdeckten Hindernisse und Hindernisse außerhalb des Erfassungsbereiches der Sensoren sind nicht repräsentiert. Sie werden erst während der Planausführung erfasst.

erst während der Planausführung erfasst werden. Zum Erreichen der Zielpose müssen mehrere Hindernisse umfahren werden. Die Anordnung der Hindernisse ist in Abbildung 6.4 dargestellt.

Dieses Testszenario dient zur Untersuchung des Verhaltens des Pfadplanungsverfahrens bei mehreren, z. T. unbekannten Hindernissen. Durch die unterschiedliche Positionierung der Hindernisse entlang der z-Achse muss ihre Lage und der Freiraum in drei Raumdimensionen erfasst werden. Die Bewegungsfreiheit des Roboters wird vor allem in Streckpose durch die Hindernisse eingeschränkt. Die Positionierung der Hindernisse gibt einen Weg zur Zielpose vor, der in allen Roboterposen passiert werden kann. Alternativen durch globale Neuplanung sind nicht möglich.

6.5.2 Ergebnisse

Die Tabellen 6.6 und 6.7 zeigen die Ergebnisse des Testszenarios Pfadplanung in unstrukturierter Umgebung.

Tabelle 6.6: Ergebnisse unstrukturierte Umgebung

	Kollision	Ziel erreicht	Strecke	Rotation	Zeit	max. Speicher
Kompakt	0.26	0.98	4.815 (0.537)	0.367 (0.252)	26.000 (3.580)	109955 (3110)
Greif	0.00	0.98	4.848 (0.547)	0.347 (0.251)	25.733 (3.559)	115111 (4843)
Tablett	0.12	0.96	4.859 (0.774)	0.377 (0.254)	26.657 (4.198)	107158 (2110)
Streck	0.04	0.96	4.933 (0.307)	0.409 (0.414)	27.367 (1.958)	106606 (2778)
Kompakt 2D	0.22	0.28	3.877 (1.960)	0.740 (0.454)	33.819 (16.133)	114623 (37018)
Greif 2D	0.58	0.36	3.768 (1.819)	0.742 (0.408)	33.619 (15.827)	108659 (6320)
Tablett 2D	0.48	0.52	4.088 (1.840)	0.766 (0.393)	34.861 (15.458)	114566 (15450)
Streck 2D	0.56	0.48	4.230 (1.927)	0.908 (0.485)	39.625 (18.402)	113616 (6225)

Tabelle 6.7: Ergebnisse unstrukturierte Umgebung (Fortsetzung)

	min. Abst.	mittl. Abst.	mittl. Beschl.	max. Beschl.	lok. Wied.	glo. Wied.
Kompakt	0.009 (0.061)	0.560 (0.045)	0.054 (0.053)	2.840 (2.377)	0.00 (0.00)	0.08 (0.44)
Greif	0.011 (0.051)	0.546 (0.046)	0.062 (0.067)	3.088 (2.879)	0.00 (0.00)	0.08 (0.44)
Tablett	0.014 (0.061)	0.504 (0.060)	0.069 (0.057)	3.229 (2.719)	0.00 (0.00)	0.22 (0.90)
Streck	0.003 (0.007)	0.516 (0.035)	0.188 (0.082)	4.452 (3.439)	0.00 (0.00)	0.12 (0.59)
Kompakt 2D	0.104 (0.163)	0.531 (0.092)	0.093 (0.087)	3.598 (5.037)	-	4.64 (5.37)
Greif 2D	0.084 (0.154)	0.483 (0.096)	0.098 (0.085)	3.315 (2.346)	-	4.36 (5.35)
Tablett 2D	0.084 (0.167)	0.482 (0.102)	0.120 (0.089)	3.389 (2.611)	-	2.86 (4.77)
Streck 2D	0.063 (0.154)	0.516 (0.114)	0.136 (0.089)	4.345 (3.185)	-	2.96 (4.52)

6.5.3 Analyse

Aufgrund der starken Auswirkung des in Abschnitt 6.4.3 beschriebenen Fehlers beim Raytracing wurde die erweiterte Costmap2D für das entwickelte Pfadplanungsverfahren in diesem Testszenario leicht modifiziert. Durch die Verzerrung der Hindernisse lag die Zielpose in der Umgebungsrepräsentation nicht innerhalb des Freiraumes und die Pfadplanung wurde während der Bewegung abgebrochen. Die Ergebnisse und die Analyse beziehen sich auf Testdurchläufe mit der Modifizierung.

Das entwickelte Pfadplanungsverfahren konnte die Zielpose in 97 % der Testdurchläufe erreichen. Die geringe Häufigkeit von Wiederherstellungsmaßnahmen und die kleinen Beschleunigungswerte belegen eine gleichmäßige Bewegung in allen Roboterposen. Im Vergleich zu dem vorhandenen Pfadplanungsverfahren benötigte der Roboter ca. 25 % weniger Zeit und rotierte ca. 52 % weniger.

