Universität Bremen

Fachbereich 3 - Mathematik und Informatik

Diplomarbeit

Entwicklung einer reaktiven Steuerung für mobile Roboter auf Basis der „*Nearness-Diagram*“-Methode für Navigation in Innenräumen

Philipp Kraetsch

Gutachter: Dr. Udo Frese, Prof. Dr. Christian Freksa

Betreuer: Dr. Udo Frese, Christian Mandel

November 2007

Abstract

Die vorliegende Diplomarbeit befasst sich mit der Entwicklung und der Implementation einer reaktiven, behavioristischen Steuerung für die autonome Navigation eines Rollstuhls. Hierbei ist als Grundlage die „*Nearness-Diagram*“-Methode verwendet worden, welche als situationsbasierte Verhaltensmethode grundlegend für holonome, kreisrunde Roboter entwickelt wurde.

Hauptbeitrag dieser Arbeit ist zum einen die implementative Umsetzung und Integration des Verfahrens in das System des Bremer autonomen Rollstuhls „*Rolland*“, zum anderen die Adaption der Methode an die spezielle Form eines Rollstuhls. Letztere Aufgabe schließt sowohl die Untersuchung und Analyse heikler Situationen in Bezug auf die statischen und dynamischen Teile der Umgebung ein, als auch der Entwurf und die Umsetzung einer Lösungsmethodik für genau jene Situationen.

Abschließende detaillierte Testläufe und ihre Dokumentation sollen die Arbeitsweise des erweiterten Grundverfahrens verifizieren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung und Motivation

Im Zentrum der vorliegenden Diplomarbeit steht die Entwicklung und Implementation einer verhaltensbasierten, reaktiven Steuerung für den autonomen Rollstuhl der Universität Bremen, *Rolland*.
Die Entwicklung eines solchen Steuerungsmoduls umfasst mehrere signifikante Bereiche, die nun eingehend erläutert und an entsprechender Stelle dieser Arbeit konkretisiert werden.

1.1 Zielsetzung

Das Ziel des entwickelten Systems liegt in der kollisionsfreien Navigation des Rollstuhles *Rolland* zum Personentransport in teilweise engen oder schwer passierbaren Innenräumen.
Die Umgebung umfasst Büroräume, Korridore und Flure, Räume jeglicher Größe mit Türen und Durchfahrten verschiedener Breite, Seminarräume mit Mobiliar und Innenausstattungen, teilweise enge und nicht leicht passierbare Durchgänge. In einer Grundannahme kann davon ausgegangen werden, dass das zu befahrende Gebäude rollstuhlgerecht ausgelegt ist. Dies soll heißen, dass die Türen, Durchgänge und Korridore eine Mindestbreite aufweisen, damit ein Rollstuhl sie passieren kann. In der Praxis sieht dies jedoch meist anders aus. Auch wenn das Gebäude architektonisch rollstuhlgerecht erbaut wurde, so können dennoch schwer zu passierende Stellen auftreten, beispielweise durch das Mobiliar (Tische, Stühle und anderes) oder aber auch durch andere Einrichtungsgegenstände wie Dekorationen (Blumentöpfe oder Feuerlöscher, die einen Korridor verengen oder ähnliches). Solche schmalen Bereiche, durch die der Rollstuhl physisch navigieren kann, stellen

eine besondere Herausforderung an die Navigation und müssen vom entwickelten Steuerungsmodul gemeistert werden.

Das System ist so entwickelt, dass in unbekannter Umgebung navigiert werden kann. Auch soll es möglich sein, in sich dynamisch verändernden Umgebungen sicher navigieren zu können. Dies umfasst beispielsweise das Zuschlagen einer Tür, auf die zugesteuert wird, wie auch das Öffnen einer Tür und somit das Erscheinen einer neuen Durchfahrtsmöglichkeit. Aber auch der umgebende Personenverkehr kann die Umgebung und somit die Steuerung des Rollstuhls stark beeinflussen. Das System muss nicht nur Sicherheit für den Fahrer bieten, sondern auch für die umgebenden Personen.

Die Kollisionsfreie Navigation ist somit Hauptbestandteil dieser Diplomarbeit und hängt zusammen mit der direkten Steuerung. Eine Kollision mit einem Objekt, ganz gleich ob Person, Mobiliar oder Wand, darf in keinem Falle stattfinden und muss unter allen Umständen vermieden werden. Dies ist Aufgabe des entwickelten Systems.

In diesen Bereich fallen spezielle Anforderungen an die Steuerung, die sich auch durch die spezifische Form des Rollstuhls ergeben und die damit folgenden Probleme in der Navigation durch enge Umgebungen. Eine Türeinfahrt oder ein enger Durchgang stellt beispielsweise eine besondere Anforderung an den Navigationsmechanismus eines Rollstuhlsystems, da in den meisten Fällen eine explizite Ausschwenkbewegung durchgeführt werden muss, um besonders schmale Türen zu passieren. Eine möglichst natürliche und intuitive Steuerung ist angestrebt.

Das Verfahren zur kollisionsfreien Navigation wird auf Basis der „*Nearness-Diagram*“-Methode realisiert, welche bereits auf Robotern an der Universität in Zaragoza, Spanien, erfolgreich angewendet wird. Dieses Verfahren ist grundlegend für runde Roboter entwickelt worden, welche sich holonom bewegen können, und nur durch weitere externe Module an nicht holonome Roboter spezieller Form, wie einen autonom navigierenden Rollstuhl, adaptierbar.

Ziel dieser Arbeit ist es, durch direkte Änderungen und Anpassungen des Grundverfahrens eine kollisionsfreie Navigation des Bremer autonomen Rollstuhls *Rolland* zu ermöglichen. Da es sich bei all den Anforderungen stets um den Personentransport handelt, muss auch

hier besondere Aufmerksamkeit auf **die Komfortabilität** gelegt werden. Anders als bei Objekten, die transportiert werden, muss - besonders bei körperlich eingeschränkten Personen - große Sorgfalt auf ein möglichst angenehmes und störfreies Fahren gelegt werden; weiche und oszillierfreie Bewegungen sind angestrebt und müssen realisiert werden.

1.2 Struktur der Arbeit

Das folgende Kapitel 2 beschäftigt sich mit der Darlegung der Grundlagen, auf denen diese Diplomarbeit basiert und gliedert sich in drei Teilbereiche. Einleitend werden die Grundlagen autonomer Roboternavigation und kollisionsvermeidender Verfahren erläutert und erklärt. Dies schließt eine Betrachtung des derzeitigen Standes der Entwicklung ein, in dessen Zuge mehrere alternative Methoden zur Hindernisvermeidung grundlegend betrachtet und diskutiert werden. Anschließend erfolgt eine Einführung in die Navigation mit Hilfe der „*Nearness-Diagram*“-Methode und es werden an dieser Stelle die grundlegenden Strategien zur kollisionsfreien Steuerung auf Basis einer speziellen Entität, dem „*Nearness-Diagram*“, dargelegt. Der letzte Abschnitt des zweiten Kapitels befasst sich mit der Betrachtung des Entwicklungsstandes im Bereich der autonomen Rollstuhlnavigation. Hier werden zum einen laufende Systeme vorgestellt und in Zusammenhang gebracht, zum anderen werden die Grundlagen der Bewegung eines differentiell angetriebenen Rollstuhls erläutert.

In Kapitel 3 wird das entwickelte Navigationsmodul betrachtet und erklärt. Besonderes Augenmerk wird hier auf die Erweiterungen gegenüber der originalen „*Nearness-Diagram*“-Steuerung gelegt. Da diese Methode in ihrer Grundfassung lediglich für kreisförmige Roboter ohne Bewegungseinschränkungen entwickelt wurde, waren Änderungen und Erweiterungen notwendig, damit die Steuerung eines autonomen Rollstuhls möglich wurde.

Im anschließenden Abschnitt (Kapitel 4) werden durchgeführte Testläufe und Testfahrten

mit dem entwickelten System beschrieben und analysiert. Besonders schwierige und/oder komplexe Anforderungen an die Steuerungen werden betrachtet und die Durchführung dieser Tests referiert.

Eine Diskussion des Navigationsmoduls erfolgt im abschließenden Kapitel 5, wobei die Ergebnisse der Testfahrten und -läufe betrachtet werden und in Zusammenhang mit den eingehend festgelegten Anforderungen an das System gebracht werden. Hier werden auch die Grenzen eines lokalen verhaltensbasierten Navigationssystems aufgezeigt und Situationen betrachtet, welche für eine Steuerung mit Hilfe der „*Nearness-Diagram*“-Methode ungeeignet beziehungsweise sogar nicht lösbar sind.
Abschließend wird ein Blick auf mögliche zukünftige Entwicklungen an diesem System geworfen.

1.3 Beitrag der Arbeit

Der Beitrag dieser Diplomarbeit liegt in der Implementation und der anschließenden Änderung des Originalverfahrens der „*Nearness-Diagram*“-Navigation, um die komplexe Steuerung des Rollstuhls *Rolland* der Universität Bremen zu ermöglichen.
Zum einen ist der Satz an Situationen der verhaltensbasierten Steuerung erweitert worden, um differenzierter auf die Umgebung reagieren zu können als es in der grundlegenden Entwicklung der Fall war und ein vorausschauenderes Fahren zu ermöglichen. Zum anderen musste eine spezielle Ausschwenkbewegung in das Verfahren eingebettet werden, die für kreisrunde Roboter nicht vorgesehen ist, ohne die ein Rollstuhl aber nicht in enge oder schmale Bereiche einlenken kann. Auch in diesem Zusammenhang wurden neue Kriterien und Situationen in den Entscheidungsbaum der „*Nearness-Diagram*“-Navigation eingefügt, um das Ausschwenkverhalten durchführen zu können.
Abschließende ausführliche Testläufe der erweiterten Mechanik mit besonderem Augenmerk auf enge und schmale Umgebungen stellen einen weiteren Bestandteil der Arbeit dar.

2 Theoretische und praktische Grundlagen

In diesem Abschnitt werden die theoretischen und praktischen Grundlagen, die dem entwickelten System inhärent sind, aufgezeigt, erläutert und dargelegt. Hierbei wird auch auf die Zusammenhänge zu existierenden Methoden und Verfahren eingegangen.
Es handelt sich grundlegend um ein „sensorbasiertes Navigationssystem“, welches auf verhaltensbasierten Methoden aufbaut. Auf diese Punkte soll im Folgenden detailliert eingegangen werden.

2.1 Grundlagen reaktiver verhaltensbasierter Methoden

Die Entwicklung von Steuerungsmechanismen für autonome Roboter ist seit Jahren ein weit erforschtes Gebiet. So gibt es in der heutigen Zeit viele Bereiche, in denen Roboter den Menschen bei seinen Tätigkeiten unterstützen. Einige dieser Bereiche erfordern das selbstständige Handeln eines Roboters in einer ihm mehr oder weniger bekannten Umgebung.
Die autonome Steuerung eines Roboters in seiner Umwelt ist dabei keine triviale Angelegenheit. So gibt es bis heute noch nicht **das** Verfahren zur Roboternavigation. Dies liegt unter anderem daran, dass sowohl die Einsatzgebiete als auch die Architekturen der Roboter stark variieren.
Die Gemeinsamkeit aller Methoden liegt in der Zielsetzung, einen Roboter von einem Startpunkt zu einem Ziel zu navigieren. Hierbei ist die genaue Berechnung eines sicheren und kollisionsfreien Bewegungsbefehles ein Hauptbestandteil von autonom navigierenden Systemen und somit Gegenstand dieser Arbeit.

Nach [21] lassen sich Navigationssysteme, welche kollisionsfreie Navigation durchsetzen, auf folgende Weise kategorisieren:

Modellbasierte Systeme verwenden ein spezifisches Weltmodell ihrer Umgebung, beispielsweise eine Karte, um dann aus diesem den direkten Bewegungsbefehl abzuleiten. Dadurch können diese *globalen* Algorithmen in einem Schritt den vollständigen Weg vom Start- zum Zielpunkt berechnen.

Reaktive Systeme arbeiten innerhalb eines „Wahrnehmen-Handeln"-Zyklus [1], wobei die von Sensoren gelieferten Informationen der Umgebung als „Wahrnehmung" und die direkte Motorsteuerung durch Bewegungsbefehle als „Handlung" interpretiert werden. Da sie nur einen Teil ihrer Umgebung aktuell wahrnehmen, haben sie *lokalen* Charakter und müssen schrittweise navigieren.

Hybride Systeme bilden den Zusammenschluss dieser beiden Verfahren. So arbeiten in diesen Systemen sowohl ein modellbasiertes System, als auch ein reaktives System miteinander, um die kollisionsfreie Navigation zu erreichen.

Globale Methoden sind meistens so genannte Bahnplaner. Der Pfad von der aktuellen Position zur Zielposition wird auf Basis der vollständig zur Verfügung stehenden Informationen der Umgebung geplant. Im Gegensatz dazu werden mit Hilfe von lokalen Algorithmen häufig reaktive Verhaltensmethoden realisiert. Es wird hierbei lediglich auf die aktuelle Sicht des Roboters „reagiert".
Globale Algorithmen (modellbasierte Systeme) greifen dabei auf ein Weltmodell - basierend auf a priori Informationen - zurück. Dies können gespeicherte Karten sein oder anderweitig abgelegte Daten der Umgebung. Lokale Methoden (reaktive Systeme) arbeiten lediglich mit einem Ausschnitt der Umwelt, der sie direkt umgibt. Die Informationen sind also begrenzt auf die aktuellen Sensordaten. Ein wichtiger Unterschied dieser beiden Formen liegt im rechnerischen Aufwand. Für globale Methoden wird ein sehr viel höherer Speicher- und Rechenaufwand verzeichnet als es für lokale Methoden der Fall ist.

Dafür werden bei ersteren aber auch Informationen verarbeitet, die sich nicht im aktuellen Sichtbereich des zu navigierenden Roboters befinden. Dieser Aspekt ist ausschlaggebend für die Wahl der Modellierungsform. In Echtzeitnavigationen ist es zumeist nicht möglich, mit Hilfe globaler Methoden, zeitgerecht auf dynamische Änderungen zu reagieren. In diesem Falle wird auf reaktive Verfahren zurückgegriffen.

Der Gegenstand dieser Diplomarbeit ist ein reaktives Navigationssystem, weshalb auf diesen Aspekt nun näher eingegangen wird. Der Grund für diese Auswahl liegt in der obigen Beschreibung in Zusammenhang mit der eingehend formulierten Zielsetzung. So wird das entwickelte System hauptsächlich in dynamischen und komplexen Umgebungen zum Einsatz kommen. Zeitkritische Reaktionen sind nötig, um auf Veränderungen reagieren zu können. Mit dieser Festlegung ist ein reaktives Verfahren den modellbasierten Systemen vorzuziehen [21].

