B. Sc. Hauke Petersen

Pfadplanung und Ausführung für einen mobilen Roboter im Kontext des RoboCup Wettbewerbs

INF FAKULTÄT FÜR INFORMATIK

Institut für Verteilte Systeme

Forschungsarbeit

Pfadplanung und Ausführung für einen mobilen Roboter im Kontext des RoboCup Wettbewerbs

Autor: B. Sc. Hauke Petersen

Professor: Junior-Prof. Dr.-Ing. Sebastian Zug

Professor: Prof. Dr.-Ing. habil. Arndt Lüder

Sommersemester 2015

Petersen, Hauke: *Pfadplanung und Ausführung für einen mobilen Roboter im Kontext des RoboCup Wettbewerbs*
Forschungsarbeit, Otto-von-Guericke-Universität
Magdeburg, 2015.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BMT	Basic Manipulation Test
BNT	Basic Navigation Test
BTT	Basic Transportation Test
CBT	Conveyor Belt Test
PPT	Precision Placement Test
ROS	Robotic Operating System
STAIR	Standford-AI-Robot-Projektes
v-rep	Virtual Robot Experimentation Platform
AMR	autonomen mobilen Robotern

1 Einleitung

Der Wettkampf, ob körperlich oder geistig, ist in allen Bereichen des Lebens ein wichtiger Motor für Innovationen. Die Robotik befindet sich derzeit auf einem technischen Stand, der es ermöglicht, autonome mobile Roboter vielseitig einzusetzen. Um den Wettbewerbsgedanken auch hier zu entfachen, wurden verschiedene Turniere ins Leben gerufen, in denen Entwicklerteams ihr Können an der Soft- und Hardware präsentieren. Einer dieser Wettkämpfe ist der RoboCup.

Der RoboCup wurde erstmals 1997 als Roboterfußball-Weltmeisterschaft ausgetragen und findet seither jährlich statt. Zudem wurde die Auswahl an Wettbewerben bis heute kontinuierlich erweitert. So gibt es abseits des Fußballs unter anderem die RoboCup Logistic League.

An der Otto-von-Guericke-Universität Magdeburg gründete sich 2010 das robOTTO-Team, um seine Innovationen in diesem Wettbewerb zu präsentieren. In diesem Turnier soll mithilfe dreier autonomer Roboter in einer fiktiven Produktionsstätte möglichst viele Produktionserzeugnisse generiert werden. (vgl. [Han12, S.4])

Fünf Jahre später entschied das Team, sich einer neuen Herausforderung zu stellen und wechselte in die RoboCup@Work Liga. In dieser müssen, mit einem mobilen Roboter mit Greifarm Industrieaufgaben gelöst werden. Eine ausführliche Beschreibung der RoboCup@Work Liga wird in Kapitel 2 vorgenommen. Vorrangig wird auf die Navigation in Arenen mit jährlich änderndem Aufbau eingegangen. Die Navigation setzt sich aus verschiedenen Teilaufgaben zusammen: der Pfadplanung vor der Bewegung des Roboters, der Planung während der Fahrt, dem Zusammenspiel beider sowie der Lokalisation des Roboters in der Arena. Die Arbeit fokusiert die Pfadplanung, die in Kapitel 3 beschrieben wird.

Die Untersuchung beschäftigt sich mit der Frage, welche der vorhandenen Planungsalgorithmen für den Einsatz während eines Wettbewerbes des Robo-

Cup@Work am geeignetsten ist. Dazu wird ein Versuchsaufbau erstellt und Kriterien festgelegt, anhand derer sich die Planungsalgorithmen testen lassen.

2 Allgemeine Grundlagen

2.1 RoboCup@Work

Der RoboCup@Work fand erstmals 2012 statt und benutzt Konzepte aus den anderen RoboCup-Wettbewerben, um neue unerforschte Szenarien für Industrie- und Serviceroboter zu erstellen. Beispiele für solche Szenarien sind:

- Laden und Entladen von Containern, die Objekte der gleichen oder unterschiedlicher Größe enthalten,
- Objekte aus geordneten Ablagen oder ungeordneten Ansammlungen suchen und transportieren,
- Maschinen bedienen, Knöpfe drücken, Türen und Schränke öffnen bzw. schließen und Ähnliches bei vorher unbekannten Mechanismen,
- Flexible Planung von Produktionsprozessen unter Berücksichtigung mehrerer Teilnehmer (Menschen, Roboter und Maschinen),
- kooperativer Zusammenbau von komplexen Objekten mit anderen Robotern und/ oder Menschen,
- kooperatives Sammeln von, auf einer Fläche verstreuter Objekten,
- kooperatives Transportieren von Objekten mit anderen Robotern und/ oder Menschen.

Die Aufgaben des RoboCup@Work beschränken sich dabei auf bisher ungelöste Probleme in Bereichen wie zum Beispiel: Robotik, künstliche Intelligenz, Pfadplanung, mobile Manipulation, etc. Zudem wird darauf geachtet, dass die Problemstellungen einerseits möglichst allgemein und unabhängig von speziellen Betriebsumgebungen gehalten werden, andererseits trotzdem realitätsnah zu wirklichen Problemen in der Industrie und mit einem angemessenen Aufwand umsetzbar sind. (vgl.[NKH+14, S.5])

2.2 Aufgaben des Wettbewerbs

Im Folgenden werden die Anforderungen an die Teams und deren Roboterplattformen erläutert. Das Regelwerk [NKH+14, S.13 ff] beschreibt Bedingungen und Bestimmungen, die sich auf alle oder spezifische Aufgaben während eines Wettkampfes auswirken.

Damit ein Roboter für den Wettbewerb zugelassen wird, muss dieser festgelegte Kriterien erfüllen. Es sind nur mobilen Roboter, die sich auf Rädern bewegen zugelassen. Die Art der Fortbewegung, wie zum Beispiel omnidirektionale Bewegung, ist nicht weiter eingeschränkt. Zudem muss der Roboter über einen Manipulator verfügen, welcher in der Lage ist, Objekte zu greifen, die von einem Parallelgreifer mit einer Spannweite von fünf Zentimeter gegriffen werden können. Diese Objekte sind nicht schwerer als 300 g. Der gesamte Roboter muss mit eingefahrenen Komponenten, wie dem Manipulator, in einen Quader mit den Abmaßen 80 cm mal 50 cm mal 80 cm passen. Mit ausgefahrenen Komponenten darf die Größe eines Quaders mit den Abmaßen 120 cm mal 80 cm mal 160 cm nicht überschritten werden.

Alle Sensoren, die ein Team nutzt, müssen am Roboter verbaut sein. Es ist möglich, externe Sensoren zu nutzen, dies führt zu Punktabzug aufgrund von vereinfachten Bedingungen. Außerdem dürfen bei Manipulationsaufgaben Objekte nicht außerhalb der sogenannten „service area“ platziert werden. Aus Sicht der Navigation ist daher eine präzise Positionierung wichtig, um eine optimale Nutzung des Arms zu gewährleisten. Ein weiteres Kriterium ist die Kollisionsfreiheit. Kontrolliertes Berühren der Spielfeldaufbauten ist durchaus erlaubt, kommt es zu stärkeren Auftreffen mit der Umwelt, führt das zu Punktabzug. Eine zügige Fortbewegung ist auch wichtig, da je nach Aufgabe unterschiedliche Zeitlimits eingehalten werden müssen.

Das Spielfeld (siehe Abbildung 2.1) ist zwischen minimal zwei mal vier und maximal acht mal zehn Meter groß. Es ist von Mauern bzw. Banden begrenzt, die, möglicherweise verschließbare, Tore enthalten können. Die Größe der Tore beträgt zwischen 20 und 40 cm. Des Weiteren befinden sich auf dem Boden des Spielfelds an speziellen Stellen wie Lade- oder Entladezonen durch AR-Code[1] gekennzeichnete Landmarken.

[1] AR: *Augmented Reality* dt.: erweiterte Realität. Die AR-Codes sind schwarz-weiße Punkt-Anordnungen ähnlich der QR-Codes. Ihre Gestaltung dient vor Allem dem Abschätzen des Standpunktes eines Betrachters. Weitere Informationen befinden sich hier : [ar-]

Abb. 2.1: Spielfeld des RoboCup@Work 2015 in Hefei, China. [selbst erstellt]

Die Wettkämpfe umfassen zwei Runden. In der ersten Runde finden der „Basic Navigation Test (BNT)“, der „Basic Manipulation Test (BMT)“ und der „Basic Transportation Test (BTT)“ statt. Darauf folgen in der zweiten Runde der „ Conveyor Belt Test (CBT)“ und der „Precision Placement Test (PPT)“.

Beim BNT wird eine reaktive Pfadplanung benötigt, da der Roboter eine vorher festgelegt Reihenfolge von Punkten anfahren und auf ihnen mit einer bestimmen Orientierung stoppen soll. Dabei sind, je nach Schwierigkeitsgrad, mehrere, vorher unbekannte Hindernisse auf dem Spielfeld verteilt.

Im darauffolgenden BMT geht es in erster Linie um die Fähigkeiten des Roboters, seinen Arm zu bedienen. Hier sind kleinere Bewegungen vorgesehen, um die Anfangsbedingung zu erschweren bzw. durch die entstehenden kleineren Abweichungen zufällig zu gestalten. Aus diesem Grund ist, wie bereits erwähnt, eine genaue Positionierung der Roboterplattform wichtig.

Der BTT stellt eine Kombination aus den beiden vorherigen Tests dar. Der Roboter wird außerhalb des Spielfeldes in der Nähe eines Tores platziert und muss selbstständig zu einer Ladezone fahren, dort verschiedene Objekte aufladen, sie zu einer Entladezone bringen und abgeben. Die Anforderungen für diesen Test kombinieren sich aus denen der vorangegangenen Tests.

Nach den drei „Basic“-Tests wird das Anforderungsniveau in Runde zwei erhöht. Beim CBT muss sich der Roboter neben einem Fließband einfinden, welches angeschaltet wird. Je nach Schwierigkeitsgrad wird eine Anzahl von Objekten auf das Fließband gelegt, die der Roboter herunternehmen muss, bevor sie das Ende des Fließbands erreichen. Aus Sicht der Navigation werden hier

eine gute Positionierung zum Fließband und gegebenenfalls das Anpassen der Geschwindigkeit an die des Fließbands gefordert.

Beim PPT werden Objekte in vorgesehene Aussparungen gelegt. Für die Navigation gelten die gleichen Anforderungen wie beim BMT.

2.3 KUKA youBot

das robOTTO Team benutzt für die Aufgaben des RoboCup@Work den youBot von KUKA (siehe Abbildung 2.2). Dieser wurde hauptsächlich für die Lehre und Forschung entwickelt und erfüllt alle im Abschnitt 2.2 genannten Kriterien. Er bietet sowohl hardware- als auch softwaretechnisch viele Möglichkeiten zur Modifikation und ist daher vielseitig einsetzbar.

Abb. 2.2: Der youBot von KUKA im Auslieferungszustand. [kuk]

Der youBot besteht aus zwei Komponenten: der beweglichen Plattform und dem Arm, die unabhängig voneinander bedient werden können. Im Folgenden werden diese Komponenten näher beschrieben. Anschließend werden die Änderungen erläutert, die das robOTTO-Team an der Hardware vorgenommen hat.