Die hohe Kollisionsrate von 26 % in Kompaktpose entstand durch die Blase um die Basis, die das kleine zentrale Hindernis auf der Ebene streifte. Der Roboter erreichte die Zielpose im Kollisionsfall zu 100 %. Die Kollisionen können daher auf Messungenauigkeiten zurückzuführen sein oder nur die Blase betreffen, nicht den simulierten Roboter. Analog verhält es sich in Tablettpose, in der die Blase der Basis das Hindernis nahe der Couch streifte und stets die Zielpose erreichte.

Das vorhandene Pfadplanungsverfahren rotierte den Roboter im Bereich vor den Hindernissen oft um ca. 90 Grad und steuerte den Roboter quer zur Ausrichtung des Sensorkopfes durch die Hindernisse. Dadurch kam es in Greif- und in Streckpose zu Kollisionen, da der Manipulator in dieser Pose über die Grundfläche des Roboters hinausragte. In einigen Testdurchläufen wurde die Zielposition erreicht, aber die Orientierung des Roboters wich von der Zielorientierung ab.

6.6 Pfadplanung durch eine Türöffnung

Im letzten Testszenario soll eine Türöffnung in einer statischen Umgebung ohne zusätzliche Hindernisse durchquert werden.

6.6.1 Aufbau und Zielstellung

Die zu durchquerende Türöffnung wird durch mehrere Wände begrenzt. Die Breite der Türöffnung entspricht der einer durchschnittlichen Bürotür und ist geringfügig breiter als

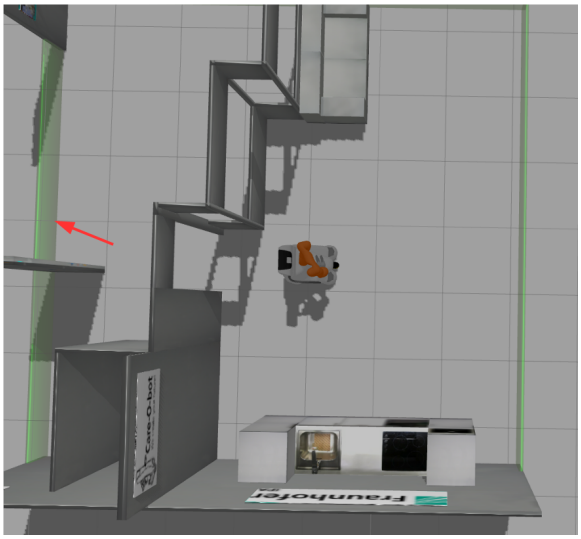
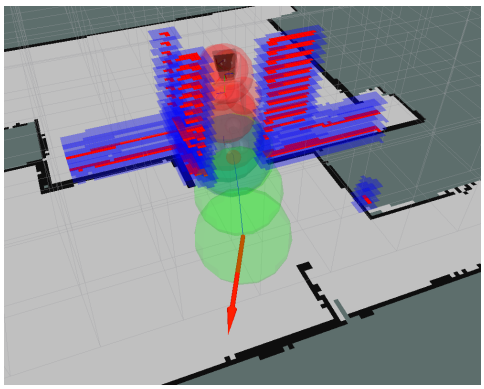
a)  b) 

Abbildung 6.5: Testszenario Durchquerung einer Türöffnung: Die Start- und Zielpose sind in a) dargestellt. An dem Pfad während eines Testdurchlaufs in b) ist der geringe Spielraum für den Roboter zu erkennen.

die maximale Breite des Roboters. In Streckpose kann der Roboter die Türöffnung nicht passieren. Die Ausgangssituation ist in Abbildung 6.5 visualisiert.

Mit der Durchquerung einer Türöffnung wird eine typische Aufgabenstellung für einen mobilen Serviceroboter simuliert. Der Roboter muss die Türöffnung möglichst geradlinig durchqueren und die Rotation zur gewünschten Orientierung in der Zielpose nach der Durchquerung ausführen. Durch den geringen Spielraum muss der Roboter die begrenzenden Wände präzise erfassen und repräsentieren. Dabei entstehen zusätzliche Schwierigkeiten, da die Wände in der Karte nur partiell als Hindernis markiert werden. Dies resultiert aus dem begrenzten Erfassungsbereich der 3D-Kamera und den teilweise in den Wänden auftretenden Aussparungen. Die Schwierigkeiten in der Umgebungserfassung sind besonders in Streckpose relevant. In dieser Roboterpose kann die Türöffnung nicht durchquert werden.

6.6.2 Ergebnisse

Die Tabellen 6.8 und 6.9 zeigen die Ergebnisse des Testszenarios Pfadplanung durch eine Türöffnung.