2.1.1 Reaktive Navigationssysteme

Reaktiven Methoden zur Navigation liegt eine gemeinsame Struktur zugrunde. Diese Grundlage ist der Zyklus, der aus den Teilen „Wahrnehmung“, „Planung“ und „Handlung“ besteht ([1], siehe Abbildung 2.1). In einem reaktiven Navigationssystem wird diese Reihenfolge so lange wiederholt, bis das vorher definierte Ziel erreicht wurde.

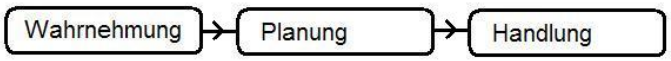

Abbildung 2.1: Zyklus reaktiver Methoden

Abbildung 2.2 konkretisiert diesen Ansatz nach [1]. So wird der Zyklus um einige Module erweitert, der Grundgedanke bleibt jedoch bestehen.

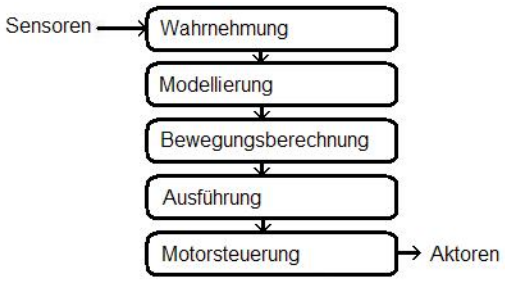

Abbildung 2.2: Erweiterter Zyklus reaktiver Methoden

In der Phase der **Wahrnehmung** werden die Daten, die die Sensoren des Roboters liefern, derart aufbereitet, dass sie im Folgenden vom System verwendet werden können. Anschließend wird mit Hilfe dieser Daten ein **Weltmodell kreiert** oder ein existierendes Weltmodell aktualisiert. Die Entscheidungen zur Bewegungssteuerung, welche im darauf folgenden **Berechnungsschritt** getroffen werden, basieren hierbei zum einen auf dem zuvor erstellten Weltmodell, welches die Umgebung darstellt, zum anderen auf einer Berechnungslogik, dem eigentlichen kollisionsvermeidenden Navigationsverfahren.
Die letzten beiden Module regeln die direkte **Bewegung des Roboters** und ermöglichen so eine Navigation.
Dieser Zyklus aus „Wahrnehmen-Planen-Handeln“ wird nun wiederum so lange durchlaufen, bis die Zielposition erreicht wurde.

Nach [18] lassen sich Algorithmen zur Kollisionsvermeidung nach der Art der Informationsanalyse und -verarbeitung gruppieren.
So gibt es Funktionen, basierend auf

- der Anwendung mathematischer Funkionen auf die Sensordaten, um als Ergebnis eine Bewegungsvorschrift zu erhalten. In diesen Bereich fallen unter anderem die Potenzialfeld-Methoden (siehe Abschnitt 2.2.1).
- einem begrenzten Satz an Bewegungsbefehlen, aus denen nach bestimmten Kriterien der „Beste“, Optimale ausgewählt wird. Ein Beispiel für solch einen Algorithmus ist der „*Dynamic Window Approach*“ (siehe Abschnitt 2.2.4).

- dem Extrahieren einer oder mehrerer höherwertiger Informationen aus den Sensordaten, um daraus im Nachhinein den gewünschten Bewegungsbefehl zu berechnen. Die „*Nearness-Diagram*“-Navigation (siehe Abschnitt 2.3) fällt in diese Gruppe.

2.1.2 Situationsbasierte Ansätze

Situationsbasierte Ansätze sind einer von drei grundsätzlichen Strategien zur Entwicklung verhaltensbasierter Methoden. Sie sind neben den ethologisch orientierten und den experimentellen Ansätzen die meist verbreiteten [1].

Ethologische Ansätze basieren auf der Untersuchung des Verhaltens anderer Lebewesen, beispielsweise dem Verhalten von Tieren in bestimmten Bereichen. So wird zur Entwicklung eines solchen Verhaltens auch immer eine biologische Grundlage verwendet.

Experimentelle Verfahrensweisen verbessern mit schrittweisen Änderungen ein implementiertes Startverhalten. In der Robotik bedeutet dies, dass ein Roboter mit einem minimalen System erstellt wird, welches nur über eine begrenzte Anzahl an Fähigkeiten verfügt. Dieses System wird anschließend in der realen Umgebung evaluiert und auf unzureichendes Verhalten untersucht. Diese Unzulänglichkeiten im System werden dann in einem zweiten Schritt verbessert, das dann schließlich neuen Untersuchungen unterzogen wird und daraufhin weiter verbessert wird. Dieser Zyklus aus Evaluation und Überarbeitung wird so lange vollzogen, bis die eingehend definierten Ziele ausreichend erreicht sind.

Situationsbasierte Verhaltensmethoden basieren auf einem definierten Satz an Situationen, in denen sich der Roboter befinden kann und auf welche dann schließlich ein bestimmtes Verhalten folgt. So erfolgt in jedem Ausführungszyklus zum einen die Erkennung der jeweiligen Situation (zumeist basierend auf der Wahrnehmung des Systems, wie Sensoren), zum anderen die Auswahl einer bestimmten Aktion, die dieser Situation entspricht. Eine Änderung der Umgebung zieht also eine Änderung der Situation und somit auch ein anderes Verhalten mit sich.

Situationsbasierte Verhalten sind nach [1] nur begrenzt einsetzbar, da es in diesem Ansatz nötig ist, mit einem möglichst kompletten Satz an Situationen alle verschiedenen Umgebungsvariationen abzudecken. Eine stark dynamische Umwelt, mit vielen unvorher-

sehbaren Zuständen, ist somit unter Umständen nur mit einem sehr großen Satz an eingehend definierten Situationen zu meistern. Eine hohe Anzahl an Situationen erfordert aber wiederum eine große Menge an verschiedenen auszuführenden Aktionen, wodurch situationsbasierte Verhaltensmethoden schnell umfangreich und entsprechend unübersichtlich werden können. Es sollte also das Ziel sein, mit einer geringen Anzahl an Situationen viele Umgebungsvariationen abzudecken. Dies setzt ein genaues Verständnis der Abbildung der Umwelt innerhalb des Systems voraus.

2.2 Kollisionsvermeidende Navigationsverfahren

Im folgenden Abschnitt werden mehrere Verfahren zur Kollisionsvermeidung grundlegend erläutert. Dies dient dazu, einen Überblick über die existierenden und angewandten Navigationsverfahren zu geben und die Grundlagen jeder dieser Methoden zu betrachten. Auch die den Methoden inhärenten Grenzen in Bezug auf ihre Arbeitsweise und die Ansprüche an die Umgebungen finden hier Erwähnung.

Hauptsächlich werden im Zusammenhang mit kollisionsvermeidenden Verfahren fünf verbreitete Methoden genannt ([21],[18]). Dies sind die Potenzialfeld-Methoden, das Vektorfeld-Histogramm, die „*Elastic Band*“-Methode, der „*Dynamic Window Approach*“ und schließlich der Gegenstand dieser Diplomarbeit, die „*Nearness-Diagram*“-Methode.
Diese Auswahl an Verfahren deckt die jeweiligen Gruppen der oben beschriebenen Kategorisierung ab (siehe Abschnitt 2.1.1).

2.2.1 Potenzialfeld-Methoden

Hinter allen Verfahren, die Potenzialfeld-Methoden (PFM) als Basis zur Kollisionsvermeidung verwenden, steht eine grundlegende Idee, die mathematisch gefasst wird.
Prinzipiell wird der autonom zu steuernde Roboter zu jeder Zeit bestimmten imaginären „Kräften“ ausgesetzt, die sowohl anziehend als auch abstoßend sein können ([9] und [10]).

So wirken lokale Hindernisse abstoßend auf den Roboter, während das Ziel eine global anziehende Kraft ausübt. Eine Summe aller gleichzeitig wirkenden Kräfte ergibt die Grundlage für die Berechnung des Bewegungsbefehles, um den Roboter zum Ziel zu führen.
Die Kalkulation der wirkenden Kräfte geschieht auf Basis der Ziel- und der Sensordaten, wobei jedes Hindernis mit unterschiedlich starkem Potenzial auf den Roboter wirkt und die Zielposition als ein „anziehender Pol“ dargestellt ist.
Ein Potenzialfeld modelliert die lokale Umgebung des Roboters, in der die Wirkung der einzelnen Kräfte für jeden Bereich (zumeist als Zellen dargestellt) mit Hilfe von Potenzialfunktionen berechnet wird (siehe Abbildung 2.3). Eine Steuerung des Roboters ist mit Hilfe dieses Kraftfeldes möglich, er wird durch die global wirkende anziehende Kraft des Zieles in dessen Richtung „gezogen“ und auf dem Weg dorthin durch die lokal wirkenden Kräfte der Hindernisse von diesen abgestoßen. Man vergleicht dieses Pfadverhalten gerne mit dem Fließen von Wasser in einem Gebirge.

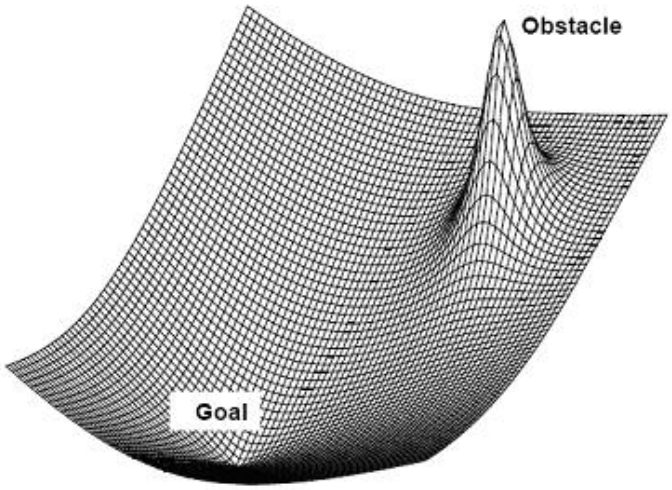

Abbildung 2.3: Beispiel eines Potenzialfeldes [9]

In verschiedenen Erweiterungen dieser Grundidee lassen sich auch komplexere Eigenschaften verwirklichen. So kann durch Änderungen der Kraftwirkungen auch die aktuelle Geschwindigkeit und Trajektion des Roboters in die Berechnung einfließen.

Der Vorteil dieser Methode liegt darin, dass sie einfach und elegant zu implementieren und umzusetzen ist. So lassen sich einfache Anwendungen mit Potenzialfeld-Methoden innerhalb kurzer Zeit verwirklichen [10].

Der mit Abstand größte Nachteil der Potenzialfeld-Methode ist die Gefahr von lokalen Minima, die zu Situationen führen, aus denen der Roboter nur mit höher geschalteten Navigationsmechanismen entweichen kann. Ein Beispiel hierfür stellt sich in so genannten U-Form-Hindernissen (siehe Abbildung 2.4). Weiterhin ist eine Navigation in besonders enge Bereiche hinein problematisch, da die abstoßenden Kräfte der Hindernisse zu beiden Seiten des Roboters ein Hereinfahren in diesen Bereich unter Umständen verhindern [10].

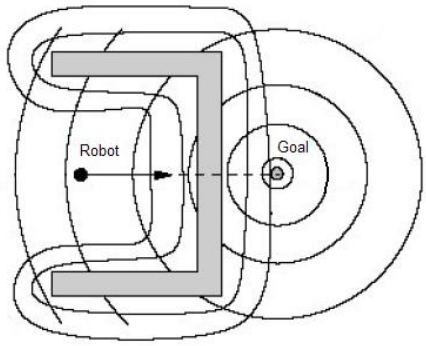

Abbildung 2.4: PFM und U-Form-Hindernisse [27]

Ein weiterer Nachteil besteht in starken Oszillationen in besonders unstrukturierten Bereichen mit einer hohen Anzahl an Hindernissen, die durch ihre dem Potenzialfeld innewohnenden Kräfte starken Einfluss auf die Bewegung des Roboters haben.

2.2.2 Vektorfeld-Histogramm

Das Vektorfeld-Histogramm (VFH) ist eine weitere Methode zur Hindernisvermeidung in Echtzeitanwendungen auf mobilen Robotern. Grundlage dieser Methode ist ein zweidi-

mensionales Histogramm-Gitter, welches ständig durch die aktuellen Sensordaten erneuert wird. Dieses Gitter wird dazu verwendet, eine interne Weltdarstellung der Umgebung zu modellieren, indem die Zellen des Gitters mit den Hindernisinformationen der Sensoren gefüllt werden [3]. Ein hoher Eintrag im Histogramm-Gitter entspricht dem Vorhandensein eines Hindernisses an dieser Stelle der realen Umgebung, ein niedriger (beziehungsweise kein) Eintrag, weist auf eine freie Fläche hin.

Jene Modellierung wird im weiteren Verlauf dazu verwendet, entsprechende Bewegungsbefehle für die Navigation zu berechnen. Dies geschieht in zwei hauptsächlichen Schritten: Im ersten Schritt werden die Einträge aus dem Histogramm-Gitter dazu verwendet, ein Polarhistogramm um die Position des Roboters herum zu erstellen. Auf diese Weise wird die Umgebung des Roboters in einzelne Sektoren unterteilt, die den Einträgen in diesem Polarhistogramm entsprechen (siehe Abbildung 2.5 oben). Hierfür werden die Histogramm-Gitter-Einträge als Hindernisvektoren betrachtet, die durch die Ausrichtung des Roboters eine bestimmte Richtung zu ihm aufweisen und somit in die Berechnung des Polarhistogramms einfließen können.

In einem zweiten Schritt wird das zuvor berechnete Polarhistogramm dazu verwendet, die direkten Bewegungsbefehle für den Roboter zu ermitteln. Hierfür wird aus den Einträgen des Histogramms, das implizit die Hindernisaufteilung der Umgebung enthält, eine weitere Information abgeleitet: ein so genanntes „Tal“ im Histogramm (mehrere niedrige oder keine Einträge nebeneinander) deutet darauf hin, dass sich in dieser Sektion der Umgebung keine Hindernisse befinden (siehe Abbildung 2.5 unten) und es sich hierbei um eine befahrbare Sektion handelt.
Schließlich wird mit dieser Information der Bewegungsbefehl ermittelt, um vorbei an etwaigen Hindernissen auf das Ziel zuzusteuern [3].