2.3.1 Komponenten des KUKA youBot

Die erste Komponente des youBot, eine Plattform mit vier Mecanum-Rädern, ist für die Mobilität zuständig. Diese kann durch die einzeln angetriebenen Räder omnidirektional bewegt werden. Sie verfügt über eine austauschbare Grundplatte, unter der sich sowohl die elektronischen Bauteile, wie zum Beispiel Motoren, Batterie und Stromversorgung, als auch der onboard-PC, ein Intel AtomTM Dual Core D510, befinden. Die Motorsteuerung der Räder kann über Ethernet oder EtherCAT erreicht werden. Dazu wird ein handelsübliches Ethernetkabel benötigt, welches mit dem onboard-PC oder externen Rechnern verbunden wird. Des Weiteren befinden sich an der Oberseite der Grundplatte ein Display, welches den Status der Motoren, des onboard-PCs und der Batterie anzeigt. Außerdem befinden sich auf dieser Seite zwei 24V-Ausgänge, ein 24V-Eingang zum Laden der Batterie und gleichzeitiges Betreiben des youBots am Stromnetz, zwei USB- und ein Ethernet-Anschluss, welche mit dem onbord-PC verbunden sind, zwei EtherCAT-Anschlüsse und ein Knopf zum Ein- und Ausschalten der Motoren und des onboard-PCs.

Die zweite Komponente des youBots ist der Arm. Dieser ist eine serielle Kette aus fünf Gliedern zwischen rotierenden Achsen. Damit hat er fünf Freiheitsgrade in denen der Endeffektor im Arbeitsraum bewegt werden kann. Der Endeffektor besteht aus einem Greifer mit zwei Fingern, die jeweils von einem Linearmotor betrieben werden und somit unabhängig voneinander angesteuert werden können. Genau wie bei der Plattform können die Motoren der Achsen und des Greifers über Ethernet bzw. EtherCAT angesteuert werden. (vgl. S.6 ff. [you12])

2.3.2 Anpassungen durch robOTTO

Zunächst wurde der youBot durch zwei Laserscanner an Vorder- und Rückseite erweitert. Diese dienen sowohl dem Erfassen der Umgebung, als auch der Lokalisation in dieser. Damit ist es zu jedem Zeitpunkt möglich, der Navigation die derzeitige Position des Roboters zur Verfügung zu stellen und einen Startpunkt für Pfadplanungen in der Karte festzulegen. Zudem wird durch die Lokalisation überprüft, ob der Roboter den berechneten Pfad fährt und Kollisionen vermeidet.

Darüber hinaus wurde der onboard-PC durch einen Intel NUC NUC5i7RYH ersetzt, welcher mit einem Intel Core i7-5557U Prozessor und 16 GB Arbeitsspeicher ausgestattet ist, um dem Team mehr Rechenkraft zur Verfügung zu stellen. Dies war nötig, da Reaktive Pfadplanung und die dazugehörige Echtzeit-Lokalisation viel Rechenleistung benötigen, um die Navigation auszuführen und gleichzeitig weitere Aufgaben bewältigen zu können.

Des Weiteren wurde der Endeffektor des youBots mit einer Kamera ausgerüstet, die zur Erkennung von Objekten dient.

Um einen größeren Öffnungswinkel zu erzielen und somit eine größere Anzahl von verschiedenen Objekten greifen zu können, wurde der lineare Greifer gegen einen Scheren-Greifer mit zwei flexiblen Fingern ausgetauscht.

2.4 Vergleich thematisch verwandter Arbeiten

In diesem Abschnitt werden drei thematisch verwandte Arbeiten vorgestellt und mit der vorliegenden Arbeit verglichen.

In dem Paper "A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games" [ASK15] von Z. A. Algfoor, M. S. Sunar und H. Kolivand werden populäre globale Pfadplanungsalgorithmen erläutert. Die Autoren legen dabei den Fokus sowohl auf aktuelle Entwicklungen als auch auf Verbesserungen der bestehenden Algorithmen und deren Einfluss auf die Robotik und Videospielindustrie. Ziel ist es, dem Leser einen fundierten Überblick über die Forschung der letzte zehn Jahre zu verschaffen. Dabei werden die prinzipiellen Ansätze zusammengefasst und ihre Lösungen aufgezeigt. Zudem werden Einschätzungen und Erwartungen zukünftiger Forschungen genannt. Im Abschnitt 3.2 der vorliegenden Arbeit werden einige der Ansätze aufgegriffen. Das Paper eignet sich aufgrund des gegebenen Überblicks als zusätzliche Literatur. Die Pfadplanungsalgorithmen werden allerdings nur aufgezählt und nicht miteinander verglichen. Die Autoren fokussieren sich auf die Theorie und erwähnen den praktischen Einsatz der Algorithmen nicht.

Das Buch "Autonome mobile Roboter" [Hop92] von P. Hoppen behandelt die Echtzeitnavigation von autonomen mobilen Robotern (AMR) in bekannten und unbekannten Umgebungen. Zunächst führt das Buch in die Thematik und die Probleme der Navigation von AMR ein und nennt eine Reihe von bekannten

Navigationsverfahren. Dabei wird festgestellt, dass der Begriff der Navigation meist in die Phasen *Wegplanung* und *Fahrt* unterteilt wird. Die meisten der aufgezeigten Arbeiten bzw. Ansätze behandeln dabei nur eine dieser Phasen. Im Anschluss stellt der Autor eine Reihe wesentlicher Forderungen an ein Navigationsverfahren für AMR auf. Teile dieser Kriterien werden in Abschnitt 3.3 der vorliegenden Arbeit verwendet um Planungsalgorithmen miteinander zu vergleichen. Hoppen stellt die Entwicklung eines Navigationsverfahrens, das diese Anforderungen erfüllt, vor. Dabei betrachtet er die beiden Phasen der Navigation als einen verzahnten Prozess, was die Grundlage für den entwickelten Ansatz bildet. Dieser Navigationsansatz soll einen zielgerichtete Navigation sowohl in bekannten als auch in unbekannten Umgebungen ermöglichen. Der Navigationsansatz wird abschließend in einer Simulationsumgebung anhand eines stark vereinfachten Robotermodells getestet. Die Grundlagen aus Hoppens Buch werden in dieser Arbeit um die Fokussierung auf einen konkreten Roboter, den KUKA youBot (siehe Abschnitt 2.3) und das ROS Software-Framework (siehe Abschnitt 2.5) erweitert. Die Aufteilung der Navigation in Unterbereiche wird übernommen und die Wegplanung in Abschnitt 3.1.2 weiter in lokale und globale Planung aufgegliedert.

In der Masterarbeit "Implementierung eines reaktiven Pfadplanungsalgorithmus für nicht holonome mobile Plattformen in industriellen Umgebungen" [Bub07] von A. Bubeck steht die Navigation in dynamischen Umgebungen im Mittelpunkt. Die Pfadplanung wird hier auch in die Teilprobleme *globale* und *lokale* Pfadplanung aufgeteilt. Bekannte Pfadplanungsverfahren werden aufgezählt und erläutert. Anschließend folgt ein Festlegen von Kriterien, anhand derer die Pfadplanungsansätze verglichen werden. Nach der Auswahl eines geeigneten Verfahrens erfolgt eine Implementierung auf einer Roboterplattform MX-500 der Firma Neobotix. Durch eine Reihe von Tests werden die sogenannte Turning-Parameter erarbeitet, die dass Verfahren verbessern sollen. Abschließend erfolgt eine Auswertung der Praxis-Tests, in der genannte Parameter für den Roboter festgehalten werden. Bubeck verfolgt einen, der vorliegenden Arbeit ähnlichen Ansatz. Er fokussiert sich nach der Aufteilung der Pfadplanung auf die lokale Planung, während in dieser Arbeit die globale Planung betrachtet wird.

Die vorliegende Arbeit betrachtet einen anwendungsbezogenen Spezialfall. Da sowohl durch die Regeln des RoboCup@Work Wettbewerb als auch durch Einigungen des Team robOTTO sowohl die Roboterplattform, die Software als auch die Versuchsumgebung definiert sind, grenzt sich diese Arbeit von anderen ab.

2.5 Robotic Operating System (ROS)

ROS ist ein flexibles Softwaregerüst für Roboter, welches 2007 im Rahmen des Standford-AI-Robot-Projektes (STAIR) am Stanford Artificial Intelligence Laboratory entwickelt wurde. Bis heute wird es, hauptsächlich durch das Robotikinstitut Willow Garage, weiterentwickelt. Die aktuell vom Team benutzte Version trägt den Namen *Indigo Igloo* und ist damit die achte Distributionsversion von ROS, welche am 22. Juli 2014 veröffentlicht wurde.

ROS ist eine Sammlung von Werkzeugen, Bibliotheken und Richtlinien, deren Ziel es ist, die Entwicklung von komplexer und robuster Robotersoftware zu vereinfachen und dabei eine große Palette verschiedener Roboter abzudecken. Es wurde darauf geachtet, dass die Software die Möglichkeit bietet, dass mehrere Personen bzw. Teams mit ähnlichen Projekten Fortschritte untereinander austauschen. Aus diesem Grund besitzt ROS eine modulare Struktur. Aufgerufenen Programmen ist es beispielsweise möglich über ein *publisher- /subscriber*-Prinzip (dt.: Sender/Hörer) zu kommunizieren, dieses wird in ROS über ein *node graph* (dt.: Knotendiagramm) dargestellt (siehe Abbildung 2.3).