Tabelle 6.8: Ergebnisse Durchquerung Türöffnung

	Kollision	Ziel erreicht	Strecke	Rotation	Zeit	max. Speicher
Kompakt	0.04	0.94	3.303 (0.023)	0.157 (0.047)	13.877 (1.827)	100596 (2466)
Greif	0.30	0.84	2.973 (0.903)	0.159 (0.047)	14.584 (3.157)	111014 (5558)
Tablett	0.82	0.48	2.548 (1.109)	0.256 (0.129)	12.344 (4.736)	115123 (3674)
Streck	0.34	0.00	0.687 (0.492)	0.552 (0.743)	15.418 (11.658)	110555 (9638)
Kompakt 2D	0.16	0.00	0.653 (0.669)	0.216 (0.257)	6.469 (7.041)	101154 (3626)
Greif 2D	0.10	0.00	0.646 (0.648)	0.221 (0.259)	6.546 (7.507)	101329 (3027)
Tablett 2D	0.25	0.00	0.791 (0.595)	0.264 (0.224)	8.580 (7.433)	99634 (2506)
Streck 2D	0.00	0.00	0.778 (0.610)	0.251 (0.220)	8.644 (7.836)	103250 (3927)

Tabelle 6.9: Ergebnisse Durchquerung Türöffnung (Fortsetzung)

	min. Abst.	mittl. Abst.	mittl. Beschl.	max. Beschl.	lok. Wied.	glo. Wied.
Kompakt	0.000 (0.000)	0.579 (0.038)	0.484 (0.106)	9.762 (7.630)	60.74 (56.13)	19.94 (18.01)
Greif	0.064 (0.203)	0.641 (0.146)	0.436 (0.140)	10.823 (10.018)	168.18 (195.45)	45.86 (51.81)
Tablett	0.066 (0.189)	0.476 (0.159)	0.540 (0.377)	14.156 (20.264)	58.3 (118.68)	16.28 (31.19)
Streck	0.144 (0.166)	0.600 (0.243)	0.527 (0.174)	5.482 (3.416)	630.98 (364.61)	171.06 (97.60)
Kompakt 2D	0.489 (0.317)	0.825 (0.244)	0.487 (0.218)	5.399 (8.438)	-	6.5 (7.38)
Greif 2D	0.524 (0.311)	0.906 (0.238)	0.319 (0.343)	5.327 (9.525)	-	8.22 (6.76)
Tablett 2D	0.257 (0.253)	0.519 (0.200)	4.001 (2.936)	88.128 (87.414)	-	11.40 (9.40)
Streck 2D	0.325 (0.278)	0.669 (0.220)	0.622 (0.088)	2.466 (1.115)	-	10.82 (8.48)

6.6.3 Analyse

In Kompakt- und Greifpose wurde die Zielpose zu 89 % erreicht. Durch die häufig ausgeführten lokalen Wiederherstellungsaktionen konnte der Roboter die Türöffnung trotz des geringen Abstandes zu den Wänden durchqueren. Im Vergleich zu den anderen Testszenarien traten größere mittlere und maximale Beschleunigungen auf. Eine mögliche Erklärung dafür liegt in der hohen Anzahl der ausgeführten lokalen und globalen Wiederherstellungsmaßnahmen, durch die der Roboter häufig stoppte und wieder anfuhr.

Schwierigkeiten entstanden durch die Wand orthogonal zur Türöffnung auf der rechten Seite des Roboters. Den einfachsten Pfad zur Durchquerung der Türöffnung bildet von der Startpose aus eine geradlinige Translation. Die Wand bewirkte als nahes Hindernis zum Roboter jedoch abstoßende Kräfte, sodass der Roboter auf die linke Seite auswich. Die daraus resultierte Rotation und der veränderte Fahrtwinkel zur Türöffnung erschwerten die Durchquerung. Beim entwickelten Pfadplanungsverfahren wurde dies durch die Drehmomente in Richtung der Zielpose reduziert. In Tablettpose war die Reduzierung unzureichend. Der Roboter wurde auf die Wand rechts der Türöffnung hinzu bewegt. Dies führte zu einer Kollision der zweiten Blase des Manipulators oder zum Abbruch der Pfadplanung. Zu ca. 54 % erreichte der Roboter die Zielpose trotz Kollision.

Probleme durch die approximierte Modellierung des Roboters zeigten sich bei eingeklapptem Tablett. Die entsprechende Blase approximiert das Tablett nicht präzise und vergrößert den minimal benötigten Freiraum. Dies bewirkte eine Verschiebung des Roboters auf die gegenüberliegende Seite. Die Verschiebung erschwerte bzw. verhinderte die Durchfahrt, da keine kollisionsfreie Pose gefunden wurde.

In Streckpose wurde die Pfadplanung trotz Nicht-Erreichbarkeit der Zielpose selten vorzeitig abgebrochen. Die durchschnittliche Zeit bis zum Abbruch lag um 46 % höher als beim vorhandenen Pfadplanungsverfahren. Der Roboter führte währenddessen keine oder geringe Bewegungen aus. Die Kollisionen in 34 % der Testdurchläufe traten auf, wenn die Wandecke auf der rechten Seite des Roboters nicht erfasst und in der 3D-Karte abgebildet wurde.