Eine Weiterentwicklung dieses Konzeptes stellt das VFH+ dar. Durch die Aufnahme der Größe und Trajektion des zu navigierenden Roboters in die Berechungen konnten die in

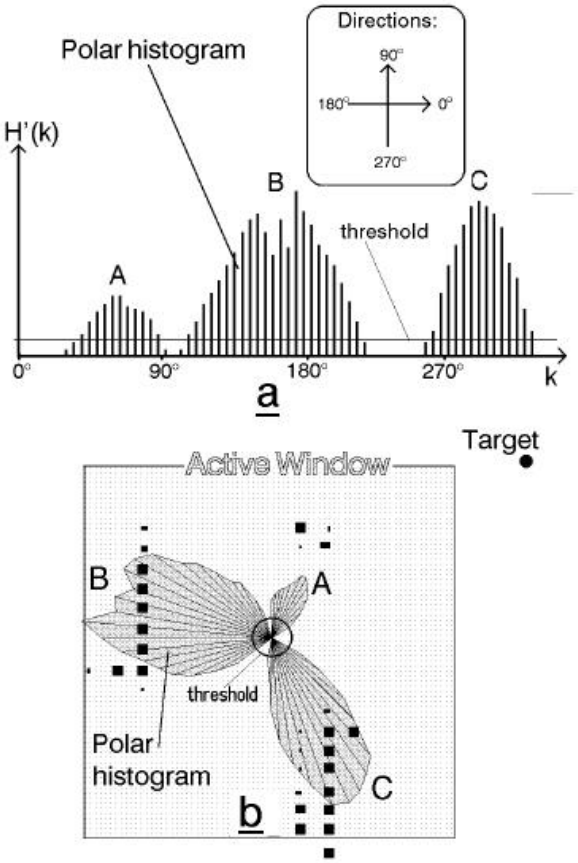

Abbildung 2.5: Beispiel eines Vektorfeld-Histogramms [3]

der originalen VFH-Implementation auftretenden starken Oszillationen reduziert werden [21].

Die Einführung eines vorausschauenden Algorithmus, der den ausgewählten Bewegungsbefehl auf seine Korrektheit und Machbarkeit hin verifiziert, führte zur Entwicklung des VFH*. Hierdurch war es möglich, Befehle auszuschließen, die in Sackgassen oder anderen „Trap"-Situationen endeten.

2.2.3 „Elastic Band"

Als nächste Methode zur kollisionsfreien Navigation soll nun die „*Elastic Band*"-Steuerung (EB) Erwähnung finden. Grundlegend soll mit Hilfe dieses Verfahrens die Lücke zwischen

einem globalen Pfadplaner und der lokalen Hindernisvermeidung geschlossen werden [29].

Ein Pfadplaner berechnet mit Hilfe eines Umgebungsmodells (Karten oder ähnlichem) einen kollisionsfreien Weg zur Zielposition. Dies ist die grundlegende Form des „*Elastic Band*“. Dieser Pfad wird anschließend während der Fahrt in Echtzeit mit den von den Sensoren wahrgenommenen Objekten „verformt“, um eine Kollision mit solchen Hindernissen zu vermeiden, die im globalen Modell der Umwelt nicht vorhanden sind. Dies sind beispielsweise Gegenstände, die nicht im Kartenmaterial verzeichnet sind, oder Personen, die den berechneten Pfad des Planers kreuzen.

Abbildung 2.6 zeigt die Arbeitsweise einer Steuerung mit Hilfe der „Elastic Band“-Methode.

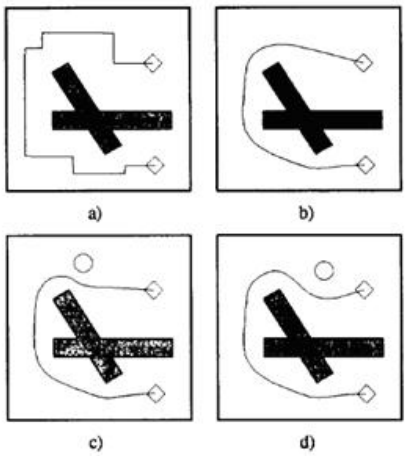

Abbildung 2.6: Beispiel eines „Elastic Band“ [29]

Abbildung 2.6 a) bezeichnet den Pfad, den ein Pfadplaner berechnet. In einem zweiten Schritt b) wird der Weg von starken „Knicken“ befreit, um für eine natürliche und glatte Bewegung zu sorgen. Gleichzeitig wird der Weg in diesem Schritt von unnötig verlängernden Abschnitten befreit.
Bei weiteren in späteren Sensoraufnahmen sichtbaren Hindernissen, welche ungeplant die nähere Umgebung des Roboters betreten, verformt sich der Pfad dementsprechend um

die kollisionsfreie Navigation zu gewährleisten (Abbildung 2.6 c und d). Jene Verformung wird mit Hilfe von so genannten „bubbles“ (Blasen) realisiert, welche über den kompletten Pfad gelegt werden und somit zu jedem Zeitpunkt der Fahrt die lokale Umgebung des Roboters frei von Hindernissen halten. Entscheidend hierbei ist die Größe jener „bubbles“ (siehe Abbildung 2.7). Eine nähere Erläuterung der Berechnungsweise steht nicht direkt im Zusammenhang mit dieser Arbeit und soll nicht weiter betrachtet werden (weiterführend siehe [29]).

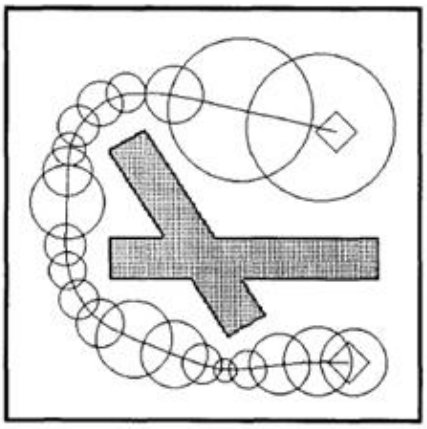

Abbildung 2.7: „Elastic Band“ mit lokalen „bubbles“ [29]

Für die Verwendung der „*Elastic Band*“-Methode ist sowohl ein globaler Pfadplaner vonnöten, als auch die Existenz eines vollständigen durchgängigen Pfades von der Roboterposition zur Zielposition. Eine komplette bekannte Karte der Umgebung muss vorhanden sein. Durch unbekanntes Gelände und sich stark dynamisch verändernde Umgebungen ist nur schwer auf diese Weise zu navigieren, da hier das durchgehende Band ausgeprägten Verformungen unterliegt. Das Schließen einer Tür würde beispielsweise unter Umständen Deformierung des „*Elasic Band*“ unmöglich machen - das Band würde „reißen“. In diesem Falle müsste wiederum eine höher geschaltete Instanz (ein globaler Pfadplaner) einen neuen Weg berechnen [29].

2.2.4 Dynamic Window Approach

Eines der am meist verbreiteten Verfahren zur kollisionsfreien Navigation stellt der Ansatz des „*Dynamic Window Approach*“ (DWA) dar. Im Gegensatz zu vielen anderen Methoden leitet sich der DWA direkt aus der Bewegungsdynamik des Roboters ab.

So erfolgt die Berechnung des Bewegungsbefehles, der eine Steuerung ermöglichen soll, auf Grundlage der jeweils möglichen Bewegungen zu jedem Zeitpunkt der Anwendung. Die Suche nach dem passenden Geschwindigkeits- und Lenkbefehl wird auf einen bestimmten Bereich beschränkt, das „Fenster“ der in einem kurzen Intervall (bis zum folgenden Berechnungsschritt in der Steuerungsschleife) erreichbaren Geschwindigkeiten ([5] und [4]).
Die Suche nach einem korrekten und sicheren Bewegungsbefehl geschieht im zweidimensionalen Suchraum, wobei sowohl nach der Geschwindigkeit (v) als auch der Lenkgeschwindigkeit (w) gesucht wird. Das Ergebnis des Algorithmus ist somit das Tupel (v,w).
Eine Beschränkung des Suchraumes nach möglichst optimalen Bewegungsbefehlen schränkt somit auch die Komplexität des Verfahrens ein und reduziert den rechnerischen Aufwand um ein Vielfaches.

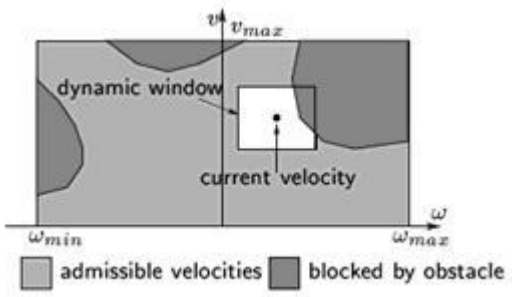

Abbildung 2.8: Beispiel DWA [4]

Um nun eine kollisionsfreie Navigation zu ermöglichen, wird innerhalb des „*Dynamic Window*“ (siehe Abbildung 2.8) nach solch einem Geschwindigkeitstupel (v,w) gesucht, das den Roboter sowohl in die Richtung des Zieles navigiert, als auch eine Kollision mit einem Hin-

dernis vermeidet. Die Größe des Fensters hängt hierbei fest zusammen mit den internen Parametern des Roboters und den Beschränkungen, welche sich durch dessen Architektur ergeben. Die Auswahl des letztendlichen Bewegungsbefehles geschieht also nur innerhalb des „*Dynamic Window*“ unter Verwendung einer Funktion, die die Länge der Laufbahn zum nächsten Hindernis maximiert und die Ausrichtung des Roboters zum Ziel dabei in die Berechnung einbezieht [4].

2.2.5 „Nearness-Diagram“-Methode

Die „*Nearness-Diagram*“-Methode ist die Grundlage der vorliegenden Arbeit, deshalb ist eine sehr viel genauere Betrachtung dieses Verfahrens nötig. Eine detaillierte Beschreibung des Algorithmus liefert der folgende Abschnitt.

2.3 Grundlagen der „Nearness-Diagram“-Navigation

Die Methode der „*Nearness-Diagram*“-Navigation ist ein solches oben beschriebenes reaktives Navigationsverfahren. Es beruht auf der Situationseinschätzung (basierend auf Sensordaten) und der dazu entsprechenden Auswahl der Handlung (Aktion).
In diesem Abschnitt sollen die Grundlagen der Implementierung nach [16], [21], [17] und [18] erläutert werden.

Das im Folgenden beschriebene Verfahren ist grundlegend für kreisrunde Roboter entwickelt worden. Eine Adaption auf andere Formen von Robotern (rechteckig, quadratisch, oval) ist nicht ohne Weiteres möglich. Mit speziell diesem Thema, der Anpassung an einen Rollstuhl, befasst sich Kapitel 3.

2.3.1 Grundlegende Situationen

Für die verhaltensbasierte Navigation wird als Basis ein Satz an Situationen benötigt. Jede mögliche Kombination von Umgebung und Systemstatus muss eindeutig auf eine Si-

tuation abgebildet werden können. Nur so kann in jedem Berechnungszyklus das System feststellen, in welchem Zustand es sich befindet und wie es entsprechend zu reagieren hat.

Um diese Situationen bestimmen zu können, ist es vonnöten, den Roboter in Verbindung mit den umgebenden und vom Sensor wahrgenommenen Hindernissen zu bringen, sie in Relation zu stellen. Diese Relation basiert intuitiverweise auf der Entfernung und Lage von Hindernissen zum Roboter.
Notwendig für die Bestimmung der Situation ist also das vollständige Vorhandensein folgender Informationen:

- die Position des Roboters
- die Zielposition relativ zum Roboter
- die Sensorinformationen, die die Umgebung widerspiegeln.

Aus diesen Informationen wird anschließend eine definitive Situation berechnet.

Es folgt eine Definition der möglichen Situationen, die eine kollisionsfreie Navigation ermöglichen sollen. Der Satz dieser Situationen ist bewusst gering gehalten. Hierdurch wird eine „Explosion“ der Möglichkeiten und somit auch ein hoher Rechenaufwand und eine unübersichtliche Zustandslogik verhindert. Wenige definierte Situationen, die alle möglichen reellen Beschaffenheiten der Umgebung abdecken, sind angestrebt [1].

Die Bestimmung der jeweiligen Situationen erfolgt durch einen Entscheidungsbaum. In jedem Berechnungszyklus werden die Sensordaten neu ausgewertet und der entsprechende Zustand bestimmt. In dieser Auswertung wird der Roboter nicht nur in Relation zu den umgebenden Hindernissen gebracht, sondern auch die Lage des Zielpunktes fließt in die Berechnung ein.
Auf diese Weise wird aus der Umgebung ein besonderes Element gefiltert: die *free walking area*, ein Bereich zwischen möglichen Hindernissen, in der ein Navigieren möglich ist. Dieser Bereich wird durch zwei Seiten aufgespannt, die an Hindernisse grenzen. Dabei

wird unterschieden zwischen der Seite, die dem Zielpunkt am nächsten liegt (im Folgenden abkürzend „Zielseite“ genannt), und der gegenüber liegenden Seite („Seite 2“ in Abb. 2.9).

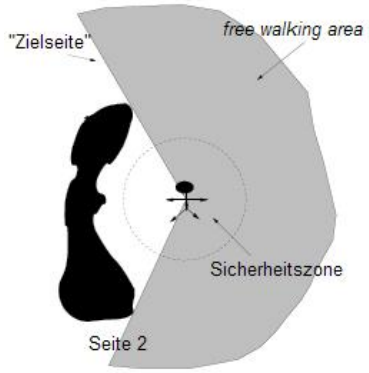

Abbildung 2.9: Beispiel einer *free walking area*, angelehnt an [13]

Die Bestimmung der *free walking area* wird später erläutert, lediglich das Vorhandensein eines solchen Bereiches ist Voraussetzung für die folgende Navigation.

Der somit entstehende Entscheidungsbaum ist wie folgt aufgebaut. Eine Beschreibung der einzelnen binären Entscheidungen wird im Anschluss an das Diagramm gegeben.