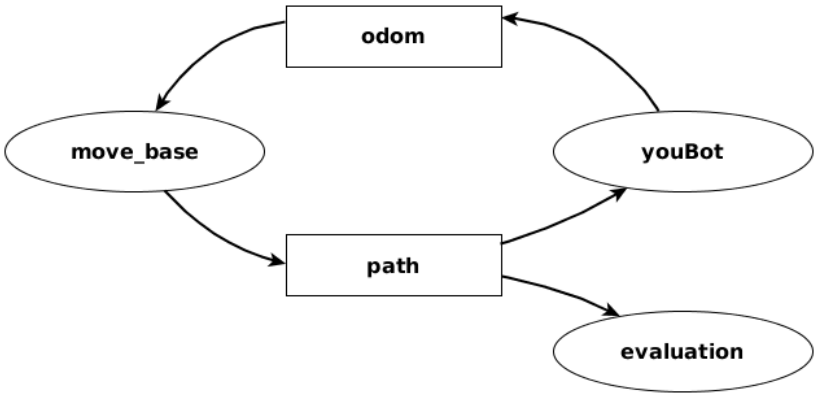

Abb. 2.3: Beispielhaftes Schema des ROS-*node-graph*. Die Ellipsen stellen die *nodes* da, während die kleinen Rechtecke die *topics* darstellen, auf die die *nodes* je nach Pfeilrichtung *publishen* oder *subscriben*. [selbst erstellt]

Programme werden als *node* (dt.: Knoten) dargestellt und hören nur auf die *topics* (dt.: Nachrichten), die sie auslesen sollen. Gleichzeitig geben sie selber *topics* unter vorgeschriebenen Namen aus. Das bedeutet, dass *nodes* jederzeit ausgetauscht werden können, solange sie sowohl zum *publishen* als auch zum

subscriben die gleichen Namen verwenden wie der ausgetauschte *node*. Innerhalb eines Teams wie robOTTO ist es durch diese Konventionen möglich, an den einzelnen Komponenten eines Roboters parallel zu arbeiten, es muss zuvor nur eine einheitliche Benennung der verwendeten *topics* stattfinden. (vgl. [rosb])

3 Definition der Anforderungen

3.1 Pfadplanung

Das Finden eines kollisionsfreien Pfades von einer Start- zu einer Zielpose wird bei mobilen Robotern als Pfadplanung bezeichnet. [SK08] Voraussetzungen dafür sind eine Beschreibung der Gestalt des Roboters und eine partielle bis vollständige Beschreibung der Umgebung sowie Posen, welche die Position und Orientierung des Roboters und der Objekte in seiner Umgebung beschreiben. Als Pfad wird eine Sequenz von Posen verstanden, die Start- und Zielpose verbindet. Ein Pfad ist unter der Voraussetzung, dass der Roboter das einzige bewegte Objekt ist, zeitunabhängig. Bei der Pfadplanung finden Algorithmen Verwendung, die, unter Berücksichtigung der Umgebung und der gewünschten Start- und Zielpose, einen Pfad zwischen diesen Posen generieren. „Ein Pfadplanungsverfahren ist vollständig, wenn es genau dann einen Pfad für eine Eingabe generiert, wenn ein gültiger Pfad existiert.“ [Sch14, S. 13]

3.1.1 Arbeits- und Konfigurationsraum

Die Pfadplanung für einen Roboter kann in zwei Räumen erfolgen, dem Arbeitsraum und dem Konfigurationsraum.

Der Arbeitsraum umfasst alle Punkte im physischen Raum, die für den Roboter erreichbar sind. Er wird geometrisch in einem euklidischen Raum mittels einem kartesischen Koordinatensystem dargestellt und ist, je nach Robotertyp, zwei- oder dreidimensional. Die Position und Die Orientierung des Roboters wird durch eine Pose in einem absoluten, arbeitsraumfesten oder relativen, roboterfesten Koordinatensystem beschrieben. Der Roboter selbst wird dabei durch ein vereinfachtes geometrisches Modell dargestellt.

Der Konfigurationsraum beschreibt die Zustände aller Gelenke und Aktoren des Roboters. Er wird durch einen n-dimensionalen, euklidischen Raum darge-

stellt, in dem der Roboter als ein Punkt mit entsprechenden n Freiheitsgraden abgebildet wird. Die Gestalt des Roboters wird dazu auf die Darstellung der Hindernisse übertragen. Diese Hindernisse werden durch alle Konfigurationen beschrieben, die eine Kollision mit dem Roboter verursachen. (vgl. [LP83])

Sowohl im Arbeits- als auch im Konfigurationsraum wird zwischen belegten und freien Zuständen unterschieden. Die Menge aller unbelegten Zustände wird durch den Freiraum abgebildet, während die belegten Zustände Hindernisse darstellen. Im Arbeitsraum sind die belegten Zustände alle Raumkoordinaten, in denen sich Hindernisse befinden. Im Konfigurationsraum sind sie die Menge der Konfigurationen, bei denen der Roboter kollidieren würde.

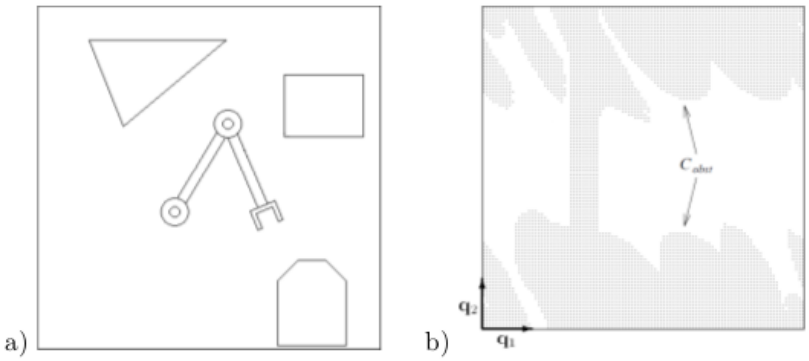

Abb. 3.1: Arbeits- und Konfigurationsraum: a) zeigt den zweidimensionalen Arbeitsraum eines Manipulators mit zwei Gelenken. Die umrandeten Bereiche bilden Hindernisse. In b) ist der zugehörige Konfigurationsraum dargestellt. Er ist ebenfalls zweidimensional, da der Manipulator Freiheitsgrade besitzt (Rotationen q1 und q2 um die beiden Gelenke). Der grau schraffierte Bereich beschreibt die transformierten Hindernisse. [Sch14, S.14]

Der Roboter wird im Konfigurationsraum als Punkt und seine Pfade als eindimensionale Kurven dargestellt. Bei der Pfadplanung ist dies vorteilhaft, da der Abstand zwischen dem Roboter und einem möglichen Hindernis dem zweier Punkte entspricht. Gegebene Posen und Hindernisse im Arbeitsraum müssen bei der Pfadplanung in den Konfigurationsraum transformiert werden. Dieser Vorgang ist vor Allem bei dreidimensionalen Umgebungen rechenintensiv. Dem kann durch Approximation der Transformation entgegengewirkt werden. Allerdings geschieht dies auf Kosten der Genauigkeit der Abbildung.

Für die manuelle Programmierung eines Roboters wird meist der Arbeitsraum verwendet, da dort Objekte und Posen anschaulicher dargestellt werden (siehe

Abbildung 3.1). Generell können Pfadplanungsalgorithmen in beiden Räumen angewendet werden.

3.1.2 Reaktive Pfadplanung

Bei den höheren Schwierigkeitsgeraden des Robocup@Work werden in der Arena vor Beginn der Aufgabe Hindernisse hinzugefügt, die vorher nicht in der Karte verzeichnet worden. Solche unbekannten Hindernisse haben einen wesentlichen Einfluss auf die Planung. Im folgenden Abschnitt 3.1.3 wird das Problem in zwei Teilprobleme unterteilt: die globale und die lokale Pfadplanung. Danach wird die Integration beider Teilprobleme in ein gemeinsames Planungsverfahren in Abschnitt 3.1.4 thematisiert.

3.1.3 Globale und lokale Pfadplanung

Eine theoretische Möglichkeit, die Pfadplanung in einer teilweise unbekannten Umgebung zu realisieren, ist eine stetige Neuplanung. Durch die kontinuierliche Auswertung der, über die Sensoren erfassten Änderungen der Umgebung ist es möglich die Umgebungsrepräsentation stetig zu aktualisieren. Anhand dieser Daten wird vom Pfadplanungsalgorithmus ein neuer Pfad erstellt. Die Praktische Umsetzung dieses Vorgehens ist nicht realisierbar, da das ständige Planen eines Pfades zu rechenintensiv ist.

Um dieses Problem zu lösen, kann die Planung in die globale und die lokale Pfadplanung aufgeteilt werden. Die globale Pfadplanung betrachtet die gesamte Umgebung des Roboters mit statischen Hindernissen wie beispielsweise die Informationen aus einer vorgegebenen Karte. Dadurch ist es möglich bei einer vollständig bekannten Umgebung diesen Teil vor dem Ausführen der Roboterbewegung durchzuführen.

Die lokale Pfadplanung umfasst nur einen Ausschnitt der Umgebung, in dem sich der Roboter aktuell befindet und die dynamischen Daten, die er von seinen Sensoren erhält. Sie realisiert die reaktive Pfadplanung, da sie auf Änderungen in der Umgebungen reagiert. Durch die Reduktion auf einen kleinen Teil der Umgebung ist die lokale Pfadplanung ressourcenschonend genug, um während der Roboterbewegung durchgeführt zu werden. Dadurch kann sie in Echtzeit auf Änderungen reagieren und eine Kollision vermeiden. (vgl. [Sch14, S. 22 ff.])

3.1.4 Integration der Pfadplanung in ROS

In ROS wird für die gesamte Navigation des Roboters von robOTTO der `move_base` *node* eingesetzt. Daher wird diese *node* auch als *Navigation Stack* bezeichnet. Wie in Abbildung 3.2 zu sehen ist, besteht `move_base` aus verschiedenen funktionalen Einheiten, die Informationen von außen brauchen, wie zum Beispiel die Karte vom `map_server` oder die Sensordaten. Anhand dieser Informationen und einer Zielvorgabe (`move_base_simple/goal`) werden die Geschwindigkeiten (`cmd_vel`) für den Roboter errechnet. Zwei der Einheiten sind für die Pfadplanung zuständig: der `global_planner` für die globale und der `local_planner` für die lokale Planung. Diese funktionalen Einheiten stellen sogenannte austauschbare *Plug-Ins* dar. Diese ermöglichen, dass verschiedene Planungsansätze geschrieben und in der `move_base` eingesetzt werden können.

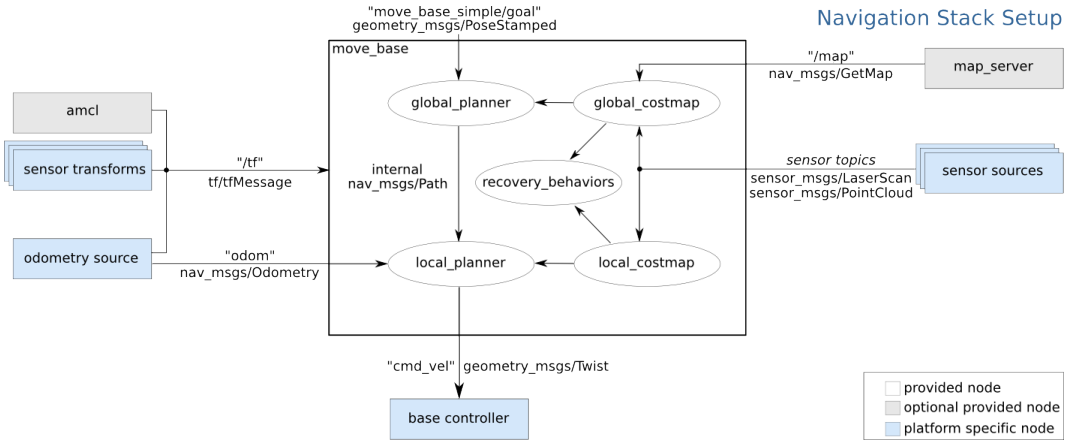

Abb. 3.2: Schematische Abbildung der "ROS move_base". [rosa]

Im robOTTO-Team werden die lokale und globale Navigation getrennt bearbeitet, um die Ressourcen des Teams besser zu nutzen. Daher liegt der Schwerpunkt dieser Arbeit bei der Untersuchung unterschiedlicher Plug-Ins des `global_planner`. Die Evaluation im Kapitel 4 bezieht sich auf die globalen Planungsalgorithmen, die in ROS bereits vorhanden sind oder von externer Seite als ROS *Plug-In* implementiert wurden.

Zunächst werden im Abschnitt 3.2 globale Pfadplanungsansätze für den globalen Planer vorgestellt und erläutert.