Das vorhandene Pfadplanungsverfahren konnte die Türöffnung nicht durchqueren. Im Vergleich zur Repräsentation durch eine Hülle wird bei der Repräsentation des Roboters durch die Grundfläche seiner Basis mehr Freiraum zur Pfadplanung benötigt. Aufgrund der engen Türöffnung konnte das vorhandene Pfadplanungsverfahren keinen kollisionsfreien Pfad erstellen. Die Pfadplanung wurde vor dem Erreichen der Türöffnung abgebrochen und es fanden keine Kollisionen innerhalb der Türöffnung statt. Die aufgeführten Kollisionen traten bei der Bewegung des Roboters zur Türöffnung auf, bevor die Pfadplanung abgebrochen wurde.

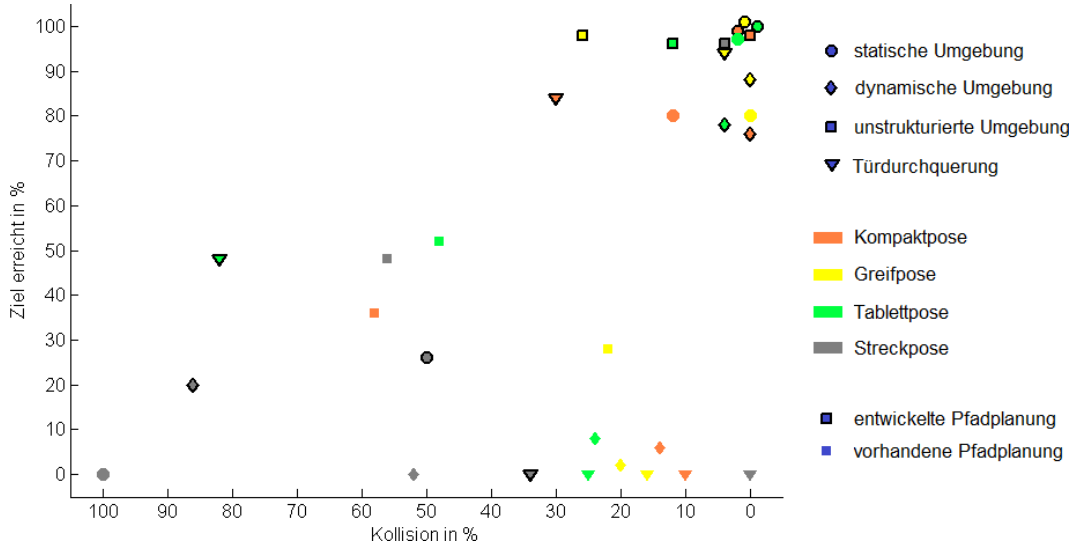


Abbildung 6.6: Zusammenfassung der Testergebnisse: Die relativen Häufigkeiten von Kollisionen und dem Erreichen des Ziels aller Tests sind in einem Streudiagramm dargestellt. Je sich eine Markierung der rechten, oberen Ecke des Streudiagramms annähert, desto besser ist das zugehörige Testergebnis. Einige Markierungen, die Testergebnisse der statischen Umgebung repräsentieren, wurden marginal verschoben, um verdeckte Markierungen zu vermeiden.

6.7 Auswertung

Die Analyse der Testszenarien zeigt im Vergleich zum vorhandenen Pfadplanungsverfahren insgesamt bessere Ergebnisse des entwickelten Pfadplanungsverfahrens. Die Zielpose wurde häufiger erreicht, die Geometrie des Roboters bei der Pfadplanung berücksichtigt und Hindernisse vollständig in drei Raumdimensionen beachtet. Das entwickelte Pfadplanungsverfahren wurde den hohen Anforderungen der Testszenarien meist gerecht. Insgesamt traten weniger Kollisionen als beim vorhandenen Pfadplanungsverfahren auf, dennoch besteht in dieser Beziehung Bedarf zur Weiterentwicklung.

Die besseren Testergebnisse des entwickelten Pfadplanungsverfahrens sind durch ihre Konzentration auf den oberen rechten Bereich des Streudiagramms in Abbildung 6.6 zu erkennen. Auffällig sind die höheren Häufigkeiten im Erreichen der Zielpose. Im Vergleich der Einzelergebnisse zeigen sich großteils Verbesserungen und einige gleichwertige Ergebnisse. Als negativ stechen die Testergebnisse mit hohen Kollisionsraten hervor, insbesondere in Streckpose.

Kollisionen entstanden meist, weil Hindernisse nicht korrekt in der Umgebungsrepräsentation erfasst wurden oder der Pfad zu nah an Hindernissen geplant wurde. Zu berücksichtigen ist, dass eine Kollision in den Testergebnissen keine Kollision des Roboters bedeuten muss, da die Ergebnisse aufgrund der approximierten Repräsentation des Roboters und Ungenauigkeiten der aufgezeichneten Daten nicht exakt sind.

Eine Möglichkeit zur Verringerung der Kollisionswahrscheinlichkeit besteht in der Optimierung der Konfiguration der Pfadplanung. Beispielsweise kann die Gewichtung der einzelnen Kräfte bei der Kombination der Kräfte eingehender untersucht werden. Die Testergebnisse lassen auf eine zu hohe Gewichtung der internen Kräfte schließen. Eine einfache, höhere Gewichtung der externen Kräfte führte in Experimenten zu abrupten Positionsänderungen der Blasen, bis hin zur Oszillation des Basisbandes. Eine andere mögliche Ursache liegt in einem zu großen Einfluss der lokalen Wiederherstellungsaktionen. Zu häufige Wiederherstellungen können dazu führen, dass ein kollisionsgefährdeter Pfad für einen zu lange beibehalten wird. Daraus können ein Pfad nah an Hindernissen mit hoher Kollisionswahrscheinlichkeit und ein Verzögern von notwendigem globalen Neuplanen resultieren.