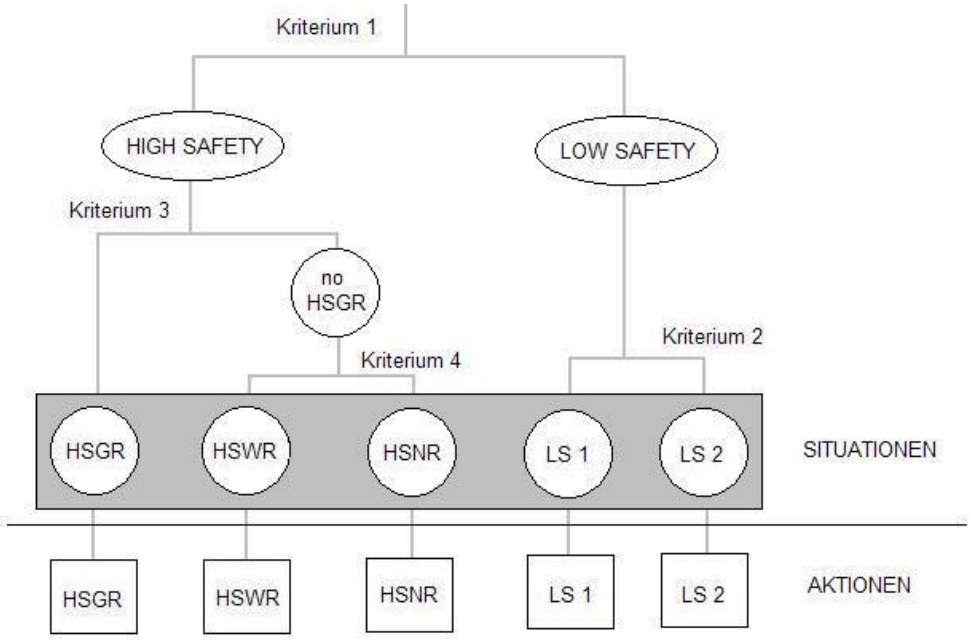

Abbildung 2.10: ND-Entscheidungsbaum

Kriterium 1: Sicherheit der Umgebung

Das erste und sicherlich wichtigste Kriterium ist die Frage der Sicherheit.
Das System unterscheidet hier zwischen den beiden Zuständen „hohe Sicherheit“ (*High Safety*) und „niedrige Sicherheit“ (*Low Safety*). Der Zustand der niedrigen Sicherheit ist immer dann gegeben, wenn sich ein Hindernis in der vordefinierten Sicherheitszone befindet. Hierzu wird als interner Parameter die Definition einer Sicherheitsdistanz benötigt. Diese Sicherheitsdistanz umschließt den Rollstuhl mit einer Sicherheitszone. In Abbildung 2.9 ist diese Zone um den kreisrunden Roboter aufgespannt und deutlich zu erkennen. Ein Hindernis in dieser Zone bedeutet ein Sicherheitsrisiko für den Roboter und es ist besondere Vorsicht geboten.
Ist die Sicherheitszone um den Rollstuhl frei von Hindernissen, ist eine Kollisionsgefahr nicht imminent. In diesem Fall braucht vorerst keine besondere Vorsicht vor behindernden Objekten genommen werden.

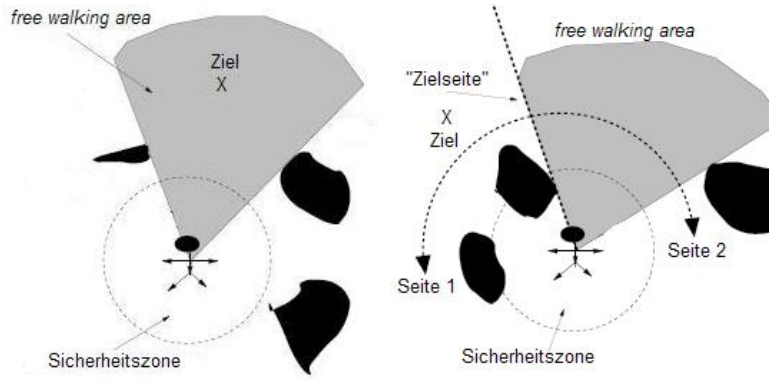

Abbildung 2.11: Links: HS - High Safety, Rechts: LS - Low Safety, angelehnt an [13]

In der beispielhaften Darstellung 2.11 lässt sich auf der linken Seite erkennen, dass hier kein Hindernis in der Sicherheitszone befindlich ist. Nicht jedoch in der rechten Abbildung, hier sind an zwei Stellen Hindernisse innerhalb dieser Zone. In den Situationen der „hohen Sicherheit“ ist das Ziel der Aktion, auf das Ziel zuzusteuern, während in Situationen der „niedrigen Sicherheit“ primär dem Hindernis ausgewichen wird.

Kriterium 2: Hindernislage in der Sicherheitszone

Das zweite Kriterium wird immer dann ausgewertet, wenn sich das System im Zustand der niedrigen Sicherheit befindet. Das bedeutet, es befinden sich ein oder mehrere Hindernisse innerhalb der Sicherheitszone und eine Kollision ist nicht ohne Weiteres auszuschließen. Eine Auswertung der Lage der Hindernisse ergibt sich durch die Überprüfung von Kriterium 2, der Lage der Hindernisse.

1. Low Safety 1: Diese Situation tritt immer dann auf, wenn sich ein Hindernis innerhalb der Sicherheitszone befindet und nur auf einer Seite der „Zielseite“ (Grenze der *free walking area*, die der Zielposition am nächsten ist) liegt. Die folgende Abbildung verdeutlicht diesen Sachverhalt:

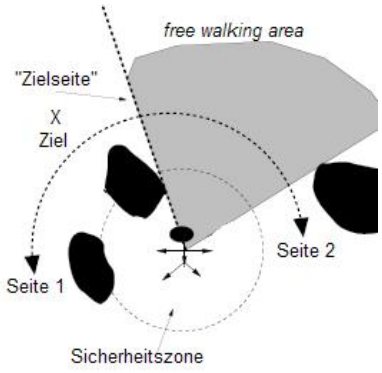

Abbildung 2.12: LS 1 - Low Safety 1, angelehnt an [13]

2. Low Safety 2: Das System befindet sich in der Situation „Low Safety 2“, sobald Hindernisse in der Sicherheitszone auftauchen und sich diese auf beiden Seiten der „Zielseite“ befinden. In der Abbilung 2.13 ist zu erkennen, dass sich hier sowohl links als auch rechts der „Zielseite“ Hindernisse befinden.

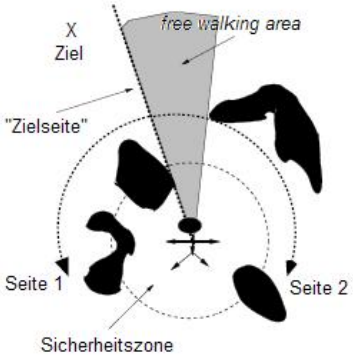

Abbildung 2.13: LS 2 - Low Safety 2, angelehnt an [13]

Kriterium 3: Ziellage zur free walking area

Das dritte Unterscheidungsmerkmal der Situationen tritt dann ein, wenn sich kein Hindernis in der Sicherheitszone befindet. Eine Kollision ist nicht imminent. Nun wird unterschieden, ob sich das Ziel innerhalb der zuvor bestimmten *free walking area* befindet.
Es gibt drei Möglichkeiten, die eine „High Safety“-Situation bestimmen:

3. High Safety Goal in Region: Liegt die Zielposition innerhalb der *free walking area*, so ist diese Situation gegeben.

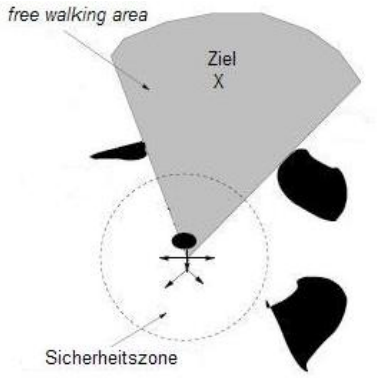

Abbildung 2.14: High Safety Goal in Region, angelehnt an [13]

Andernfalls wird das vierte und letzte Kriterium ausgewertet:

Kriterium 4: Beschaffenheit der free walking area

Das vierte Kriterium wertet eine direkte Eigenschaft der ermittelten *free walking area* aus: ihre Breite.

4. High Safety Wide Region: Hier liegt das Ziel außerhalb der *free walking area*, selbige erreicht oder überschreitet allerdings eine bestimmte Breite.

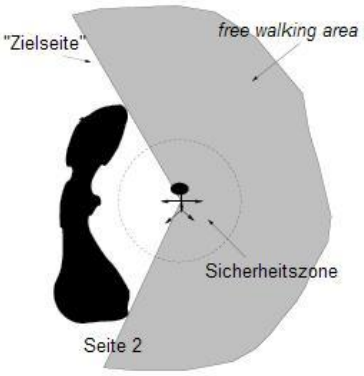

Abbildung 2.15: HSWR - High Safety Wide Region, angelehnt an [13]

5. High Safety Narrow Region: Im Gegensatz zu HSWR ist die *free walking area* schmal, das heißt die Größe unterschreitet eine bestimmte Breite. Das Ziel befindet sich hier allerdings auch außerhalb der *free walking area*.

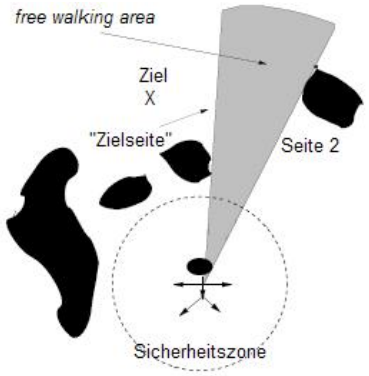

Abbildung 2.16: HSNR - High Safety Narrow Region, angelehnt an [13]

Besonders wichtig bei situationsbasierten Verhaltensmethoden ist die sichere Abbildung der Umgebung auf eine bestimmte Situation. Es darf also in keinem Falle vorkommen, dass die Umgebung nicht auf eine der oben genannten Situationen abgebildet werden kann, oder aber eine bestimmte Umgebungsvariante auf zwei Situationen.
Durch die binären Kriterien wird dies umgangen und am Ende der Auswertung ist gewiss eine der Situationen als Systemstatus ausgewählt.

2.3.2 Relation von Situation und Aktion

Zu jeder der oben genannten Situationen, die durch die Beschaffenheit der Umgebung und die Positionen von Roboter und Zielpunkt definiert wird, wird nun ein Satz an Aktionen benötigt, der in der entsprechenden Situationen ausgeführt wird.

Im Folgenden wird die für jede Situation angebrachte Aktion beschrieben, mit der eine kollisionsfreie Navigation ermöglicht wird.

1. Low Safety 1 (LS1): In dieser Situation befindet sich ein Hindernis in der Sicherheitszone. Vorrangiges Ziel ist es, durch einen Bewegungsbefehl die Sicherheitszone von Hindernissen zu befreien, dabei aber - sofern möglich - auf das Ziel zuzusteuern. Die Aktion versucht, das Hindernis zu umfahren und dabei den Roboter in Richtung der „Zielseite“ der *free walking area* zu manövrieren.
2. Low Safety 2 (LS2): Ähnlich wie in der Situation LS1 ist das Hauptziel der Aktion die Befreiung der Sicherheitszone von Hindernissen und dabei auf die Zielseite der *free walking area* zu navigieren. Dies ist nur dadurch möglich, dass der Roboter zwischen den Hindernissen hindurch fährt und die Hindernisse auf gleicher Distanz zu beiden Seiten hält.
3. High Safety Goal in Region (HSGR): Da sich hier das Ziel innerhalb der berechneten *free walking area* befindet, zudem die Sicherheitszone frei von Hindernissen ist, lautet hier die Aktion, den Roboter in die Richtung der Zielposition zu navigieren.
4. High Safety Wide Region (HSWR): Ziel der Aktion ist es, den Roboter in gewisser Entfernung entlang der „Zielseite“ der *free walking area* zu navigieren und somit das Hindernis, welches diese Seite definiert, zu umfahren. Die Navigation in Richtung der Zielseite schließt implizit ein Fahren in die Richtung des Ziels ein.
5. High Safety Narrow Region (HSNR): Da die Breite der *free walking area* in dieser Situation eine bestimmte Schwelle unterschreitet, muss die Aktion in diesem Falle den Roboter zwischen die Grenzen der *free walking area* zentrieren und ihn mittig hinein navigieren. Dies entspricht einer Navigation in der Mitte eines Korridors bzw. das mittige Hineinfahren in einen engen Bereich.

Alle Aktionen zu den jeweiligen Situationen erfüllen die Kriterien, die nötig sind, um eine kollisionsfreie Navigation zu einem Zielpunkt zu erfüllen: die Bewegungsrichtung ist stets die Zielrichtung, die implizit durch die „Zielseite“ der *free walking area* festgelegt wird, (HSGR, HSWR, HSNR) und umgeht Hindernisse, die in der Sicherheitszone auftauchen (LS1 und LS2).

Die Auswertung der Sensordaten führt also in jedem Berechnungszyklus zu einer eindeutigen Bestimmung einer Situation, aus der sich anschließend eine zugehörige Aktion ergibt. Mit Hilfe dieser Aktion wird die Kollisionsvermeidung durchgesetzt.
Voraussetzung ist allerdings das Vorhandensein der drei benötigten Informationen - Sensordaten, Zielposition, Rollstuhlposition.

2.3.3 Definition der „Nearness-Diagrams“

Die soeben dargestellte Situationsfindung (Abschnitt 2.3.1) basiert auf einer Art Diagramm, dem sogenannten „*Nearness-Diagram*“. Es sei noch einmal darauf hingewiesen, dass die Entwicklung in ihrer Grundfassung für runde Roboter entstanden ist. Die Erweiterung auf einen autonomen Rollstuhl, also einen nicht holonomen Roboter mit spezieller Form, erfolgt an entsprechender Stelle in Kapitel 3.

Grundlegend stellt ein solches „*Nearness-Diagram*“ die Umgebung dar, wobei dies durch die Einteilung des Arbeitsbereiches in eine bestimmte Anzahl an Sektoren n realisiert wird. Jeder einzelne Sektor enthält Informationen über die von den Laserscannern wahrgenommenen Hindernisse.
Im Folgenden wird dazu lediglich ein Arbeitsbereich W benötigt, der zweidimensional ist ($\mathbb{R}^2$). In diesem Arbeitsbereich ist jeder Punkt $p = (x, y) \in \mathbb{R}^2$ mit einer x- und einer y-Koordinate fest definiert.
Ein solcher Punkt $p_{robot} = (x_{robot}, y_{robot})$ ist die Position des Roboters.

Die Einträge im „*Nearness-Diagram*“ basieren auf den Sensordaten der Laserscanner. Diese Tiefeninformationen werden wiederum als Punktinformationen im System abgelegt. Jeder Hindernispunkt ist seinerseits durch einen Punkt $p_{obstacle} = (x_{obstacle}, y_{obstacle}) \in \mathbb{R}^2$ definiert. Somit haben wir eine Liste L, in der eine Anzahl an wahrgenommenen Hindernispunkten abgelegt ist. Eine detaillierte Beschreibung der Speicherung der Hindernisin-

formationen erfolgt in Abschnitt 3.2.1.