3.2 Pfadplanungsansätze für den globalen Planer

Das Reduzieren eines echten Roboters auf die grundsätzliche Pfadplanung ist ein komplexes Problem. Daher gibt es in diesem Feld viele Ansätze mit unterschiedlichen Herangehensweisen und vorausgesetzten Bedingungen, wie zum Beispiel, dass der Arbeitsraum zweidimensional ist und die Objekte darin Polygone sind. Alle Methoden können auf die drei folgenden Ansätze zurückgeführt werden: Roadmaps, Zelldekompositionen (eng.: *cell decomposition*) und Potentialfeldmethoden (eng.: *potential field*). Diese drei Ansätze sollen im Folgenden näher erläutert werden.

3.2.1 Roadmap

Der Ansatz der Roadmap ist, freie Punkte im Konfigurationsraum des Roboters miteinander zu verbinden. Somit wird ein Netz von befahrbaren Wegen erstellt: die sogenannte Roadmap. Anschließend, werden die Start- und Zielpose mit Punkten der Roadmap verbunden und die Pfadplanung wird auf das Finden der richtigen Punktverbindungen reduziert. Der gefundene Weg setzt sich aus drei Teilwegen zusammen: von der Startpose zur Roadmap, durch die Roadmap und von der Roadmap zum Ziel.

Für diesen Ansatz gibt es verschiedene Umsetzungen, wie beispielsweise Sichtbarkeitsgraphen oder Voronoi-Diagramme, die unterschiedliche Roadmaps erstellen.

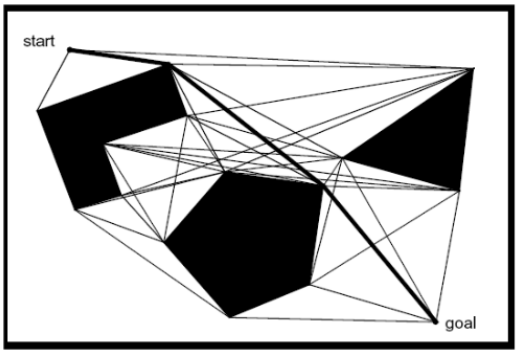

Abb. 3.3: Beispiel der Sichtbarkeitsgraphen, die schwarzen Linien stellen die Sichtlinien zwischen den Ecken der bereits aufgeblähten Objekte dar. Die dickere schwarze Linie ist der gewählte Weg. [US13]

Die Methode der Sichtbarkeitsgraphen arbeitet lediglich mit n-eckigen Hindernissen. Diese werden um mindestens die Hälfte des Roboterradius aufgebläht und deren Eckpunkte werden, sofern kein weiteres Hindernis in direkter Linie liegt, miteinander verbunden (siehe Abbildung 3.3). Somit wird die eigentliche Roadmap erstellt und mit dem Start- und Zielpunkt verbunden. Der Vorteil dieser Methode ist, dass je nach Aufblähung der Hindernisse ein Pfad mit möglichst kurzer Distanz gefunden wird. Allerdings führt eine zu geringe Vergrößerung zu Kollisionen und eine zu große zum Ausschließen von Pfaden, da fälschlicherweise Hindernisse erkannt werden.

Eine weitere Methode, eine Roadmap zu erstellen, bietet das Voronoi-Diagramm. Hierbei werden Punkte in die Karte eingetragen, die sowohl zur Kartenbegrenzung als auch zu den in der Karte liegenden Hindernissen den gleichen Abstand haben. Werden diese Punkte verbunden, ergeben sich Wege durch die Karte, die je nachdem ob die Hindernisse und Begrenzungen Polygone sind oder nicht, Geraden oder Kurven beschreiben. Diese Wege bieten dem Roboter die maximale Distanz zu Kollisionen, verlängern jedoch seinen Weg. Auch hier werden anschließend die Start- und Zielpose mit der Roadmap verbunden. (vgl. [Lat92, S. 12 ff.])

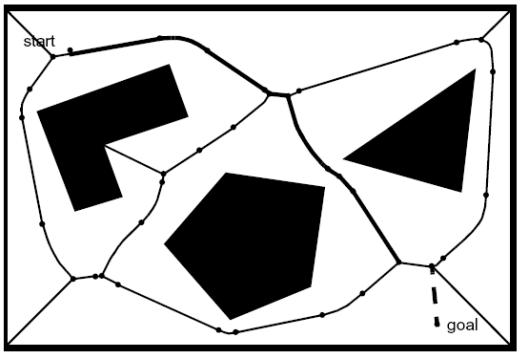

Abb. 3.4: Beispiel für das Voronoi-Diagramm, die schwarzen Linien stellen die Sichtlinen zwischen den Ecken der bereits aufgeblähten Objekte dar. Die dickere schwarze Linie ist der gewählte Weg. [US13]

3.2.2 Zelldekomposition

Der Ansatz der Zelldekomposition (eng.: *cell decomposition*) basiert auf der Zerlegung des freien Arbeitsraumes des Roboters in kleine, einfach zu beschreibende Regionen, den Zellen. Eine Unterteilung wird solange durchgeführt, bis

der Roboter innerhalb einer Zelle jede Konfiguration ungehindert einnehmen kann. Danach wird der *connectivity graph* erstellt, der die Nachbarschaftsbeziehungen zwischen den Zellen beschreibt. Die Knoten des Graphen repräsentieren die freien Zellen und werden nur miteinander verbunden, wenn zwei Zellen nebeneinander liegen. Auf diese Weise entsteht ein freier, kontinuierlicher Pfad.

Die Zelldekomposition gliedert sich in zwei Unterkategorien, die exakte und die approximierte Zelldekomposition:

- exakte Zelldekomposition teilt den freien Raum in Zellen ein, die diesem exakt[1] entsprechen. Die Abgrenzungen der einzelnen Zellen werden hier anhand bestimmter Kriterien gezogen, beispielsweise wenn die Umgebung einen Einfluss auf die Bewegung des Roboters hat.
- approximierte Zelldekomposition teilt den freien Raum in Zellen ein, deren Form vorgegeben ist, beispielsweise Quadrate. Die Übergänge sind hierbei von dem Roboter unabhängig und haben für die physikalische Gegebenheit keine Bedeutung.

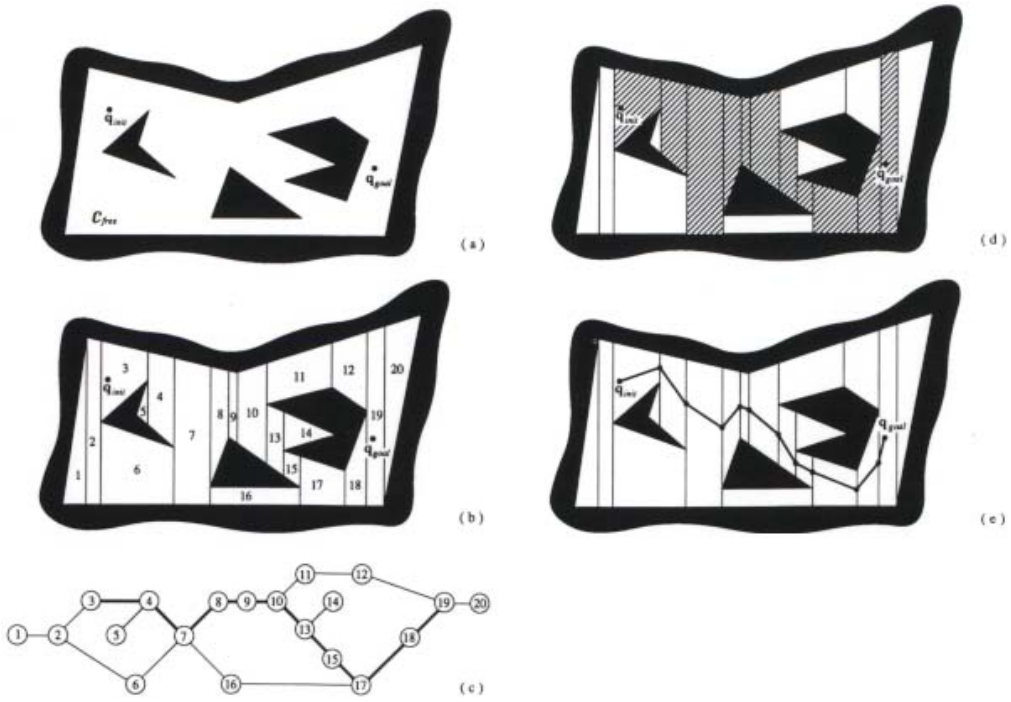

Abb. 3.5: Ablauf einer exakten Zelldekomposition. [Lat92]

[1] Wobei "exakt" sich in diesem Fall auf die gegebene Karte bezieht, nicht auf die reale Welt, welche nur approximiert werden kann

In Abbildung 3.5 wird der Ablauf der exakten Zelldekomposition gezeigt. Hier ist der freie Raum von einem Polygon umgeben und es befinden sich drei polygonale Hindernisse in ihm (a). Als erstes wird der freie Raum in Zellen, in Form von Dreiecken und Trapezen, eingeteilt (b). Danach wird der *connectivity graph* erstellt (c) und durchsucht (dickere Linie). Auf den Arbeitsraum abgebildet, ergibt der Graph einen Kanal (d), aus dem ein Pfad gewonnen wird, zum Beispiel durch Verbindungen der Mittelpunkte der Grenzen der einzelnen Zellen (e).

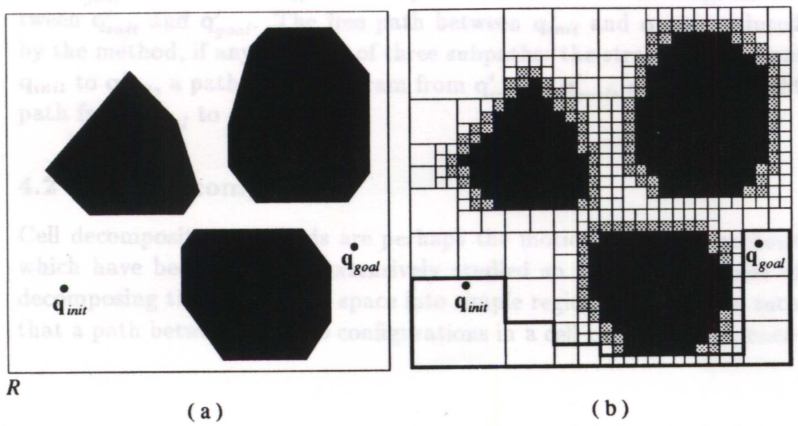

Abb. 3.6: Einteilung des Arbeitsraumes in Zellen bei einer approximierten Zelldekomposition. Der Kanal vom Start zum Ziel ist dicker umrandet. [Lat92]

Die Abbildung 3.6 zeigt die approximierte Zelldekomposition innerhalb des Arbeitsraums. Der Raum wird rekursiv in kleinere Rechtecke zerteilt, wodurch bei jeder Zerlegung vier neue identische Rechtecke entstehen.[2] Wenn sich in den Zellen sowohl Hindernisse als auch freier Raum befinden, geht die Zerlegung weiter, sodass an den Rändern von Hindernissen eine feinere Auflösung entsteht. Ab einem gewissen Feinheitsgrad wird die Dekomposition gestoppt und der *connectivity graph* von der Start- zu der Zielzelle gesucht. Ist diese Suche erfolgreich, wird ein Pfad erstellt. Andernfalls ist entweder die Auflösung des Verfahrens nicht fein genug oder es existiert kein Pfad. Bei manchen Methoden wechselt sich die Suche mit der Zerlegung der Zellen ab: immer wenn die Suche nach dem *connectivity graph* fehlschlägt, wird die Zerlegung der Zellen verfeinert und erneut gesucht. [Lat92, S. 14 ff.]