Das Fahrverhalten des Roboters wies beim entwickelten Pfadplanungsverfahren leichte Verbesserungen auf. Der Roboter erreichte die Zielpose tendenziell in geringer Zeit und führte dabei weniger Rotationen aus. Die Analyse zeigte keine signifikanten Unterschiede bezüglich der Beschleunigungswerte des Roboters und der zurückgelegten Strecke zum Erreichen der Zielpose.

Die interne 3D-Karte hat die Umgebung ausreichend präzise repräsentiert. Fehlerhaftes Raytracing der erweiterten Costmap2D führte allerdings bei einigen Hindernissen zu einer fehlerhaften Repräsentation des Freiraumes, die die Pfadplanung teilweise signifikant beeinträchtigte. Der maximale Speicherverbrauch lag beim entwickelten Pfadplanungsverfahren im Durchschnitt um 8,8 % höher als beim vorhandenen Pfadplanungsverfahren, bzw. um 2,2 %, wenn der Ausreißer im Testszenario dynamische Umgebung in Tablettpose ignoriert wird. Der Speicherverbrauch wurde demnach durch die Nutzung einer zusätzlichen lokalen 3D-Gitterkarte nicht signifikant erhöht.

Durch die lokalen Wiederherstellungsaktionen wurden erfolgreich Korrekturen an einzelnen Blasen ausgeführt, die ein aufwendiges globales Neuplanen oder den Abbruch der Pfadplanung verhinderten. In Situationen, in denen kein Pfad zur Zielpose möglich war, führten die lokalen Wiederherstellungsaktionen für einen längeren Zeitraum zur wiederholten lokalen Optimierung, statt zum Abbruch der Pfadplanung. Wenn die Pfadplanung in einer solchen Situation aktiv bleibt, kann sie auf auftretende Hindernisse reagieren bzw. bei Veränderungen der Umgebung einen gültigen Pfad finden. Allerdings wird die Nicht-Erreichbarkeit spät oder nicht an den Nutzer bzw. eine höhere Planungsinstanz propagiert.

Zusätzlich zu den Tests in der Simulationsumgebung wurden einfache Tests mit dem physischen Roboter durchgeführt. Durch sie wurde die korrekte Ausführung der implementierten Pfadplanung in einer realen Umgebung nachgewiesen.

7 Fazit und Ausblick

In dieser Arbeit wurde ein reaktives Pfadplanungsverfahren für eine dreidimensionale Umgebung auf Basis des Elastic-Strip-Verfahrens implementiert. Das vorhandene Elastic-Band-Verfahren wurde erfolgreich zur kollisionsfreien und effizienten Pfadplanung unter Nutzung einer 3D-Karte erweitert. Schwächen des vorhandenen Pfadplanungsverfahrens konnten dabei überwunden werden. Der Mehraufwand durch das entwickelte Pfadplanungsverfahren stellt für den praktischen Einsatz keine Einschränkung dar. Die Eignung der implementierten Pfadplanung für den Einsatz mit dem Serviceroboter Care-O-bot® 3 wurde durch ausführliche Tests in einer Simulationsumgebung und einfache Tests mit dem physischen Roboter nachgewiesen.

Weiterführende Arbeiten sind im Bereich der verwendeten Umgebungsrepräsentation möglich. Zu untersuchen ist, ob in der Evaluierung festgestellte Ungenauigkeiten der Costmap2D bei der Erfassung von Freiraum hinter Hindernissen durch die Umstellung auf eine andere Implementierung einer 3D-Karte vermieden werden können. Die Präzision der erstellten 3D-Karte kann außerdem erhöht werden, indem der Erfassungsbereich des 3D-Sensors erweitert wird bzw. zusätzliche 3D-Sensoren integriert werden.

Die Pfadplanung kann durch Optimierung der Konfiguration und durch Implementierung zusätzlicher Komponenten weiter verbessert werden. Beispiele dafür sind weitere Wiederherstellungsaktionen und ein Verhalten zur temporären Unterbrechung des elastischen Streifens, um ein bewegliches Hindernis den Pfad passieren zu lassen. Weiterhin können die Blasen einer Hülle dynamisch zur Laufzeit generiert werden, statt basierend auf Daten einer Konfigurationsdatei. Dadurch kann die Teilmenge des Freiraumes um den Roboter präziser repräsentiert werden. Eine andere Möglichkeit bietet die Umsetzung des Ansatzes der vollständigen Optimierung aller Blasen des elastischen Streifens. Somit ließe sich das Pfadplanungsverfahren auf whole-body motion planning oder zusätzliche Freiheitsgrade in der Bewegungsmöglichkeit des Roboters erweitern.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AM12] Igi Ardiyanto und Jun Miura. 3D Time-Space Path Planning Algorithm in Dynamic Environment Utilizing Arrival Time Field and Heuristically Randomized Tree. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 187–192, May 2012.