Zur Definition des „*Nearness-Diagram*“ wird weiterhin die Einführung einer Funktion nötig, welche die Umgebung auf die Einträge im „*Nearness-Diagram*“ abbildet.
So sei $\delta_i(x, L)$ eine Funktion, die den Abstand vom Zentrum des Roboters zum nächsten Hindernispunkt in Sektor i liefert. Der Rückgabewert der Funktion sei 0, wenn sich im Sektor i kein Hindernispunkt befindet: $\delta_i(x, L) = 0$. Der maximale Rückgabewert soll die Reichweite des Laserscanners sein: $\delta_i(x, L) \leq d_{max}$. Schließlich wird eine Liste an Hindernispunkten D definiert, die für jeden Sektor den nächsten Hindernispunkt zur Mitte des Roboters enthält, also $D = \{d_i = \delta_i(x, L), i = 1..n\}$.
Somit lassen sich zwei „*Nearness-Diagrams*“ definieren: das PND, welches die Nähe zu den umgebenden Objekten zur Mitte des Roboters darstellt, und das RND, welches die Nähe zu den Hindernissen zur Aussenseite des Roboters enthält.

Das PND wird wie folgt berechnet:

$$\begin{aligned} PND : \mathbb{R}^2 \times D &\rightarrow \{\mathbb{R}^+ \cup \{0\}\}^n \\ (x, d_i) &\rightarrow \{PND_i(x, d_i)\}^n \\ if\ d_i > 0,\ PND_i(x, d_i) &= d_{max} + 2R - d_i \\ else\ PND_i(x, d_i) &= 0 \end{aligned}$$

wobei

- d_{max} der maximalen Reichweite der Laserscanner entspricht.
- R dem Radius des Roboters entspricht.

Nach dieser Definition entsprechen also die Einträge im PND der Nähe zu den Hindernispunkten in den entsprechenden Sektoren. Je höher der Eintrag desto näher befindet sich ein Hindernis, ist der Wert des Eintrags 0 so befindet sich kein Hindernis an dieser Stelle beziehungsweise es befindet sich außerhalb der Reichweite der Laserscanner.

Auf diese Weise folgt die Definition eines zweiten Diagramms, des RND.

$$\begin{aligned} RND : \mathbb{R}^2 \times D &\rightarrow \{\mathbb{R}^+ \cup \{0\}\}^n \\ (x, d_i) &\rightarrow \{RND_i(x, d_i)\}^n \\ if\ d_i > 0,\ RND_i(x, d_i) &= d_{max} + E_i - d_i \\ else\ RND_i(x, d_i) &= 0 \end{aligned}$$

wobei wiederum

- d_{max} der maximalen Reichweite der Laserscanner entspricht.
- E_i der Funktion entspricht, die den Abstand vom Mittelpunkt des Roboters zu dessen Außenseite in Sektor i darstellt. Bei einem runden Roboter wäre dies der Radius.

Das RND repräsentiert also die Nähe zu den umgebenden Objekten zur Außenseite des Roboters analog zur Funktionsweise des PND.

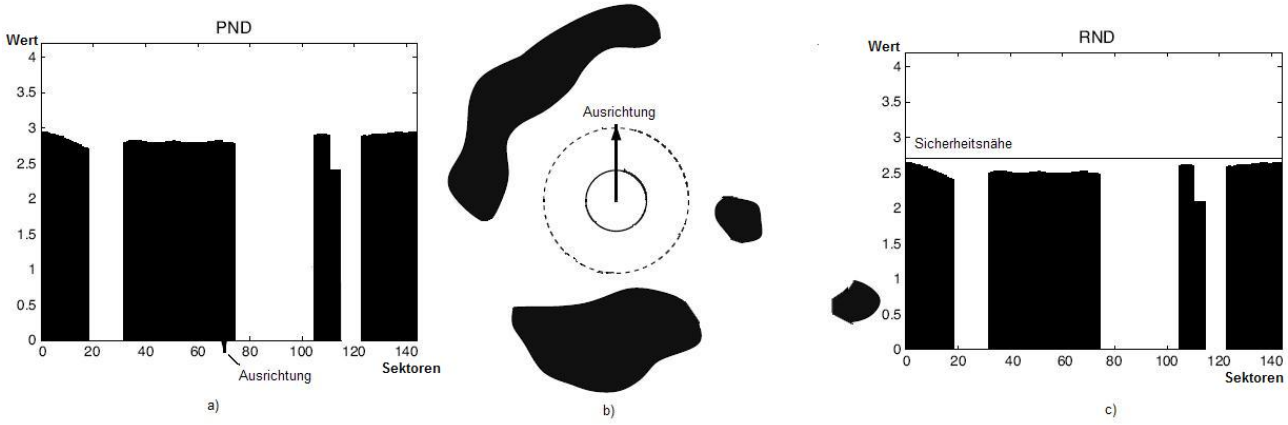

Abbildung 2.17: Beispiel von PND und RND, angelehnt an [18]

Die Abbildung 2.17 verdeutlicht die soeben beschriebene Methode. Dabei werden die Diagramme stets im Uhrzeigersinn um den Roboter erstellt. Der mittlere Eintrag entspricht der Blickrichtung des Roboters, die Einträge links und rechts davon folglich der linken und

rechten Seite der Umgebung. Es ist zu erkennen, dass die Objekte in Abbildung 2.17 b) um den Roboter herum einen Ausschlag an entsprechender Stelle des PND (2.17 a) und des RND (2.17 c) verursachen.

Der Lesbarkeit halber wird die Schreibweise der „*Nearness-Diagrams*“ im Folgenden festgelegt auf:
$PND_i(x, d_i)$ entspricht PND_i und $RND_i(x, d_i)$ entspricht RND_i.

Alle folgenden Untersuchungen und Entscheidungskriterien werden mit Hilfe dieser beiden Diagramme durchgeführt, da in ihnen alle relevanten Informationen zur lokalen Navigation enthalten sind.

2.3.4 Die „Nearness-Diagram“-Analyse

Die Situationsfindung geschieht anhand der soeben definierten Diagramme. Dabei werden sie zu verschiedenen Zwecken verwendet.

Die Relation zwischen Roboter und Hindernissen, als Auswertung für das Sicherheitskriterium, wird aus dem RND abgeleitet, da dieses die Nähe der wahrgenommenen Objekte zur Außenseite des Roboters repräsentiert.
Hierzu ist die Definition einer Sicherheitsdistanz ds nötig. Dies ist jene Distanz, welche die minimal zu tolerierende Entfernung vom Roboter zu den Hindernissen darstellt. Alle Objekte, die diese Distanz unterschreiten, stellen ein Sicherheitsrisiko dar und müssen vermieden werden.
So lässt sich eine Sicherheitsnähe ns bestimmen: $ns = dmax - ds$, welche innerhalb des RND als Sicherheitskriterium verwendet wird.

Das PND wird hinzugezogen, wenn es um die Herstellung einer Relation von Roboter und Zielposition geht. Dies geschieht durch die Berechnung einer so genannten *free wal-*

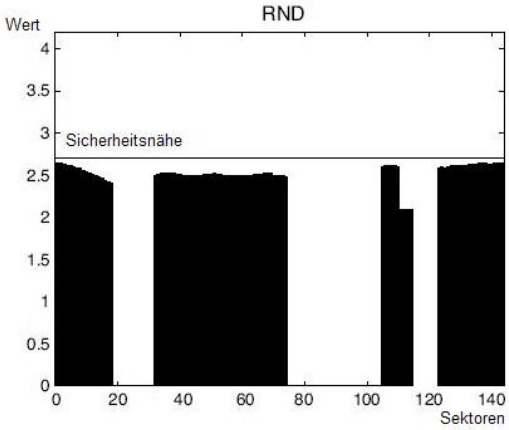

Abbildung 2.18: Sicherheitsnähe des RND, angelehnt an [18]

king area, welche bereits weiter oben eingeführt wurde und nun konkretisiert wird. Sie repräsentiert eine befahrbare Region zwischen Hindernissen und ist ein Hauptbestandteil der kollisionsvermeidenden Navigation der *„Nearness-Diagram“*-Methode.

Die Analyse des PND geschieht schrittweise, da vor der Bestimmung der Situation verschiedene Eigenschaften aus den Einträgen und Informationen in diesem Diagramm extrahiert werden. Diese sind in der Reihenfolge ihrer Detektion:

- Lücken zwischen den Hindernissen
- Regionen, welche durch 2 Lücken gebildet werden
- und schließlich die *free walking area*, die eine der berechneten Regionen mit speziellen Eigenschaften ist.

Dies führt uns zur Definition folgender Elemente eines PND:

- Lücken

Der erste Schritt der ND-Analyse besteht in der Findung von Lücken. Starke Unstetigkeiten zwischen zwei nebeneinander liegenden Einträgen im PND werden hierbei als Lücke interpretiert.

Zwei benachbarte Sektoren i und j enthalten genau dann eine Unstetigkeit, wenn die Differenz ihrer Einträge einen bestimmten Wert überschreitet. Dieser Wert entspricht dabei dem Durchmesser des Roboters (2R), da nur nach solchen Lücken gesucht werden soll, durch die der Roboter navigieren kann, also: $|PND_i - PND_j| > 2R$.
In Abbildung 2.19 sind die Lücken nummeriert zu erkennen.

Im Folgenden wird zwischen zwei Arten von Unstetigkeiten unterschieden: der aufsteigenden und der fallenden. Unterscheiden sich zwei benachbarte Sektoreinträge in der Form $|PND_i - PND_j| > 2R$ und $PND_i > PND_j$ so wird in diesem Falle von einer *aufsteigendenen Unstetigkeit von j nach i* oder von einer *fallenden Unstetigkeit von i nach j* gesprochen.

- Regionen

Eine Region wird durch zwei aufeinander folgende Lücken gebildet. Die Suche nach Regionen in der Umgebung entspricht der Suche nach *Tälern* innerhalb des PNDs. Abbildung 2.19 verdeutlicht diesen Sachverhalt.

Ein *Tal* ist in diesem Zusammenhang definiert als eine bestimmte Reihe von Sektoren $S = \{i + p\}_{p=0,\ldots,r}$, wobei $n > r \geq 0$ ist und folgende Eigenschaften erfüllt sein müssen:

- Es existieren zwei Unstetigkeiten, also:
  $|PND_i - PND_{i+1}| > 2R$ und $|PND_{i+p} - PND_{i+p+1}| > 2R$
- Eine dieser beiden Unstetigkeiten ist eine steigende, entweder von i nach $i - 1$ oder aber von $i + p$ nach $i + p + 1$:
  $PND_i > PND_{i+1}$ oder $PND_{i+p+1} > PND_{i+p}$
- Es dürfen keine weiteren Unstetigkeiten zweier benachbarten Sektoren innerhalb von S existieren.

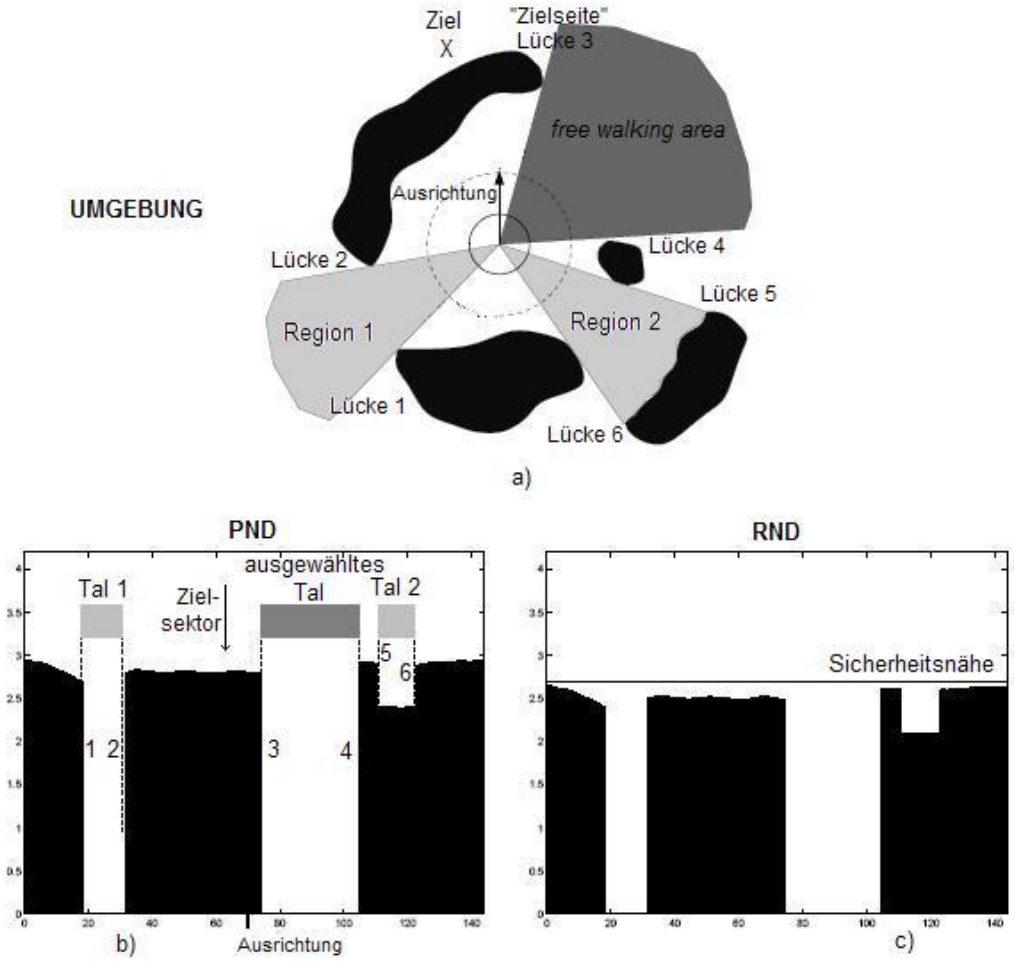

Abbildung 2.19: a) Umgebung mit Lücken, Regionen und *free walking area*, b) PND, c) RND, angelehnt an [13]

Auf diese Weise werden je zwei Lücken innerhalb des PND zu einem Tal zusammengefasst, sofern die oben stehenden Kriterien erfüllt sind. In Abbildung 2.19 werden sowohl Lücken, als auch die entsprechenden Regionen deutlich.

Hier wird beispielsweise Region 1 durch die beiden Lücken 1 und 2 gebildet, die beides steigende Unstetigkeiten innerhalb des PNDs sind. Aber auch Tal 3 ist eine befahrbare Region, da es nach oben stehender Definition eine aufsteigende Unstetigkeit hat.

Für alle ausgewählten Regionen dieser Abbildung sind die jeweiligen Kriterien erfüllt. Das heißt, dass jede Region durch zwei Unstetigkeiten begrenzt ist, eine davon eine steigende

Unstetigkeit ist und sich innerhalb der Region keine weiteren Unstetigkeiten befinden.