[2]Dieses spezielle Verfahren wird *quadtree* genannt, da ein "Suchbaum" entsteht, bei dem von jedem Ast vier neue Äste ausgehen.

3.2.3 Potentialfeldmethode

Die Potentialfeldmethode ist ein weiterer klassischer Ansatz für die Pfadplanung. Hierbei werden, ausgehend von der Roboterkonfiguration, künstliche Potentialkräfte errechnet, welche die Bewegung bestimmen. (vgl. [Bub07, S. 3 f.]) Dabei ergeben sich diese Kräfte aus zwei verschiedenen Potentialfeldarten, dem anziehenden Potentialfeld, das zum Ziel führt und den abstoßenden Potentialfeldern, die von den Hindernissen ausgehen, siehe Abbildung 3.7.

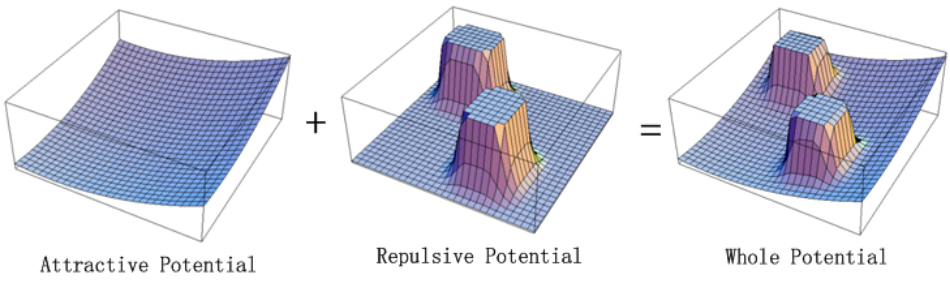

Abb. 3.7: Bildliche Erläuterung der Potentialfeldmethode. Links: Anziehendes Potential des Ziels, Mitte: Abstoßendes Potential der Hindernisse, Rechts: Kombination bzw. entstandenes Potentialfeld. [img]

Zusätzlich wird eine maximale Entfernung bestimmt, bei der das Potential nur noch als linear wachsend definiert wird. So wird verhindert, dass die anziehende Kraft endlos wächst und ab der maximalen Entfernung konstant bleibt. (vgl. [PDB14])

Das Potentialfeld eines Hindernisses wirkt abstoßend. Zu dessen Berechnung wird die „Force Inducing an Artificial Repulsion from the Surface of the obstacle“ kurz FIRAS eingeführt, welches in [Kha86] näher erläutert wird.

Es handelt sich um ein nicht negatives differenzierbares Potential. Es findet eine Fallunterscheidung statt, ob der Mindestabstand zum Hindernis überschritten wird. Ist das der Fall, so wird das Potential null. Durch diese Unterscheidung wird verhindert, dass jedes Hindernis, egal wo es sich befindet, eine Wirkung auf die mobile Plattform hat. (vgl. [Bub07])

3.3 Vergleich der Pfadplanungsalgorithmen

Um einen Vergleich von globalen Pfadplanungsalgorithmen anstellen zu können, müssen Kriterien ermittelt werden, anhand derer eine Beurteilung der unterschiedlichen Verfahren möglich ist.

Kollisionsfreier Weg

Dies ist das wichtigste Kriterium. Der geplante Pfad darf zum Zeitpunkt der Planung keine Kollisionen mit der Umgebung und möglichen Hindernissen aufweisen. Ungewollte Berührungen zwischen Roboter und Umwelt, die bereits auf dieser Ebene der Planung zustande kommen, sind ein Ausschusskriterium.

Länge des Pfades

Das Ergebnis der Wegplanung sollte ein Pfad sein, bei dem die Gesamtlänge der Teilstrecken minimal ist. Auch bei mehreren möglichen Wegen sollte immer der kürzere Weg gewählt werden. (vgl. [Hop92, S. 35 f.])

Minimaler Abstand zu Hindernissen

Um den Weg zum Ziel zu optimieren, sollte die Pfadplanung minimale Abstände zu Hindernissen berücksichtigen. Dabei führt ein zu geringer Mindestabstand zu ungewollten Kollisionen.

Rechenzeit zur Wegbestimmung

Ein wichtiges Kriterium, um die Performance der Wegplanung beurteilen zu können, ist die Rechenzeit. Da alle Wettbewerbe zeitlich begrenzt sind und viele Planungen zwischen den einzelnen Arbeitsschritten stattfinden, ist ein performanter Algorithmus zu bevorzugen.

4 Evaluation

In diesem Kapitel werden die *Plug-Ins* für den `global_planner` der `move_base` miteinander verglichen. Dazu werden verschiedene Karten mit steigendem Komplexitätsgrad gewählt, in denen ein Weg geplant wird. In diesen wird der Roboter auf eine feste Anfangsposition gestellt und muss ein bestimmte Zielposition erreichen. Durch einen, im Zuge dieser Arbeit erstellten, *node* wird der Planungsvorgang überprüft und ausgewertet.

Zunächst wird eine Vorauswahl der `global_planner` *Plug-Ins* getroffen. Danach werden die Parameter der verbleibenden Planer aufgezeigt, um eine Vergleichbarkeit zu gewährleisten. Anschließend wird der zugrundeliegende Versuchsaufbau erklärt und die Test-Karten beschrieben. Die Planer werden schließlich getestet und es erfolgt eine Auswertung der Ergebnisse.

4.1 Vorauswahl der globalen Planer

Neben den globalen Planern, die mit der `move_base` ausgeliefert werden, gibt es noch weitere Planer, die von den Entwicklern unter Open-Source-Lizenzen veröffentlicht werden.

Auch diese Planer *Plug-Ins* werden für den Zweck der Arbeit ausgewertet. Aufgrund von Missachtung von Dokumentationsrichtlinien kann bei einigen Planern nicht nachvollzogen werden, ob diese nur für bestimmte Szenarien oder andere ROS-Versionen entwickelt wurden. Daher sind die Planer partiell nicht kompatibel und können nicht als *Plug-In* in die `move_base` geladen werden.

Um diese Planer einzusetzen, müssen unter anderem Änderungen am Quellcode vorgenommen und Probleme behoben werden. Da dies nicht Ziel der Arbeit ist und die Behandlungen einzelner Planer den Rahmen der Arbeit übersteigen würden, wurde in Tabelle 4.1 eine Vorauswahl getroffen.

Tabelle 4.1: Vorauswahl der globalen Planer. [selbst erstellt]

Planer	Dokumentation	Kompilier-Fähigkeit	Testergebnisse	Quelle	Stand
Navfn	✓	✓	sinnvoll	[nav]	21.10.14
Global-Planner	✓	✓	sinnvoll	[glo]	21.10.14
Voronoi	-	-	-	[vor]	Commit: 16.09.15
OMPL-Planner	-	✓	wird nicht erkannt	[omp]	Commit: 24.11.14
Srl-Global-Planner	✓	✓	findet keinen Weg	[srl]	Commit: 08.05.15
Lattice-planner	✓	✓	findet keinen Weg	[lat]	Commit: 16.09.15

Aus der Tabelle geht hervor, dass nur die ROS-eigenen Planer sinnvolle Ergebnisse erzeugen. Daher werden Sie im Folgenden näher erläutert und in den Karten getestet.

4.1.1 Navfn

Dem *Navfn*-Planer liegt der *Dijkstra*-Algorithmus zugrunde, dieser basiert auf dem Ansatz der Zelldekomposition (siehe Abschnitt 3.2.2). Der *Navfn*-Planer ist veraltet und wird somit nicht mehr weiterentwickelt. Dennoch wird er weiterhin mit der `move_base` ausgeliefert und ist dort als Standard-Planer eingestellt.

Die vollständige Dokumentation zum *Navfn*-Planer befindet sich in [nav].

4.1.2 Global_Planner

Der *Global_Planner* wurde entwickelt, um den veralteten *Navfn*-Planer abzulösen. Nach Aussage des Entwicklers bietet er eine größere Performance als der *Navfn*-Planer und behebt einige Fehler, die während der Planung auftreten.

Der *Global_Planner* arbeitet in den Standard-Einstellungen ebenfalls mit dem *Dijkstra*-Algorithmus, lässt sich aber wahlweise auch auf den *A**-Algorithmus

(auch *A-Stern* oder *A-star* genannt) umschalten. Dieser stellt eine Erweiterung des *Dijkstra* dar. Er fügt ihm eine gerichtete Suche nach dem Ziel hinzu, dass heißt, dass er den Abstand zum Ziel einbezieht und daher den Raum, der nicht zwischen Start und Ziel liegt, vernachlässigt. Beide Einstellungen werden bei der Evaluation verglichen, daher wird zur Kennzeichnung *GP/Dijkstra* und *GP/A-star* verwendet.

Die vollständige Dokumentation zum *Global_Planner* befindet sich in [glo].

4.2 Parameter

Um Vergleichbarkeit zu schaffen, müssen die Planer möglichst mit gleichen Parametern betrieben werden. Der Großteil der einstellbaren Parameter der `move_base` bezieht sich auf andere Unterknoten wie die `costmap`, welche daher für alle *Plug-Ins* gleich bleiben.

Tabelle 4.2: Parameter der Planer *Plug-Ins*. [selbst erstellt]

Navfn	Global_Planner	gemeinsame Parameter
`planner_window_x`	`use_dijkstra`	`allow_unknown`
`planner_window_y`	`use_quadratic`	`default_tolerance`
	`use_grid_path`	
	`old_navfn_behavior`	

In der Tabelle 4.2 sind die Parameter aufgeführt, die direkt von den Planern eingelesen werden. Diese haben folgende Bedeutung:

- `allow_unknown` erlaubt bzw. verbietet das Fahren durch unbekannte Gebiete der Karte. Da alle Karten vollständig bestimmt sind und die Regeln des Wettbewerbs bisher keine unbekannten Gebiete vorsehen, ist dieser Parameter ausgeschaltet.
- `planner_window_x` bzw. `planner_window_y` kann genutzt werden, um den Planer innerhalb der Karte zu begrenzen. Da eine Begrenzung des Planers nicht vorgesehen ist, werden diese Parameter nicht verwendet.
- `default_tolerance` gibt dem Planer einen Abstand zum Ziel vor. Diesen muss er mindestens erreichen, bevor er die Planung beendet. Dieser

Parameter ist bei beiden Planern vorhanden und wurde bei beiden auf dem Standardwert 0,0 belassen.