[Aur91] Franz Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys (CSUR)*, 23(3): 345–405, 1991.

[Bek05] George A. Bekey. *Autonomous Robots: From Biological Inspiration to Implementation and Control*. MIT Press, Cambridge, London, 2005.

[BFE96] Johann Borenstein, Liqiang Feng und H.R. Everett. *Navigating Mobile Robots: Sensors and Techniques*. AK Peters, Ltd., Wellesley, 1996.

[BK89] Johann Borenstein und Yorem Koren. Real-Time Obstacle Avoidance for Fast Mobile Robots. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(5): 1179–1187, Sept 1989.

[BK90] Johann Borenstein und Yorem Koren. Real-Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 572–577 vol.1, May 1990.

[BK91] Johann Borenstein und Yorem Koren. The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots. *Robotics and Automation, IEEE Transactions on*, 7(3): 278–288, Jun 1991.

[BK98a] Oliver Brock und Oussama Khatib. Elastic Strips: Real-Time Path Modification for Mobile Manipulation. In *Robotics Research*, pages 5–13. Springer-Verlag, Berlin, Heidelberg, 1998.

[BK98b] Oliver Brock und Oussama Khatib. Executing Motion Plans for Robots with Many Degrees of Freedom in Dynamic Environments. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 1–6. IEEE, 1998.

[BK98c] Oliver Brock und Oussama Khatib. Mobile Manipulation: Collision-Free Path Modification and Motion Coordination. In *Proceedings of the 2nd International Conference on Computational Engineering in Systems Applications*, pages 839–845, 1998.

[BK00] Oliver Brock und Oussama Khatib. Elastic Strips: A Framework for Integrated Planning and Execution. In *Experimental Robotics VI*, pages 329–338. Springer-Verlag, Berlin, Heidelberg, 2000.

[BK02] Oliver Brock und Oussama Khatib. Elastic Strips: A Framework for Motion Generation in Human Environments. *The International Journal of Robotics Research*, 21(12): 1031–1052, 2002.

[BKV02] Oliver Brock, Oussama Khatib und Sriram Viji. Task-Consistent Obstacle Avoidance and Motion Behavior for Mobile Manipulation. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 388–393. IEEE, 2002.

[BL90] Jérôme Barraquand und Jean-Claude Latombe. A Monte-Carlo Algorithm for Path Planning with Many Degrees of Freedom. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1712–1717 vol.3, May 1990.

[BLP85] Rodney Allen Brooks und Tomas Lozano-Perez. A Subdivision Algorithm in Configuration Space for Findpath with Rotation. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-15(2): 224–233, March 1985.

[Bro86] Rodney Allen Brooks. A Robust Layered Control System for a Mobile Robot. *Robotics and Automation, IEEE Journal of*, 2(1): 14–23, 1986.

[Bro00] Oliver Brock. *Generating Robot Motion: The Integration of Planning and Execution*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, USA, August 2000.

[BRSM+09] Radu Bogdan Rusu, Aravind Sundaresan, Benoit Morisset, Kris Hauser, Motilal Agrawal, Jean-Claude Latombe und Michael Beetz. Leaving Flatland: Efficient Real-Time Three-Dimensional Perception and Motion Planning. *Journal of Field Robotics*, 26(10): 841–862, 2009.

[BS11] Dominik Belter und Piotr Skrzypczynski. Integrated Motion Planning for a Hexapod Robot Walking On Rough Terrain. In *Prepr. 18th IFAC World Congress, Milan, Italy*, pages 6918–6923, 2011.

[CR87] John Canny und John Reif. New Lower Bound Techniques for Robot Motion Planning Problems. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 49–60. IEEE, 1987.

[DSL03] Amador R. Diéguez, Rafael Sanz und Joaquin López. Deliberative On-Line Local Path Planning for Autonomous Mobile Robots. *Journal of Intelligent and Robotic Systems*, 37(1): 1–19, 2003.

[DVX13] Ivan Dryanovski, Roberto G. Valenti und Jizhong Xiao. Fast Visual Odometry and Mapping From Rgb-D Data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2305–2310. IEEE, 2013.

[EHS+14] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers und Wolfram Burgard. 3D Mapping with an RGB-D Camera. *IEEE Transactions on Robotics*, 30(1): 177–187, Feb 2014.

[Elf89] Alberto Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6): 46–57, June 1989.

[EP00] Pedro Encarnacao und Antonio Pascoal. 3D Path Following for Autonomous Underwater Vehicle. In *Proc. 39 th IEEE Conference on Decision and Control*, 2000.

[FACS03] Edward H.L. Fong, Williams Adams, Frederick L. Crabbe und Alan C. Schultz. Representing a 3-D Environment with a 2 1/2 -D Map Structure. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 3, pages 2986–2991 vol.3, Oct 2003.