Besondere Beachtung muss der Situation entgegengebracht werden, wenn sich die Zielposition exakt zwischen dem Roboter und einem Hindernis befindet. In diesem Fall kann es passieren, dass der Sektor des PND, welcher die Zielposition enthält, im Folgenden als s_{goal} bezeichnet, nicht zu einer der berechneten Regionen gehört und nicht angefahren wird.
In diesem Fall wird eine künstliche Region erzeugt, die dem Zielsektor entspricht. Dies geschieht durch das Setzen des Wertes an der Stelle $PND_{s_{goal}} = 0$. Abbildung 2.20 verdeutlicht diesen Sachverhalt.

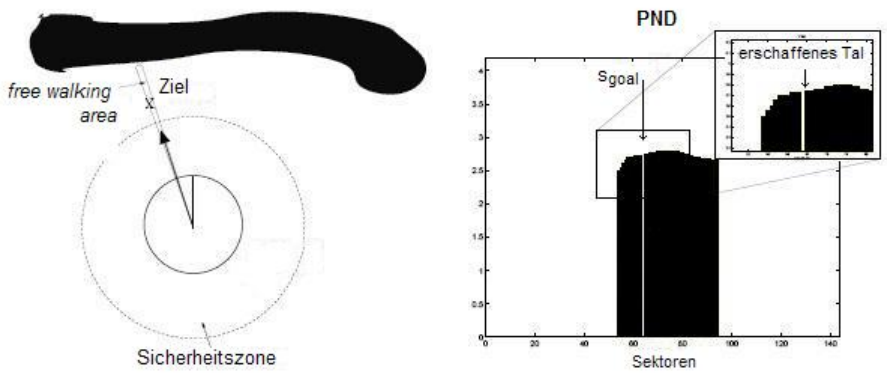

Abbildung 2.20: Künstliche Region, angelehnt an [13]

- *free walking area*

Aus den soeben berechneten Regionen wird nun eine ausgewählt, die im Folgenden als *free walking area* bezeichnet wird. Die Auswahl der *free walking area* hängt direkt mit der Lage der Zielposition zusammen.
So wird genau die Region ausgewählt, deren aufsteigende Unstetigkeit die geringste Entfernung zur Zielposition hat. Diese wird die „Zielseite“ der *free walking area* genannt. Anschließend wird geprüft, ob diese Region überhaupt befahrbar ist. Dies geschieht durch eine Korridor-Prüfung der entsprechenden Hindernispunkte (siehe Abschnitt 3.2.3). Sollte die ausgewählte Region nicht passierbar sein, so wird eine weitere Region mit der

zweitnächsten aufsteigenden Unstetigkeit auf Navigierbarkeit geprüft, bis schließlich eine Region als *free walking area* ausgewählt ist. Sollte dies nicht möglich sein, weil alle detektierten Regionen nicht befahrbar sind, ist dies ein Sonderfall, der separat behandelt werden muss. Eine Navigation mit Hilfe des „*Nearness-Diagrams*“ ist nicht mehr möglich, da das Vorhandensein einer *free walking area* für die weitere Berechnung zwingend erforderlich ist.

Am Ende dieser drei beschriebenen Schritte ist aus den zuvor berechneten Diagrammen ein besonderes Element gefiltert worden: die *free walking area*. Diese Region zeichnet sich dadurch aus, dass sie sowohl befahrbar ist, als auch eine Seite aufweist, die dem Zielpunkt zugewandt ist, die „Zielseite“ (siehe Abbildung 2.19).

2.3.5 Situationsfindung auf Basis des „Nearness-Diagrams“

Die grundlegenden Situationen und die dazugehörigen Aktionen, mit denen die kollisionsfreie Navigation durchgeführt wird, sind in den Abschnitten 2.3.1 und 2.3.2 bereits erläutert worden. Das Ziel dieses Abschnittes ist es nun, die Verbindung dieser Grundlage mit der soeben definierten Datenstruktur der Diagramme PND und RND herzustellen.

Hierzu erfolgt erneut eine Betrachtung des Entscheidungsbaumes (siehe Abbildung 2.21), diesmal in entsprechendem Zusammenhang mit den „*Nearness-Diagrams*“.

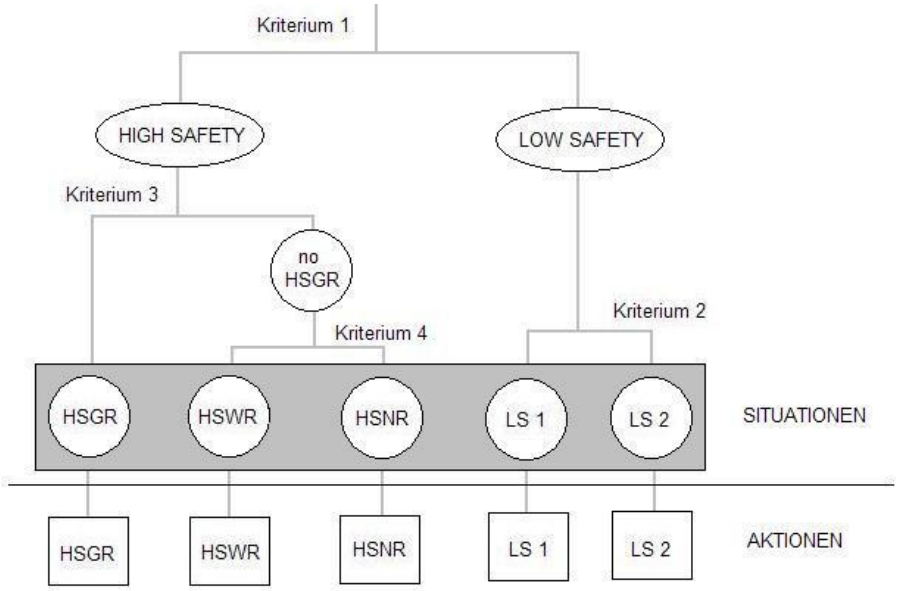

Abbildung 2.21: ND-Entscheidungsbaum

Kriterium 1: Sicherheit der Umgebung

Die Auswertung des ersten Merkmales wird mit Hilfe des RND durchgeführt. Da dieses Diagramm die Nähe des Roboters zu den umgebenden Objekten widerspiegelt, lässt sich mit Hilfe der definierten Sicherheitsdistanz ns überprüfen, ob Einträge im RND existieren, die diese Schwelle überschreiten.

Ist dies der Fall, so befindet sich das System im Zustand der niedrigen Sicherheit (LS), andernfalls im Zustand der hohen Sicherheit (HS).

Kriterium 2: Hindernislage in der Sicherheitszone

1. Low Safety 1 (LS1): Diese Situation wird genau dann erreicht, wenn sich die Hindernisse nur auf einer Seite der „Zielseite“ der „*free walking area*“ befinden, im Folgenden s_{rd} genannt[1]. Die Sektoren des RND, welche die Sicherheitsnähe überschreiten, sind also einseitig zur Lage des s_{rd}. Auf der entgegengesetzten Seite existieren keine Sektoren, deren Wert höher als die Sicherheitsnähe ist.

[1] s_{rd} abgeleitet aus dem englischen Begriff „*sector with rising discontinuity*“

In Abbildung 2.22 a) ist zu erkennen, dass sich die Ausschläge des RND, die die Sicherheitsnähe überschreiten, lediglich auf einer Seite des s_{rd} befinden.

2. Low Safety 2 (LS2): Im Gegensatz zu LS1 befinden sich hier Sektoren des RND, welche die Sicherheitsdistanz verletzen, auf beiden Seiten des s_{rd}. Abbildung 2.22 b) verdeutlicht diesen Sachverhalt.

Kriterium 3: Ziellage zur free walking area

3. High Safety Goal in Region (HSGR): Einfacherweise lässt sich diese Situation detektieren, sobald sich der Zielsektor (s_{goal}) innerhalb der ausgewählten Region befindet (siehe Abbildung 2.22 c).

Kriterium 4: Beschaffenheit der free walking area

Liegt der Zielsektor s_{goal} nicht innerhalb der *free walking area* wird das letzte Kriterium ausgewertet.

4. High Safety Wide Region (HSWR): Als auswertendes Merkmal wird hier die Breite der *free walking area* hinzugezogen. Die Anzahl der Sektoren gibt Auskunft darüber, wie breit sie ist, da jeder Sektor einer festen Gradzahl entspricht. Sobald also die Summe der Sektoren innerhalb der *free walking area* einen bestimmten Wert s_{max} erreicht oder überschreitet, zählt diese als „wide region“. Eine Breite von 90 Grad ist hierbei sinnvoll [18], dies entspricht einer Anzahl an

$$s_{max} = \frac{n}{4}$$

Sektoren (wobei n der Anzahl an Sektoren der Diagramme entspricht). In Abbildung 2.22 d) besteht die *free walking area* beispielsweise aus 102 Sektoren, was einer Breite von 255 Grad entspricht, somit die HSWR-Situation detektiert wird.

5. High Safety Narrow Region (HSNR): Unterschreitet die Breite der *free walking area* jedoch den Wert s_{max}, so ist die Region eng und löst die Situation HSNR aus (siehe Abbildung 2.22 e).

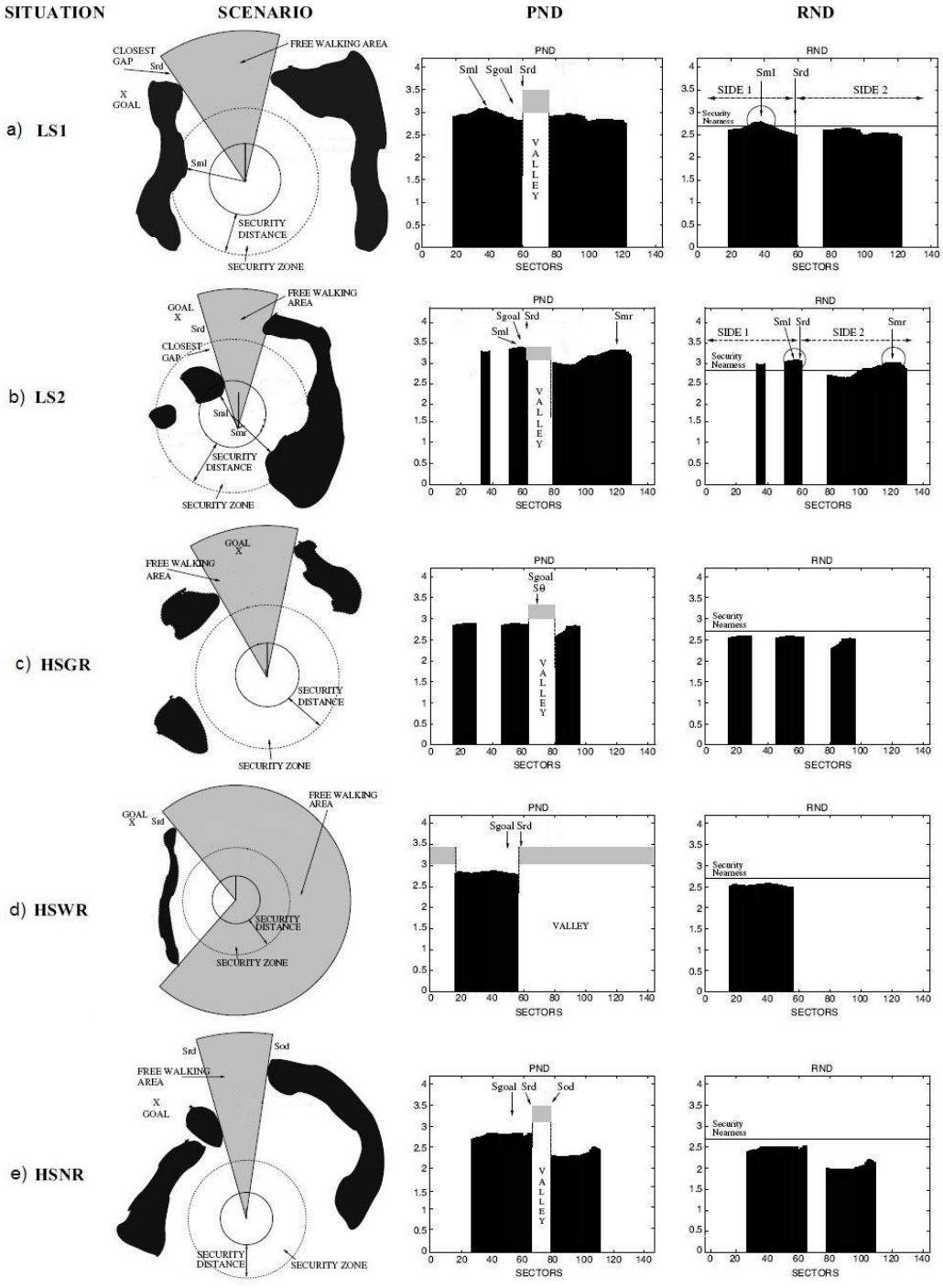

Abbildung 2.22: Zusammenfassung der Situationen, angelehnt an [13]

2.3.6 Aktionsberechnung auf Basis des „Nearness-Diagrams“

Die entsprechende Aktion zu der soeben bestimmten Situation führt schließlich die kollisionsfreie Navigation aus. Hierzu wird innerhalb der Aktions-Algorithmen ein Bewegungsbefehl berechnet, der sich in dem Tripel (v_m, θ, w) zusammensetzt, wobei

v_m der Geschwindigkeit

θ der Bewegungsrichtung

w der Rotationsgeschwindkeit

entspricht.
Nachstehend wird die Rechenvorschrift zur Kalkulation dieses Bewegungsbefehls dargelegt.

Bewegungsrichtung

Die Berechnung der Bewegungsrichtung entspricht bei der „*Nearness-Diagram*“-Navigation der Berechnung eines anzusteuernden Sektors. Der Algorithmus liefert also für jede Situation einen festen Sektor, $s_\theta \in \mathbb{R}$, als Richtungssektor, woraus sich die Bewegungsrichtung θ dann explizit aus der Winkelhalbierenden dieses Zielsektors ableitet.

$$\theta = bisec(s_\theta) = \pi - \frac{2 * \pi}{n} * s_\theta$$

Befindet sich das System im Zustand der hohen Sicherheit (HS), so muss den umgebenden Hindernissen keine besondere Beachtung entgegengebracht werden. Vorrangig wird hier in die Richtung der Zielposition gelenkt.

1. HSGR: Der Richtungssektor entspricht hierbei dem Zielsektor.

$$s_\theta = s_{goal}$$

2. HSWR: In dieser Situation liegt der Zielsektor nicht innerhalb der *free walking area*. Es wird hier lediglich die Grenze der *free walking area* (s_{rd}) als richtungsweisend

herangezogen, wobei gleichzeitig das diese Grenze definierende Hindernis umfahren wird. Dies geschieht durch die Addition eines festen Winkels.

$$s_\theta = s_{rd} \pm \frac{s_{max}}{2}$$

wobei

- s_{max} als $\frac{n}{4}$, also 90 Grad, definiert wurde.