- `use_dijkstra` lässt den *Global_Planer* zwischen dem *Dijkstra* und dem *A-star* Planungsansatz wechseln.
- `use_quadratic` schaltet die Berechnung der Potentiale des Weges in der *costmap* zwischen dem normalen und einem einfacheren Ansatz um. Hier wird der normale Ansatz genutzt, da auch *Navfn* diesen verwendet.
- `use_grid_path` lässt den *Global_Planer* nur Wege planen, die aus Geraden bestehen. *Navfn* verfügt nicht über einen solchen Parameter, daher wird dieser nicht gesetzt.
- `old_navfn_behavior` lässt den *Global_Planer* das Verhalten des *Navfn* nachahmen. Da *Navfn* selber getestet wird, wäre das Setzen dieses Parameters redundant. Außerdemist es nicht das Ziel zu testen, wie genau der *Global_Planer* den *Navfn* nachahmen kann.

4.3 Virtual Robot Experimentation Platform (v-rep)

Von seinen Erfindern wird *v-rep* als das "Schweizer Taschenmesser" unter allen Roboter-Simulatoren bezeichnet. [v-r] Ein Simulator ist für Evaluationen von großem Vorteil, da die Randbedingungen sich nicht ändern und eine Abfolge von Versuchen beschleunigt werden kann. *V-rep* kann sowohl mit ROS arbeiten als auch den KUKA youBot simulieren (siehe Abbildung 4.1) und ist daher eine exzellente Wahl für die Tests.

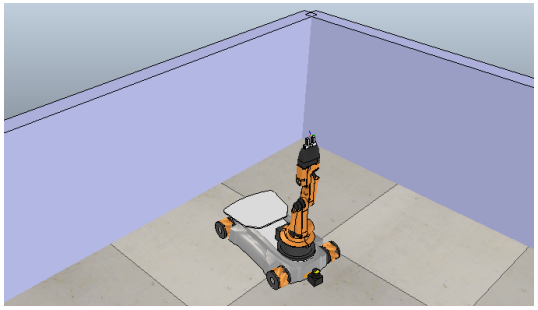

Abb. 4.1: Der KUKA youBot in der Simulationsumgebung v-rep. [selbst erstellt]

4.4 Versuchsaufbau

In Abbildung 4.2 ist Versuchsaufbau dargestellt. Alle Untersuchungen werden auf einem Dell Lantitude E6430 AGT ausgeführt. Der Rechner verfügt über einen Intel CORE i7-3540M Prozessor mit zwei Kernen die jeweils eine 3.0 GHz Taktung besitzen und einen DDR-3 Arbeitspeicher von 8 GB mit 1600 MHz Taktung. Auch die Simulation findet mit dem in Abschnitt 4.3 beschriebenen Simulator *v-rep* auf diesem Computer statt.

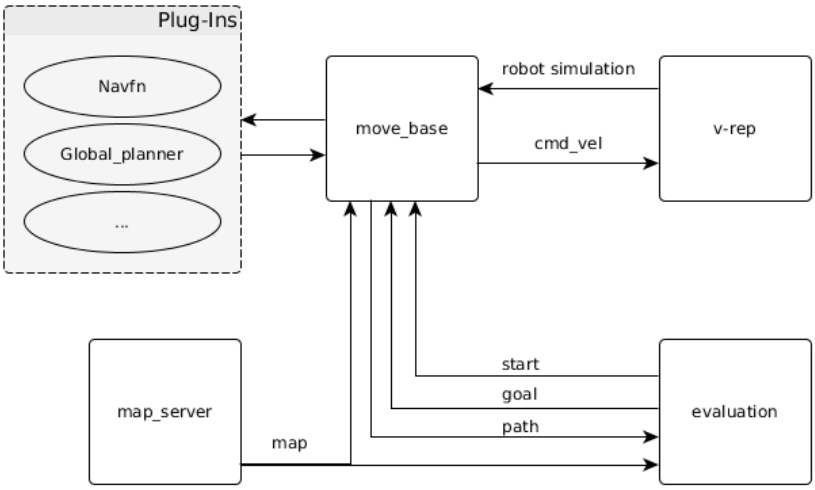

Abb. 4.2: Versuchsaufbau. [selbst erstellt]

Der zentrale Punkt der Untersuchungen ist die ROS `move_base`, die bereits im Abschnitt 3.1.4 beschrieben wurde. Hier werden die einzelnen Planer geladen, die den Weg berechnen. Dafür braucht dieser *node* einige Informationen. Die erste benötigte Information ist die Karte, die ihm vom sogenannten `map_server` mit den entsprechenden Meta-Daten zur Verfügung gestellt wird. Der Start- und Zielpunkt wird der `move_base` von dem `evaluation`-*node* vorgegeben, welcher im Zuge dieser Arbeit geschrieben wurde.

Der `evaluation`-*node* ist in der Lage, die Daten des geplanten Pfads auszulesen, die für den Vergleich der Planer wichtig sind, siehe Abschnitt 3.3. Zudem werden weitere Information ausgelesen, die Rückschlüsse auf die Berechnungen der Planer zulassen: die Anzahl errechneter Zwischenpunkte und deren maximaler sowie durchschnittlicher Abstand voneinander. Mit diesen Werten kann zum Beispiel festgestellt werden, ob es eine Korrelation zwischen errechneten Zwischenpunkten und Berechnungszeit gibt.

Alle weiteren Konfigurationsparameter werden der `move_base` direkt von ROS zur Verfügung gestellt und werden daher nicht dargestellt. Nach der Berechnung übergibt der *node* dem Roboter in *v-rep* die *command velocities* (kurz: `cmd_vel}`). Diese stellen Geschwindigkeiten da, die der Roboter bezogen auf alle Raumfreiheitsgrade annehmen muss. Bei einem holonomen Roboter in der Ebene sind dies die Geschwindigkeiten für seine Längs- und Querachse sowie die Drehgeschwindigkeit um seine Hochachse.

4.5 Karten

Für die Tests wurden vier Karten ausgewählt, auf denen jeweils ein Pfad mit festem Start- und Endpunkt geplant werden soll. Die Karten *Offener Raum* und *German Open 2015* stellen dabei realistische Szenarien dar, siehe Abbildung 4.3 und 4.4.

Offener Raum ist ein einfacher Test zur Durchquerung eines Raumes, bei der sich keine Hindernisse zwischen dem Start und dem Ziel befinden. *German Open 2015* wurde während des RoboCup-Wettbewerbs aufgenommen und nachbearbeitet. Die grauen Kästen stellen größtenteils Tische dar, die der Aufnahme und Ablage von Bauteilen dienen. Die Pfadplanungsaufgabe besteht dabei darin, vom offiziellen Start der Arena zum letzten Tisch zu fahren.

Diesen realistischen Karten stehen die beiden zufallsgenerierten Labyrinthe gegenüber, die nicht im Wettbewerb vorkommen. Diese machen die Planung bei gleicher Kartengröße und Auflösung sowie ähnlichem Aufbau vergleichbar. Die Labyrinthe unterscheiden sich in der Länge ihrer Lösungswege. Die Planer müssen dementsprechend mehr Zwischenpunkte berechnen, was die Planungszeit verlängern könnte, siehe Abbildung 4.5 und 4.6.

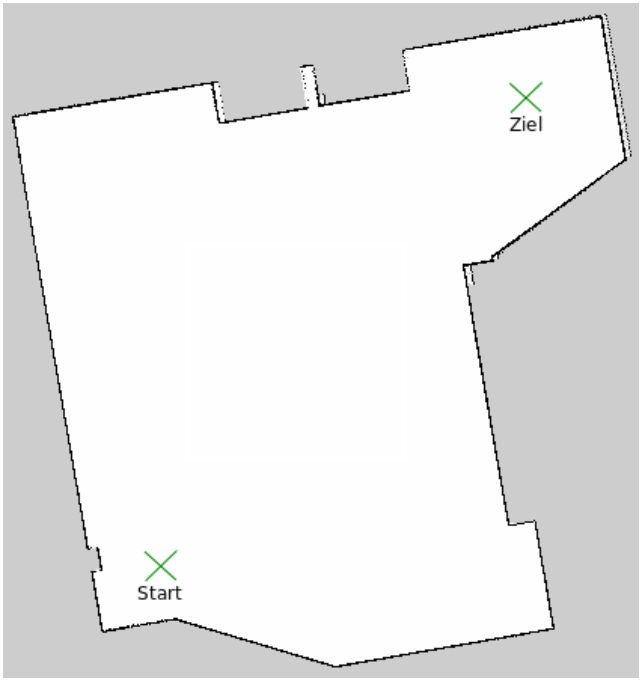

Abb. 4.3: Karte - Offerner Raum. [selbst erstellt]

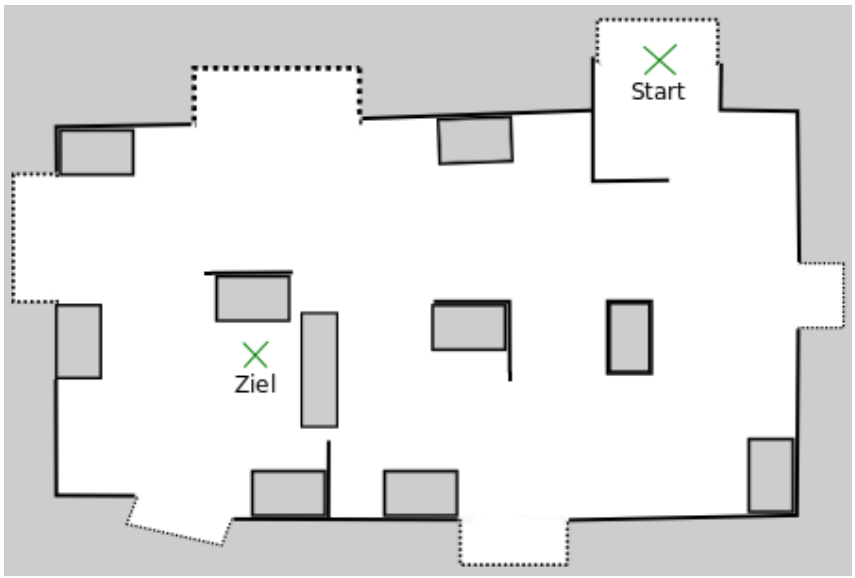

Abb. 4.4: Karte - German Open 2015. [selbst erstellt]

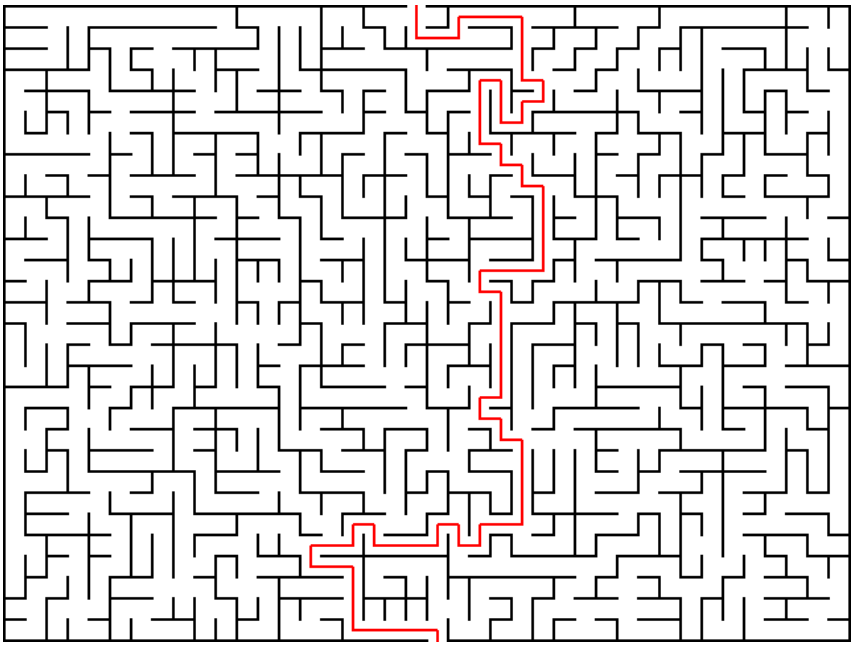

Abb. 4.5: Karte - Labyrinth mit kurzem Weg. [selbst erstellt]