[FBB09] Christian Frese, Thomas Batz und Jürgen Beyerer. Kooperative Bewegungsplanung zur Unfallvermeidung im Straßenverkehr mit der Methode der elastischen Bänder. In *Autonome Mobile Systeme 2009*, pages 193–200. Springer-Verlag, Berlin, Heidelberg, 2009.

[FBT96] Dieter Fox, Wolfram Burgard und Sebastian Thrun. Controlling Synchro-Drive Robots with the Dynamic Window Approach to Collision Avoidance. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 3, pages 1280–1287. IEEE, 1996.

[FBT97] Dieter Fox, Wolfram Burgard und Sebastian Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics & Automation Magazine*, 4(1): 23–33, 1997.

[Gat92] Erann Gat. Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. In *AAAi*, volume 1992, pages 809–815, 1992.

[GHS04] Birgit Graf, Matthias Hans und Rolf D. Schraft. Care-O-Bot II - Development of a Next Generation Robotic Home Assistant. *Autonomous Robots*, 16(2): 193–205, 2004.

[GMBJ11] Santiago Garrido, Luis Moreno, Dolores Blanco und Piotr Jurewicz. Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching. volume 2, pages 42–64, 2011.

[GRH+09] Birgit Graf, Ulrich Reiser, Martin Hägele, Kathrin Mauz und Peter Klein. Robotic Home Assistant Care-O-Bot 3-Product Vision and Innovation Platform. In *Advanced Robotics and its Social Impacts (ARSO), 2009 IEEE Workshop on*, pages 139–144. IEEE, 2009.

[GS00] Stefan K. Gehrig und Fridtjof J. Stein. Collision Avoidance Using Elastic Bands for an Autonomous Car. In *Intelligent Autonomous Systems 6*, pages 1065–1072. IOS Press, Amsterdam, 2000.

[Gut94] Ralf Gutsche. *Fahrerlose Transportsysteme*. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig, Wiesbaden, 1994.

[GWS01] Birgit Graf, José Manuel Hostalet Wandosell und Christoph Schaeffer. Flexible Path Planning for Nonholonomic Mobile Robots. In *Proc. 4th European workshop on advanced Mobile Robots (EUROBOT01)., Fraunhofer Inst. Manufact. Eng. Automat.(IPS). Lund, Sweden*, pages 199–206, 2001.

[HLN12] Joachim Hertzberg, Kai Lingemann und Andreas Nüchter. *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. Springer-Verlag, Berlin, Heidelberg, 2012.

[Hop92] Peter Hoppen. *Autonome Mobile Roboter: Echtzeitnavigation in bekannter und unbekannter Umgebung*. BI-Wiss.Verlag, Mannheim, 1992.

[HPJ+12] Armin Hornung, Mike Phillips, Edward Gil Jones, Maren Bennewitz, Maxim Likhachev und Sachin Chitta. Navigation in Three-Dimensional Cluttered Environments for Mobile Manipulation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 423–429, May 2012.

[HWB+13] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss und Wolfram

Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based On Octrees. *Autonomous Robots*, 34(3): 189–206, 2013.

[JT80] Chris L. Jackins und Steven L. Tanimoto. Oct-Trees and Their Use in Representing Three-Dimensional Objects. *Computer Graphics and Image Processing*, 14(3): 249–270, 1980.

[Kha86] Oussama Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The international journal of robotics research*, 5(1): 90–98, 1986.

[KJCL97] Maher Khatib, Hazem Jaouni, Raja Chatila und Jean-Paul Laumond. Dynamic Path Modification for Car-Like Nonholonomic Mobile Robots. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 2920–2925 vol.4, Apr 1997.

[KJIF06] Voemir Kunchev, Lakhmi Jain, Vladimir Ivancevic und Anthony Finn. Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4252 of *Lecture Notes in Computer Science*, pages 537–544. Springer-Verlag, Berlin, Heidelberg, 2006.

[Kod87] Daniel E. Koditschek. Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1–6, Mar 1987.

[Kru98] Eckhard Kruse. *Bewegungsplanung für mobile Roboter in dynamischen Umgebungen auf Basis automatisch erzeugter statistischer Daten*. Shaker Verlag, Aachen, 1998.

[KS08] David Kortenkamp und Reid Simmons. Robotic Systems Architectures and Programming. In *Springer Handbook of Robotics*, pages 187–206. Springer-Verlag, Berlin, Heidelberg, 2008.

[KYB+99] Oussama Khatib, Kazu Yokoi, Oliver Brock, Kyong-Sok Chang und Arancha Casal. Robots in Human Environments: Basic Autonomous Capabilities. *The International Journal of Robotics Research*, 18(7): 684–696, 1999.

[KYK12] Jinsung Kwon, Taizo Yoshikawa und Oussama Khatib. Elastic Strips: Implementation On a Physical Humanoid Robot. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3369–3376, Oct 2012.

[Lat91] Jean-Claude Latombe. Robot Motion Planning. *Kluwer Academic Publishers, Boston*, 1991.

[LaV06] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, New York, 2006.