3. HSNR: Da die Breite der *free walking area* eine gewissen Größe unterschreitet, wird in diesem Falle mittig in die Region hineingelenkt, um eine Kollision mit den beiden begrenzenden Hindernissen zu umgehen.

$$s_\theta = \frac{s_{rd} + s_{od}}{2}$$

wobei

- s_{od} der Sektor ist, der die andere Grenze der *free walking area* beinhaltet. Somit spannen s_{rd} und s_{od} die *free walking area* auf.

Im Zustand der niedrigen Sicherheit ist die Hauptaufgabe des Navigationsalgorithmus das Befreien der Sicherheitszone von Hindernissen. Aus diesem Grund sind hier die Sektoren von Bedeutung, welche ein Hindernis enthalten.

4. LS 1: Es befindet sich auf einer Seite des s_{rd}-Sektors ein Hindernis innerhalb der Sicherheitszone. Nun wird der Sektor berechnet, der sowohl vom Hindernis fort als auch in Richtung des Zieles navigiert. Hierzu wird der Zielsektor analog zur Situation HSWR berechnet, und zudem ein Winkel γ addiert, der von der Position des nächsten Hindernisses auf der entsprechenden Seite abhängig ist:

$$s_\theta = s_{goal} \pm (\frac{s_{max}}{2} \pm \gamma)$$

$$\gamma = \frac{D_s - D_{s_{ml}}}{D_s} * \left|(\pi + s_{ml}) - \left|s_{rd} - \frac{s_{max}}{2}\right|\right|$$

wobei

- D_s der Sicherheitsdistanz entspricht.
- s_{ml} der Sektor mit dem nächsten Hindernis (welches die LS1-Situation ausgelöst hat) ist.
- $D_{s_{ml}}$ die Distanz zum Hindernis in Sektor s_{ml} ist.

5. LS 2: Da sich Hindernisse auf beiden Seiten der „Zielseite“ der *free walking area* befinden, muss der Roboter zwischen diesen Hindernissen „hindurch“ gelenkt werden.

$$s_{med_1} = \frac{s_{m_l} + s_{m_r}}{2} \quad s_{med_2} = \frac{s_{m_l} + s_{m_r} + n}{2}$$

$$if \ \ |s_{rd} - s_{med_1}| < |s_{rd} - s_{med_2}| \, then \ s_\theta = s_{med_1} \pm c$$

$$else \ s_\theta = s_{med_2} \pm c$$

wobei

- s_{ml} dem Sektor mit dem nächsten Hindernis auf der linken Seite des s_{rd}-Sektors entspricht.
- s_{mr} dem Sektor mit dem nächsten Hindernis auf der rechten Seite des s_{rd}-Sektors entspricht.
- der Parameter c als Korrekturterm verwendet wird, um den Roboter auf die gleiche Distanz zu den beiden umgebenden Hindernissen zu bringen.

Um nicht zu heftige Drehbewegungen und ein natürliches Lenkverhalten zu erzeugen, wird der Bereich der anfahrbaren Richtungen auf das Intervall $\theta \in [-\pi, \pi]$ beschränkt.
Mit Hilfe dieser Rechenvorschriften ist für jede der fünf grundlegenden Situationen eine Bewegungsrichtung zu bestimmen, die den Roboter kollisionsfrei in die Richtung des Ziels navigiert.

Bewegungsgeschwindigkeit

Die Bewegungsrichtung allein macht noch keine vollständige Bewegung aus. So wird an dieser Stelle die Rechenvorschrift für die Kalkulation der Bewegungsgeschwindigkeit v_m geliefert.
Diese ist abhängig vom Zustand der Sicherheit. In Situationen der hohen Sicherheit (HS) muss keine Rücksicht auf Hindernisse genommen werden, da eine Kollision nicht imminent ist. Bei niedrigen Sicherheiten (LS) muss die Geschwindigkeit entsprechend gedrosselt werden, um ein „vorsichtiges“ Umfahren der Hindernisse zu gewährleisten.
Dies wird auf folgende Weise erreicht:

1. HS-Situationen (HSGR, HSWR, HSNR):
$$v_m = v_{max} * (\frac{\frac{\pi}{2} - |\theta|}{\frac{\pi}{2}})$$
2. LS-Situationen (LS1, LS2):
$$v_m = v_{max} * \frac{D_{obs}}{D_s} * (\frac{\frac{\pi}{2} - |\theta|}{\frac{\pi}{2}})$$
wobei
   - v_{max} die maximale Geschwindigkeit des Roboters sei.
   - D_s die eingehend definierte Sicherheitsdistanz sei.
   - D_{obs} die Distanz zum nächsten Hindernis innerhalb der Sicherheitszone sei.

Auf diese Weise wird sichergestellt, dass die Geschwindigkeit herabgesetzt wird, sobald Hindernisse in die Sicherheitszone gelangen. Je näher das Hindernis, desto langsamer wird die Bewegung um eine Kollision zu vermeiden.
Gleichfalls wird die derzeitige Bewegungsrichtung beachtet. So wird die Geschwindigkeit reduziert, sobald starke Schwankungen in der Richtung auftreten. Dies ermöglicht weiche Drehungen und verhindert ein Umkippen des Roboters bei hohen Geschwindigkeiten in Verbindung mit starken Richtungsänderungen.

Distanz zum nächsten Hindernis in cm	Bewegungsrichtung θ in Grad	errechnete Geschwindigkeit v_m in mm/sec
kein Hindernis in der Sicherheitszone	0	300 ($= v_{max}$)
40	0	240
30	0	180
20	0	120
10	0	60
kein Hindernis	45	150
40	45	120
30	45	90
20	45	60
10	45	30
kein Hindernis	90	0
40	90	0
30	90	0
20	90	0
10	90	0

Tabelle 2.1: Geschwindigkeitsreduzierung mit $D_s = 50cm,\ v_{max} = 300mm/sec$

Die Tabelle 2.1 verdeutlicht die Arbeitsweise dieser Funktion, indem sie beispielhafte Geschwindigkeitsberechnungen darstellt.

Rotationsgeschwindigkeit

Schließlich wird aus der berechneten Bewegungsrichtung die nötige Rotationsgeschwindigkeit berechnet, um den vollständigen Bewegungsbefehl zu erzeugen. Dies ist nötig, um bei kleinen Richtungsänderungen nicht mit der maximal möglichen Lenkbewegung zu reagieren, sondern diese auf die Richtungsänderung abzustimmen. Bei kleinen Änderungen ist auch nur eine leichte Drehbewegung nötig, während eine hohe Rotationsgeschwindigkeit angestrebt wird, wenn die Richtungsänderung hoch ist.
Dies wird folgendermaßen realisiert:

$$w = w_{max} * \frac{\theta}{\frac{\pi}{2}}$$

wobei

- w_{max} die maximale Rotationsgeschwindigkeit darstellt.

In nachstehender Tabelle 2.2 lässt sich erkennen, wie sich der Befehl der Rotationsgeschwindigkeiten bei beispielhaften Bewegungsrichtungen verhält:

Bewegungsrichtung θ in Grad	errechnete Rotationsgeschwindigkeit w in rad/sec
0	0
22,5	0,1964
45	0,3927
67,5	0,5891
90	0,7854 (= w_{max})

Tabelle 2.2: Rotationsgeschwindigkeit bei $w_{max} = 0,7854 rad/sec$

Am Ende dieser Kalkulation steht nun ein Bewegungsbefehl in der Form (v_m, θ, w), mit welchem die kollisionsfreie Navigation durchgeführt werden kann. Je nach Situation wird durch den Befehl erreicht, dass ein Hindernis in der Sicherheitszone umgangen wird (LS1),

zwischen zwei Hindernissen hindurch navigiert wird (LS2) oder aber direkt (HSGR) oder indirekt auf das Ziel zugesteuert wird (HSWR und HSNR).
Nachstehende Tabelle 2.3 fasst die soeben dargelegte Berechnung der Bewegungsbefehle zusammen.

Situation	Bewegungsrichtung $\theta = bisec(s_\theta)$	Geschwindigkeit	Rotations-geschwindigkeit
HSGR	$s_\theta = s_{goal}$	$v_m = v_{max} * (\frac{\frac{\pi}{2} - \lvert\theta\rvert}{\frac{\pi}{2}})$	$v_m = w_{max} * \frac{\theta}{\frac{\pi}{2}}$
HSWR	$s_\theta = s_{goal} \pm \frac{s_{max}}{2}$		
HSNR	$s_\theta = \frac{s_{rd}+s_{od}}{2}$		
LS1	$s_\theta = s_{goal} \pm (\frac{s_{max}}{2} + \gamma)$ $\gamma = \frac{D_s - D_{s_{ml}}}{D_s} * \lvert(\pi + s_{ml}) - \lvert s_{rd} - \frac{s_{max}}{2}\rvert\rvert$	$v_m = v_{max} * \frac{D_{obs}}{D_s} * (\frac{\frac{\pi}{2} - \lvert\theta\rvert}{\frac{\pi}{2}})$	
LS2	$s_{med_1} = \frac{s_{m_l}+s_{m_r}}{2}$ $s_{med2} = \frac{s_{m_l}+s_{m_r}+n}{2}$ $if \ \lvert s_{rd} - s_{med_1}\rvert < \lvert s_{rd} - s_{med_2}\rvert then \ s_\theta = s_{med_1} \pm c$ $else \ s_\theta = s_{med_2} \pm c$		

Tabelle 2.3: Aktionsberechnungen

2.3.7 Vorteile der „Nearness-Diagram“-Navigation

In diesem Abschnitt soll erläutert werden, wo die Vorteile im Vergleich zu den anderen beschriebenen Methoden (siehe Abschnitt 2.2) liegen.

So sind die bekannten Nachteile der Methoden, die Potenzialfelder als Grundlagen verwenden (siehe auch 2.2.1), dass diese sich unter Umständen in Situationen bringen können, aus denen sie nicht mehr entweichen können. Dies geschieht aufgrund lokaler Minima des berechneten Kräfteflusses, beispielsweise beim Einfahren in ein U-förmiges Hindernis (Abbildung 2.4). So ziehen die Kräfte den Roboter stets in die Mitte dieses Bereiches hinein. Mit Hilfe der „*Nearness-Diagram*“-Navigation können solche Hindernisse umgangen werden. Die Voraussetzung dafür ist allerdings, dass das U-förmige Hindernis komplett sichtbar ist. Auf diese Weise wird verhindert, dass es als Region erkannt und angesteuert wird. Nach [16] ist ein weiterer Nachteil der PFM, dass es zu stark uneinheitlichen Bewegungen in besonders unstrukturierten oder engen Umgebungen kommt, da hier der Kräftefluss unter Umständen sehr uneinheitlich ist. Solche oszillierenden Bewegungen werden bei der „*Nearness-Diagram*“-Methode nicht betrachtet, da hier - beispielsweise in engen Situationen - auch sehr nah an ein Hindernis herangefahren wird, um solch eine Situation zu meistern, was bei den PFM aufgrund der starken abstoßenden Kräfte eines Hindernisses normalerweise selten auftritt.

Einer der Nachteile der VFH-Methoden ist nach [16] die Missachtung der Breite eines Roboters bei der Berechnung der Bewegungen. Dieses geschieht innerhalb der „*Nearness-Diagram*“-Navigation explizit bei der Berechnung möglicher Regionen, um solche zu vermeiden, durch die nicht navigiert werden kann. Die Weiterentwicklung VFH+ beseitigte die meisten Problemfälle, nur war es in dieser Lösung nicht möglich, auf Hindernisse zuzusteuern, was besonders in engen Bereichen teilweise unabdingbar ist. In der „*Nearness-Diagram*“-Navigation ist es nicht ausgeschlossen, in Richtung eines Hindernisses zu steuern, sofern die Region es zulässt.

„*Elastic-Band*“-Methoden sind auf eine vollständig bekannte Umgebung angewiesen, um im ersten Schritt einen globalen Pfad zum Ziel zu berechnen (siehe auch Abschnitt 2.2.3). Eine „EB“-Navigation in unbekanntem Gelände ist also nicht möglich, wogegen das „*Nearness-Diagram*“ nur die aktuell sichtbare Umgebung betrachtet. So kann auch in vollkommen unbekanntem Gelände navigiert werden.

Die Berechnung des Bewegungsbefehles innerhalb der DWA-Methoden erfolgt aufgrund einer stark heuristischen Funktion [16], die versucht die Zielposition, die Bewegungsgeschwindigkeiten des Roboters und die Hindernisvermeidung abzugleichen. Diese Gleichung wird durch den heuristischen Charakter leicht unvorhersehbar und schlecht planbar. Die Bewegungen der „*Nearness-Diagram*“-Navigation basieren auf einem abgeschlossenen Satz an Situationen und den daraufhin festgelegten Aktionen. Es ist also in jedem Schritt möglich abzusehen wie die Bewegung aussehen wird.

2.4 Grundlagen von autonom navigierenden Rollstuhlsystemen

Allen hier vorgestellten Systemen ist eine gemeinsame Zielsetzung inhärent:
Behinderte beziehungsweise in der Bewegung eingeschränkte Personen sollen unterstützt werden. Mit steigendem Entwicklungsgrad in der Robotik ist ein autonom steuerndes Fahrzeug oder ein sich selbstständig bewegender Roboter keine Zukunftsvision. Gegenteilig haben die Entwicklungen der letzten Jahre gezeigt, dass auf dem Gebiet der autonomen Steuerung großes Potenzial liegt. Besonders die Unterstützung eingeschränkter Personen ist ein Bereich, der intensiv erforscht wurde und wird.

Der Personentransport ist hierbei von besonderer Bedeutung, da ein Höchstmaß an Sicherheit und Komfort gewährleistet sein muss. Verschiedene entwickelte Systeme versuchen diese Anforderungen zu realisieren.

2.4.1 Stand der Technik: Autonom navigierende Rollstuhlsysteme

Die folgende Liste ist eine kleine Auswahl an entwickelten Systemen. Die Beschränkung erfolgte aufgrund der Übersichtlichkeit, denn es existieren weltweit rund dreißig weitere ähnliche Architekturen. Die hier vorgestellten Rollstühle lassen sich aufgrund ihres ähnlichen Aufbaus gut vergleichend darstellen.