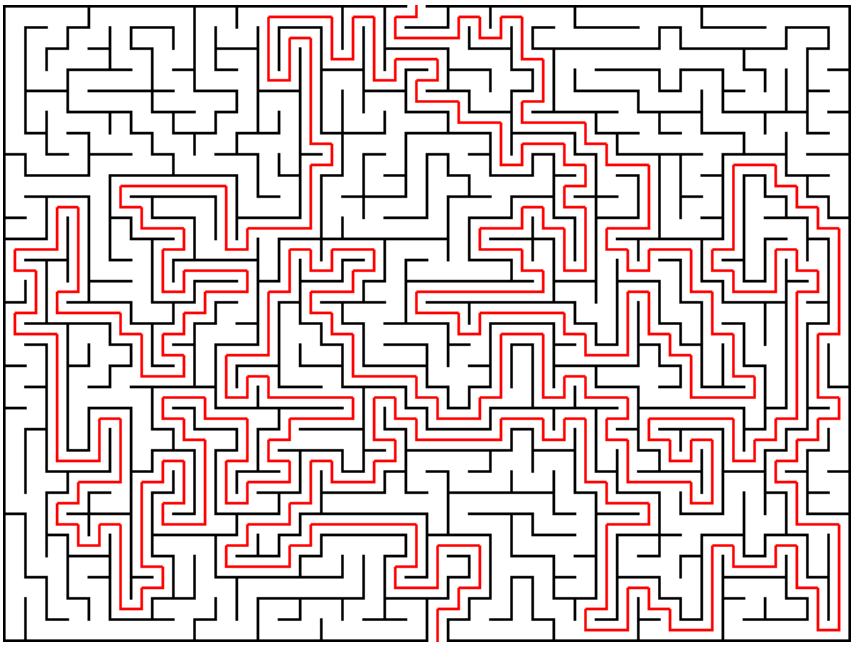

Abb. 4.6: Karte - Labyrinth mit langem Weg. [selbst erstellt]

In der Tabelle 4.3 werden die Größe und die Auflösung der Karten genannt. Außerdem wird eine Bewertung der folgenden der Eigenschaften der Karten vorgenommen:

- die durchschnittliche Weglänge möglicher Pfade,
- die Hindernismenge, die sich im direkten Weg zwischen Start und Ziel befindet,
- die nötigen Wegänderungen bzw. Wegabzweigungen, die berücksichtigt werden müssen und ein Schwierigkeitsgrad,
- der sich aus diesen Eigenschaften ergibt.

Tabelle 4.3: Übersicht über Eigenschaften der Karten. Mehr ● bedeuten ein höheres Anforderungsniveau. [selbst erstellt]

	offener Raum	German Open 2015	Labyrinth kurzer Weg	Labyrinth langer Weg
Größe [px]	422×448	449×302	642×482	642×482
Auflösung [m/px]	0,02	0,02	0,1	0,1
Weglänge	●	●●	●●	●●●
Hindernisse	●	●●	●●●	●●●
Wegänderungen	●	●●	●●●	●●●
Schwierigkeit	●	●●	●◖	●●●

4.6 Auswertung

In diesem Abschnitt werden die Planer anhand der Kriterien aus Abschnitt 3.3 getestet und die Ergebnisse in der Tabelle 4.4 aufgezeigt. Zusätzlich werden errechnete Zwischenpunkte und deren Abstand zueinander eingetragen, um mögliche Zusammenhänge zu Pfadlänge und Planungszeit darzustellen. Danach erfolgt eine Auswertung anhand dieser Daten.

Tabelle 4.4: Vergleich der Ergebnisse der Planer auf den Karten. [selbst erstellt]

Karte	Planer	Zeit für die Planung	Länge des Pfades	min. Abstand	Zwischen-punkte	Abstand zw. Punkten
Offener Raum	Navfn	61,20 ± 30,53 ms	8,31 m	0,71 m	831	10,01 ±0,73 mm
	GP/Dijkstra	74,38 ± 25,04 ms	8,12 m	0,70 m	814	9,99 ±0,38 mm
	GP/A-star	77,78 ± 34,65 ms	8,60 m	0,65 m	315	24,56 ±6,59 mm
German Open	Navfn	61,86 ± 29,89 ms	10,57 m	0,44 m	1057	10,01 ±0,65 mm
	GP/Dijkstra	330,41 ±144,37 ms	10,77 m	0,45 m	1079	9,99 ±0,31 mm
	GP/A-star	-	-	-	-	-
Labyrinth kurzer Weg	Navfn	90,06 ± 30,52 ms	102,24 m	0,75 m	2041	50,12 ±2,75 mm
	GP/Dijkstra	232,31 ±132,75 ms	105,07 m	0,75 m	2103	49,88 ±1,23 mm
	GP/A-star	-	-	-	-	-
Labyrinth langer Weg	Navfn	-	-	-	-	-
	GP/Dijkstra	253,33 ±101,70 ms	766,27 m	0,75 m	15327	49,88 ±0,44 mm
	GP/A-star	-	-	-	-	-

In Tabelle 4.4 fällt besonders auf, dass nur für den *GP/Dijkstra* Ergebnisse auf allen Karten existieren. Dies begründet sich darin, dass nur dieser Planer Wege fehlerfrei in den realistischen Karten und den Labyrinthen berechnet. In der *A-Star*-Variante findet er jedoch auf komplexeren Karten keine Wege und gibt nur eine Fehlermeldung aus, die in Abbildung 4.7 dargestellt wird. Im Quellcode tritt dieser Fehler durch eine falsche Berechnung der gerichteten Suche auf. Der *Navfn*-Planer funktioniert nur auf den realistischen Karten weitestgehend ohne Probleme. Bei Labyrinthen mit langen Lösungswegen berechnet er nur einen Teil der Lösung und bricht ohne Meldung oder erkennbaren Grund die Planung ab.

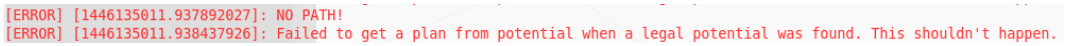

Abb. 4.7: Fehlermeldung des *Global_Planners* in der *A-star*-Variante. [selbst erstellt]

Kollisionsfreie Wege, sofern vorhanden, wurden von den getesteten Planern erfolgreich gefunden. Dies gelang auch, wenn Start- und Zielposition innerhalb der Karte beliebig verändert wurden. Zu keinem Zeitpunkt wurde ein Pfad durch ein Hindernis oder in der Nähe eines Hindernisses geplant, sodass ein berührungsfreier Pfad gefährdet war.

In zwei von drei Karten, in denen sowohl *GP/Dijkstra* als auch der *Navfn*-Planer Wege gefunden haben, hat letzterer den kürzeren Pfad berechnet. *Im Labyrinth mit kurzem Lösungsweg* ergab sich eine Differenz von mehreren Metern. Ihre Auswirkung auf die Fahrzeit einer Roboterplattform, wie dem youBot, beträgt jedoch nur wenige Sekunden. Im Verlauf eines Wettkampfes werden viele solcher Wege geplant und führen langfristig zu einer nicht zu vernachlässigenden Zeitersparnis, die Einfluss auf die Platzierung hat.

Die Abbildung 4.8 und 4.9 stellen die minimalen Abstände zu allen Hindernissen bezogen auf den Weg vom Start zum Ziel dar. Kritische Bereiche, wie beispielsweise Abstände unter dem Radius des youBots von 0,35 m werden während der Fahrt nicht erreicht. Lediglich bei der Zieleinfahrt kann dies nicht verhindert werden. Somit sind die geplanten Wege, frei von Berührungen mit der Umwelt. Da die Planer nahezu den selben Abstand zu den Hindernissen berechnen, sind sie in diesem Kriterium gleichwertig.

Die Planungszeit, die bei den einzelnen Wegfindungen gemessen wurde, unterliegt großen Schwankungen. Teilweise wurden Standardabweichungen von über

50 % erreicht. Dies begründet sich in der periodischen Generierung von Aufrufen der *Callback*-Funktionen, wie beispielsweise die in dem `evaluation`-*node* die Zeit der Planung stoppt. Das hat zur Folge, dass der Aufruf dieser Funktion zeitversetzt zur erfolgreichen Planung des Pfades stattfindet. Dies liegt daran, dass ROS kein echtzeitfähiges System ist. Im Vergleich zum *Global_Planner* berechnet der *Navfn*-Planer in größeren Karten schneller einen Weg zum Ziel.

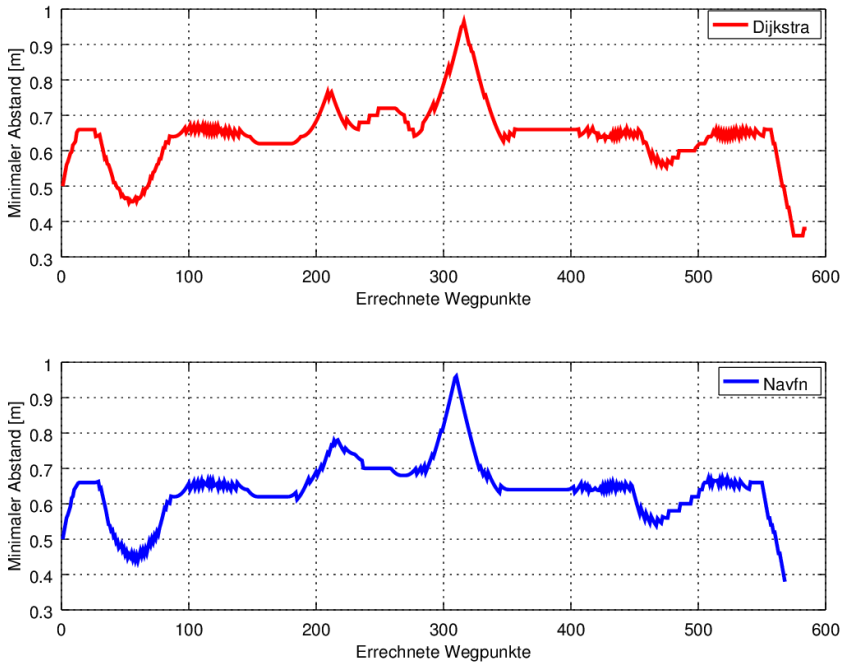

Abb. 4.8: Vergleich des minimalen Abstands auf der Karte - German Open 2015. [selbst erstellt]

Vor der Durchführung der Tests wurde im Abschnitt 4.5 angenommen, dass bei gleicher Kartengröße und Auflösung der beiden Labyrinthe die Brechungszeit mit der Anzahl der Zwischenpunkte korrelieren müsste. Dies bestätigte sich nach der Durchführung nicht. Der Unterschied beträgt ca. 20 ms bei einer Differenz von 13224 Zwischenpunkten.