[LM13] Mathieu Labbé und François Michaud. Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation. *Robotics, IEEE Transactions on*, 29(3): 734–745, June 2013.

[LP83] Tomas Lozano-Perez. Spatial Planning: A Configuration Space Approach. *Computers, IEEE Transactions on*, 100(2): 108–120, 1983.

[LPW79] Tomás Lozano-Pérez und Michael A. Wesley. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *Communications of the ACM*, 22(10): 560–570, 1979.

[LT11] Chi-Tai Lee und Ching-Chih Tsai. 3D Collision-Free Trajectory Generation Using Elastic Band Technique for an Autonomous Helicopter. In *Next Wave in Robotics*, volume 212 of *Communications in Computer and Information Science*, pages 34–41. Springer-Verlag, Berlin, Heidelberg, 2011.

[ME85] Hans Peter Moravec und Alberto Elfes. High Resolution Maps From Wide Angle Sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121. IEEE, 1985.

[Mea82] Donald Meagher. Geometric Modeling Using Octree Encoding. *Computer graphics and image processing*, 19(2): 129–147, 1982.

[MM00] Javier Minguez und Luis Montano. Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 2094–2100 vol.3, 2000.

[MM04] Javier Minguez und Luis Montano. Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. *Robotics and Automation, IEEE Transactions on*, 20(1): 45–59, Feb 2004.

[MM08] Maja J. Mataric und François Michaud. Behavior-Based Systems. In *Springer Handbook of Robotics*, pages 891–909. Springer-Verlag, Berlin, Heidelberg, 2008.

[Neh02] Ulrich Nehmzow. *Mobile Robotik*. Springer-Verlag, Berlin, Heidelberg, 2002.

[Nie13] Adam Niewola. Rough Surface Description System in 2,5D Map for Mobile Robot Navigation. *Journal of Automation Mobile Robotics and Intelligent Systems*, 7, 2013.

[Nou97] Illah Reza Nourbakhsh. *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers, Boston, 1997.

[PS03] Roland Philippsen und Roland Siegwart. Smooth and Efficient Obstacle Avoidance for a Tour Guide Robot. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 446–451, Sept 2003.

[PTB07] Patrick Pfaff, Rudolph Triebel und Wolfram Burgard. An Efficient Extension to Elevation Maps for Outdoor Terrain Mapping and Loop Closing. *The International Journal of Robotics Research*, 26(2): 217–230, 2007.

[QCG+09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler und Andrew Y. Ng. ROS: An Open-Source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[QK93] Sean Quinlan und Oussama Khatib. Elastic Bands: Connecting Path Planning and Control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807 vol.2, May 1993.

[Qui94] Sean Quinlan. *Real-Time Modification of Collision-Free Paths*. PhD thesis, Stanford University, 1994.

[RK03] Ananth Ranganathan und Sven Koenig. A Reactive Robot Architecture with Planning On Demand. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1462–1468. IEEE, 2003.

[SB05] Thomas Sattel und Thorsten Brandt. Ground Vehicle Guidance Along Collision-Free Trajectories Using Elastic Bands. In *American Control Conference, 2005. Proceedings of the 2005*, pages 4991–4996 vol. 7, June 2005.

[SH75] Michael Ian Shamos und Dan Hoey. Closest-Point Problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 151–162. IEEE, 1975.

[SK08] Bruno Siciliano und Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2008.

[Spa92] Horst Spandl. *Lernverfahren zur Unterstützung der Routenplanung für einen mobilen Roboter*. VDI-Verlag, Düsseldorf, 1992.

[SS83] Jacob Theodore Schwartz und Micha Sharir. On the "Piano Movers"Problem. II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Advances in applied Mathematics*, 4(3): 298–351, 1983.

[Thr02] Sebastian Thrun. Robotic Mapping: A Survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[TMD+06] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann et al. Stanley: The Robot That Won the DARPA Grand Challenge. *Journal of field Robotics*, 23(9): 661–692, 2006.

[TPB06] Rudolph Triebel, Patrick Pfaff und Wolfram Burgard. Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2276–2282. IEEE, 2006.

[WG02] José Manuel Hostalet Wandosell und Birgit Graf. Non-Holonomic Navigation System of a Walking-Aid Robot. In *Proc. 11th IEEE Int. Workshop on Robot and Human Interactive Communication (ROMAN 2002)*, pages 518–523, 2002.

[WSD03] Marcus Walther, Peter Steinhaus und Rüdiger Dillmann. A Robot Navigation Approach Based On 3D Data Fusion and Real Time Path Planning. In *Multisensor Fusion and Integration for Intelligent Systems, MFI2003. Proceedings of IEEE International Conference on*, pages 45–50, July 2003.

[YB06] Yuandong Yang und Oliver Brock. Elastic Roadmaps: Globally Task-Consistent Motion for Autonomous Mobile Manipulation in Dynamic Environments. 2006.

[YB10] Yuandong Yang und Oliver Brock. Elastic Roadmaps-Motion Generation for Autonomous Mobile Manipulation. *Autonomous Robots*, 28(1): 113–130, 2010.