Durch die Tatsache, dass nur der *GP/Dijkstra* in der Lage ist, zuverlässig für jede Karte einen Weg, sofern dieser existiert, zu finden, wird dieser Planer zum Favoriten. Zuverlässigkeit ist in einer Wettkampfsituation ein entscheidender Faktor. Die Arenen sind vor den Wettbewerbstagen kaum oder gar nicht bekannt und Wege müssen aus allen erdenklichen Posen innerhalb einer Karte geplant werden. Ein unvorhergesehener Ausfall der Pfadplanung ist unter

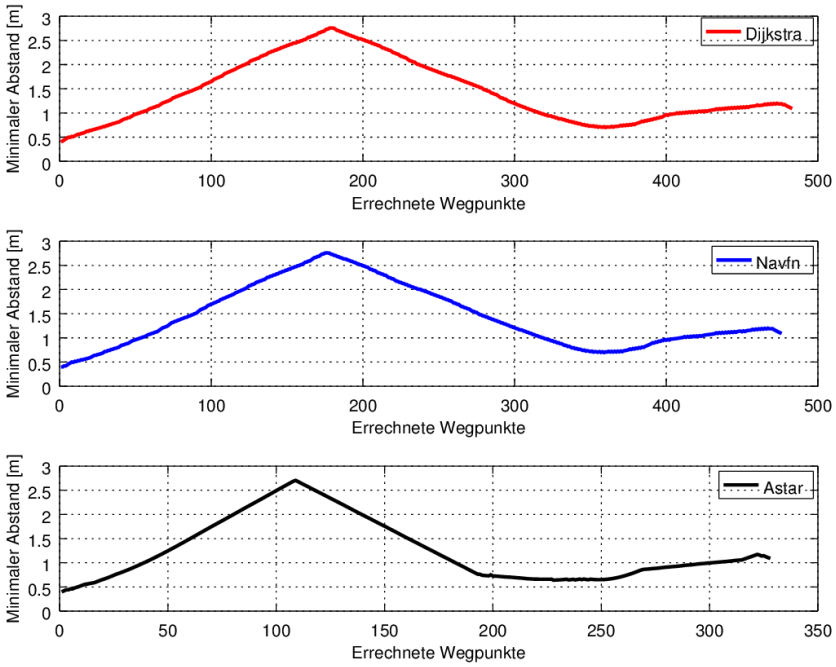

Abb. 4.9: Vergleich des minimalen Abstands auf der Karte - freier Raum. [selbst erstellt]

solchen Bedingungen fatal. Die Benutzung des *GP/Astar* ist daher aufgrund von Unzuverlässigkeit auszuschließen. Auch der *Navfn*-Planer hat sich nicht bei allen Karten bewährt und wurde bei langen Lösungswegen unzuverlässig. Jedoch berechnete er bei realistischen Karten schnell einen Weg, welcher aufgrund seiner kurzen Strecke im Vergleich zu den Anderen zu bevorzugen ist. Unter der Voraussetzung, dass der Planer überarbeitet wird um eine stabile Wegplanung zu gewährleisten, kann der *Navfn*-Planer verwendet werden. Allerdings ist er, wie im Abschnitt 4.1.1 beschrieben, veraltet und wird von offizieller Seite nicht mehr unterstützt. Daher und aufgrund seiner guten und stabilen Ergebnisse in den Tests ist der *Global_Planner* den übrigen *Plug-Ins* vorzuziehen und sollte vom Team robOTTO verwendet werden.

5 Fazit und Ausblick

Mit der `move_base` liefert ROS ein Navigationswerkzeug, dass die Möglichkeit bietet, die globale Pfadplanung von austauschbaren, mitgelieferten oder fremden *Plug-Ins* übernehmen zu lassen. Eine Reihe dieser Planer wurden im Zuge dieser Arbeit getestet. Mit dem Ergebnis, dass viele der verfügbaren Planer nicht oder nicht mehr kompatibel sind, wie in Tabelle 4.1 aufgelistet. Dies begründet sich in der Einstellung der Entwicklung der *Plug-Ins*. Daher sind sie nicht mit der getesteten ROS Version kompatibel oder wurden nur für bestimmte Szenarien bzw. Projekte entworfen.

Die verbleibenden Planer wurden anhand der Kriterien aus Abschnitt 3.3 verglichen. Im Zuge dieses Vergleichs wurde unter anderem ein eigener ROS *node* geschrieben, der es ermöglicht hat, die für den Vergleich notwendigen Daten zu evaluieren. Diese Ergebnisse befinden sich in der Tabelle 4.4 und zeigen auf, dass sich vor allem der *Global_Planner* in der Einstellung *GP/Dijkstra* für die Bedürfnisse des robOTTO-Teams eignet, da dieser sehr zuverlässig ist und praktische Ergebnisse erzielt. Auch der *Navfn*-Planer kann durch schnelle Berechnungszeiten und kurze Wege überzeugen, ist aber nicht in der Lage, bei sehr langen Strecken fehlerfreie Ergebnisse zu liefern und birgt damit ein zu großes Risiko in einem Wettbewerb.

Daher sollte für die Zukunft der *Global_Planer/Dijkstra* vom robOTTO-Team verwendet werden.

Inhalt der Arbeit war hauptsächlich die Implementierung bestehender globaler Pfadplaner in die `move_base`, ohne nach Gründen für Fehlverhalten dieser zu suchen. Die Behebung von Fehlern war im Rahmen der Arbeit nicht vorgesehen. Durch den vorzeitigen Wegfall der meisten *Plug-Ins* aufgrund von Inkompatibilität basieren die getesteten Planer alle auf den Planungsansatz der Zelldekomposition. Durch Fehlerbehebung im Quellcode der *Plug-Ins* könnte die Basis an benutzbaren Planern und somit auch die Vielfalt an Planungsansätzen vergrößert werden. Möglicherweise würde sich so aus einem der Planer eine Basis ergeben, auf der ein eigenen Ansatz entwickelt werden kann.

Auf jeder Karte wurde jeweils ein fester Start- und Zielpunkt vorgegeben um den geplanten Pfad zwischen diesen zu vergleichen. Dadurch wurde auf den Karten nur jeweils ein Weg geplant. Allerdings gibt es bei diesen Karten unendlich viele mögliche Posen, aus denen der Roboter starten bzw. in denen er enden kann. Eine Evaluation von kompletten Wettkämpfen mit der entsprechenden Anzahl von Pfadplanungen könnte zu weiteren Erkenntnissen über die Pfadplanungsalgorithmen führen. Um die entsprechenden Datensätze zu erfassen, sollte der in dieser Arbeit erstellte `evaluation`-*node* weiterhin genutzt werden.

Literaturverzeichnis

[ar-] *ARToolKit.* – `http://www.hitl.washington.edu/artoolkit/` Stand: 02.11.2015

[ASK15] Algfoor, Z. A. ; Sunar, M. S. ; Kolivand, H.: A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games. In: *International Journal of Computer Games Technology Volume 2015* (2015)

[Bub07] Bubeck, A.: Implementierung eines reaktiven Pfadplanungsalgorithmus für nicht holonome mobile Plattformen in industrieller Umgebung / Otto-von-Guericke-Universität Magdeburg. 2007. – Forschungsbericht

[glo] *Srl_global_planner.* – `http://wiki.ros.org/global_planner` Stand: 08.10.2015

[Han12] Hanses, M.: *Steuerungsarchitektur für die Bahnführung eines mobilen Roboters innerhalb eines Produktionssystems*, Otto-von-Guericke-Universität Magdeburg, Diplomarbeit, jul 2012

[Hop92] Hoppen, P.: *Autonome Mobile Roboter - Echtzeitnavigation in bekannterr und unbekannter Umgebung.* BI Wissenschaftsverlag, 1992

[img] *My point cloud.* – `https://taylorwang.wordpress.com/2012/04/06/collision-free-path-planning-using-potential-field-method-for-highly-redundant-manipulators/` Stand: 05.10.2015

[Kha86] Khatib, O.: *3. The International Journal of Robotics Research 5.* 1986

[kuk] *KUKA youBot*. – http://www.youbot-store.com/media/catalog/product/cache/1/image/9df78eab33525d08d6e5fb8d27136e95/3/j/3j8g3115-kopie518273ed20b04.png Stand: 04.10.2015

[lat] *Lattice_planner*. – https://github.com/marinaKollmitz/lattice_planner Stand: 22.10.2015

[Lat92] Latombe, J.-C.: *Robot Motion Planning*. Kulwer Academic Publishers, 1992

[LP83] Lozano-Perez, T.: In: *IEEE TRANSACTIONS ON COMPUTERS, Vol C-32*. IEEE, 1983

[nav] *Navfn*. – http://wiki.ros.org/navfn Stand: 08.10.2015

[NKH+14] Nowak, W. ; Kraetzschmar, G. ; Hochgeschwender, N. ; Bischoff, R. ; Kaczor, D. ; Hegger, F. ; Carstensen, J.: *Robocup@Work RuleBook*. Walter Nowak, 2014. – http://www.robocupatwork.org/download/rulebook-2014-07-06.pdf Stand: 06.05.2015

[omp] *Move_base_ompl*. – https://github.com/windelbouwman/move-base-ompl Stand: 22.10.2015

[PDB14] Prof. Dr. Bittel, O.: *Pfadplanung: Potentialfeldmethoden*. HTWG Konstanz, 2014. – http://www-home.htwg-konstanz.de/~bittel/msi_robo/Vorlesung/05_Planung_Potentialfeldmethoden.pdf Stand: 12.05.2015

[rosa] *move_base*. – http://wiki.ros.org/move_base Stand: 04.10.2015

[rosb] *RoboCup Logistics League*. – http://www.ros.org/history/ Stand: 11.05.2015

[Sch14] Scheel, H.: *Konzeption und Implementierung eines reaktiven Pfadplanungsverfahrens für 3D-Umgebungen basierend auf dem Elastic Band Framework*, Fachhochschule Brandenburg University of Applied Sciences, Diplomarbeit, oct 2014

[SK08] Siciliano, B. ; Khatib, O.: *Springer Handbook of Robotics*. Springer-Verlag, 2008

[srl] *Global_Planner*. – https://github.com/srl-freiburg/srl_global_planner Stand: 22.10.2015

[US13] U. Schmucker, Prof. D. t.: *Vorlesung - Grundlagen mobiler Roboter*. 2013

[v-r] *v-rep*. – `http://www.coppeliarobotics.com/` Stand: 04.10.2015

[vor] *Voronoi_planner*. – `https://github.com/frontw/voronoi_planner` Stand: 22.10.2015

[you12] *KUKA youBot User Manual*. : *KUKA youBot User Manual*. Locomotec, 2012

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als den angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Diese Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Magdeburg, den 19. Januar 2016 B. Sc. Hauke Petersen