- ***Rolland* - Der Bremer autonome Rollstuhl**

An der Universität Bremen wird seit dem Jahr 1995 an diesem Projekt gearbeitet. Bei dem hier betrachteten Modell handelt es sich um die dritte Version einer Reihe von Entwicklungen, weshalb es als *Rolland 3* bezeichnet wird.
Rolland 3 ist ein differenziell angetriebener Rollstuhl der Firmenmarke Meyra, ausgestattet mit einigen technischen Erweiterungen. So befinden sich frontal und rückseitig montierte Infrarot-Laserscanner, die eine Wahrnehmung der Umgebung ermöglichen. Das Fahrwerk ist mit einer Odometrie-Sensorik ausgerüstet worden, welche die Bewegung der Räder aufzeichnet und so die Bewegungswahrnehmung realisiert. Weiterhin ist es möglich an der Lehne des Rollstuhls ein Kamerasystem zu befestigen, das eine omnidirektionale Sicht der Umgebung ermöglicht.
Alle Sensordaten werden über eine Schnittstelle einem angeschlossenen Laptop zur Verfügung gestellt, der die softwareseitige Navigation übernimmt.
Derzeit sind mehrere Steuerungsmodule in der Entwicklung, die autonome Navigation ermöglichen sollen, unter anderen der Gegenstand dieser Diplomarbeit, welcher auf der „*Nearness-Diagram*“-Methode basiert, aber auch Versionen des „*Dynamic-Window-Approach*“ (siehe Abschnitt 2.2.4) und des „*Virtual-Force-Field*“ werden verfolgt.
Eine detaillierte Beschreibung der Architektur des *Rolland 3* und einiger ausgewählter Module findet sich in Kapitel 3, da dieses System die Grundlage für das entwickelte Navigationsmodul darstellt.

Abbildung 2.23: Rolland 3 der Universität Bremen [12]

- **Triton 3 - Universität Zaragoza, Spanien**

Eine ganz ähnliche Architektur weist ein Projekt der Universität Zaragoza in Spanien auf. Der hier entwickelte autonome Rollstuhl *Triton 3* ist ebenfalls ein handelsüblicher elektronischer Rollstuhl, der über einen Differenzialantrieb betrieben wird. Im Unterschied zum Rollstuhl der Universität Bremen verfügt dieses Model jedoch nur über einen frontal montierten SICK-Laserscanner, welcher über einen 180-Grad-Sichtbereich nach vorne verfügt. Dieser Fakt stellt größere Anforderungen an das System, da hierbei nur aktuelle Umgebungsdaten von der Vorderseite des Rollstuhles geliefert werden können. Zwei auf dem System montierte Computer übernehmen zum einen die Bewegungskontrolle, zum anderen die Berechnungen der Navigationsarchitektur [24].
Das für dieses System entwickelte Steuerungsmodul ist ein so genanntes hybrides System [22], welches aus drei Teilen besteht:

- Der Model-Builder erzeugt mit Hilfe von aktuellen und vergangenen Sensordaten eine Abbildung der Umgebung des Rollstuhls.
- Der Planer berechnet mit Hilfe von dynamischen Navigationsfunktionen einen initialen Pfad zur Zielposition.
- Das reaktive Modul setzt schließlich die kollisionsfreie Navigation auf Grundlage der „*Nearness-Diagram*“-Methode durch.

Abbildung 2.24: Triton 3 der Universität Zaragoza [15]

- **MAid: Mobility Aid for Elderly and Disabled People**

Das Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW) in Ulm entwickelte im Laufe der letzten Jahre den „intelligenten“ Rollstuhl MAid. Sein Aufgabenfeld liegt in der Unterstützung stark motorisch eingeschränkter Personen sowohl im öffentlichen Umfeld, als auch im heimischen Bereich [28]. Auch dieser Rollstuhl ist mit einer Reihe von Sensoren ausgestattet worden. Dies schließt einen Laserscanner im vorderen Brustbereich des Fahrers, Odometrie-Sensoren und Entfernungssensoren rund um die Außenseite des Rollstuhles ein.
Die Funktionalität dieses elektronischen Rollstuhls erstreckt sich von lokalen Navigationsmanövern in engen Bereichen, wie zum Beispiel Innenräumen, bis zur Steuerung in hochfrequentierten öffentlichen Umgebungen, wie Bahnhofsplätzen oder Ähnlichem.
Eine Kommunikation mit dem Rollstuhl über natürliche Sprachbefehle ist genauso möglich wie die Begleitung einer Person mit Hilfe eines Personen-Verfolgungsmoduls.

- **MICA: Mobile Internet Connected Assistant**

Ein weiteres mobiles System wurde in Schweden entwickelt. Der MICA-Rollstuhl ist ein elektronischer Rollstuhl der Firma Boden Rehab AB. Seine Architektur entspricht der des *Rolland*, allerdings befinden sich die Antriebsräder in dieser Variante an der Vorderachse, während die hinteren Räder nur Gleit-Räder sind [6]. Dies unterscheidet MICA von

Abbildung 2.25: MAid der FAW Ulm [28]

den restlichen hier vorgestellten Rollstühlen. Auch die Befestigung des Entfernungssensors unterscheidet sich von den übrigen Systemen. Der entfernungsmessende Laserscanner ist oberhalb der Kopfposition des Fahrers montiert.
Ein Hauptaugenmerk der Entwickler ist die Anbindung MICAs an das Internet über eine kabellose WLAN-Schnittstelle, die eine Fernsteuerung des Rollstuhls erlaubt. Weiterhin ist eine Steuerung über ein aufsetzbares Kopfteil möglich, wodurch eine Lenkung mit Hilfe von Kopfbewegungen realisiert wird.

Abbildung 2.26: MICA [6]

2.4.2 Grundlagen der Rollstuhlbewegung

Kollisionsfreie Navigation geschieht stets auf Grundlage von Bewegungen. Die korrekten Bewegungsbefehle zu errechnen, die den Roboter von der Ausgangsposition in die Zielposition überführen, ist Aufgabe des Navigationsmoduls. So wird in jedem Berechnungszyklus aufgrund der Sensordaten ermittelt, mit welcher Bewegung ein Roboter an einem Hindernis vorbei in Richtung des Zieles navigiert. Doch die Unterschiede in der Roboter-Bewegung sind mannigfaltig, von mehrbeinigen Laufrobotern bis hin zu fahrzeugähnlichen Robotern mit einer oder mehreren Achsen.

Dieser Abschnitt soll sich mit den Grundlagen der Bewegung von fahrzeugähnlichen Robotern beschäftigen, da dies der Architektur des Bremer autonomen Rollstuhls *Rolland* entspricht.
Während die Vorgängerversion noch mit Hilfe einer Ackermann-Lenkung entsprechend der eines KFZ angetrieben wurde, so wird der aktuelle Prototyp *Rolland 3* (siehe Abbildung 2.27, links) mit Hilfe eines Differenzialantriebs fortbewegt. Dabei ist die Lenkachse gleichzeitig die Antriebsachse. Die vorderen Räder sind lediglich mitgleitend (siehe Abbildung 2.27, rechts).

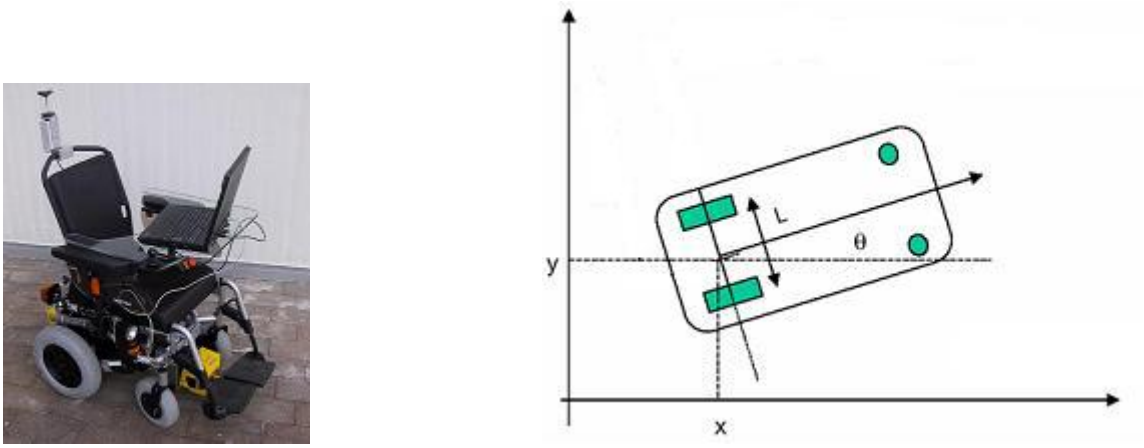

Abbildung 2.27: Links: Rolland, Rechts: schematischer Differenzialantrieb [12]

Der Differenzialantrieb ist von den meisten Antriebsarten am einfachsten zu konstruieren. Die Besonderheit dabei liegt darin, dass die zwei hinteren Räder mit je einem Motor angetrieben werden, die beiden vorderen nur als Stützräder dienen. Laufen die beiden Motoren der hinteren Räder mit der gleichen Drehzahl, so bewegt sich der Rollstuhl nach vorne (beziehungsweise nach hinten). Besteht eine Drehzahldifferenz, wird eine Drehung erzeugt. Dreht beispielsweise das rechte Rad schneller als das linke, so dreht sich der Rollstuhl in die linke Richtung. Je größer die Differenz der beiden Drehzahlen desto schneller die Drehbewegung.
Drehen die beiden Antriebsräder sogar in die entgegensetzten Richtungen, so wird erreicht, dass sich der Rollstuhl auf der Stelle dreht.

Mit Hilfe des Differenzialantriebs wird die der Ackermann-Lenkung innewohnende Einschränkung der Bewegungsfreiheit verringert. So ist es mit der Ackermann-Lenkung nicht möglich gewesen, auf der Stelle zu drehen, so wie es mit differentiell angetriebenen Motoren möglich ist. Es konnte lediglich eine Drehbewegung auf einer Kreisbahn erzeugt werden.

3 Implementierung und Umsetzung: Das entwickelte System

Dieses Kapitel beschreibt auf ausführliche Weise die Implementierung und die dazu verwendeten Algorithmen und Funktionen. Hierzu wird im ersten Abschnitt des Kapitels das System des *Rolland 3* beschrieben, für welches das Navigationsmodul entwickelt wurde, zum anderen wird in einem zweiten Teil die Methode der „*Nearness-Diagram*“-Navigation konkretisiert und geschildert.
Eine Betrachtung der Änderungen aufgrund der speziellen Form des Rollstuhls schließt sich an die entsprechenden Abschnitte an. Dies betrifft sowohl die Planung als auch die Ausführung der kollisionsvermeidenden Methode.

3.1 Beschreibung Rolland 3

Eine kurze Beschreibung des Bremer Autonomen Rollstuhls *Rolland* wurde bereits bei der Erläuterung der Grundlagen in Abschnitt 2.4.1 gegeben. Dieses Kapitel soll nun einen detaillierteren Blick auf den Rollstuhl werfen und die einzelnen Komponenten des Systems beschreiben.

3.1.1 Hardware

Die Basis-Architektur des *Rolland 3* (siehe Abbildung 3.1) ist ein handelsüblicher Rollstuhl der Firma Meyra. Es handelt sich hierbei um die Version Champ Modell 1.594, die seitens der Universität Bremen mit einigen technischen Erweiterungen ausgestattet wurde.

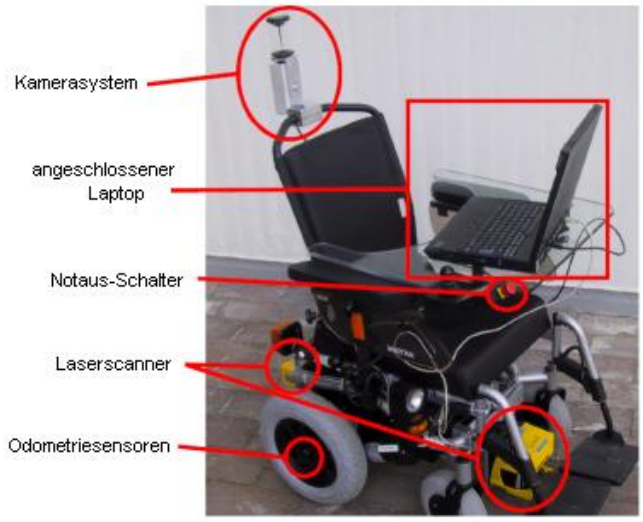

Abbildung 3.1: Detailbetrachtung: Rolland 3 [12]

Im Vergleich zu seinen Vorgängern wird der *Rolland 3* mit Hilfe eines Differenzialantriebes bewegt, was für eine höhere Beweglichkeit sorgt, da beispielsweise auf der Stelle gedreht werden kann (siehe auch Abschnitt 2.4.2). Die Antriebsachse befindet sich im hinteren Bereich des Rollstuhls, während die vorderen Räder lediglich Stützräder sind und nur gleiten, da auch die Lenkung mit Hilfe der hinteren Räder erreicht wird.

An den Rädern der Hinterachse befinden sich die Odometriesensoren. Die zwei magnetischen Sensoren der Marke MiniCoder GEL 248 messen die Drehungen der Räder und können so verwendet werden, um jederzeit die zurückgelegte Strecke und die aktuelle Position zu berechnen. Ein solches Verfahren zur Positionsbestimmung wird als „*Dead Reckoning*“ bezeichnet.

Im vorderen und im hinteren Bereich sind kurz unterhalb der Sitzposition zwei Laserscanner der Marke Leuze Lumiflex RotoScan montiert. Diese Infrarotscanner haben eine maximale Reichweite von 50 Metern und scannen die Umgebung in einem Winkel von 190 Grad nach vorne und nach hinten bis zu 25mal in der Sekunde. Durch die Art der Anbrin-

gung entsteht zu beiden Seiten des Rollstuhls eine Zone, die nicht von den Laserscannern abgedeckt wird. Diese tote Zone wird softwareseitig durch die Erstellung einer lokalen Karte mit den Hindernisinformationen aus vergangenen Scans (dem *Evidence Grid*, siehe auch Abschnitt 3.2.1) abgedeckt.
Alle Sensoren werden über ein USB-Hub einem extern angeschlossenen Laptop zur Verfügung gestellt, auf welchem die Simulations- und Testumgebung GTRolland vom Benutzer zur Steuerung des Rollstuhls verwendet werden kann.

3.1.2 Software

Die softwareseitige Steuerung des *Rolland 3* wird über die Simulations- und Steuerungssoftware GTRolland realisiert (siehe Abbildung 3.2). Sie basiert auf der Entwicklung einer Steuerungssoftware für Laufroboter des Robocup German Team und wurde für die Verwendung für einen autonomen Rollstuhl entsprechend angepasst. Die Software wird unter Windows XP betrieben und kann mit Hilfe der Entwicklungsumgebung Microsoft Visual Studio geändert und weiterentwickelt werden.
Der besondere Vorteil der Software ist die stark modulare Entwicklung. So können auch während der Laufzeit verschiedene Module zur Wahrnehmung, Steuerung oder Navigation aktiviert oder deaktiviert werden. Eines diese Module ist die verhaltensbasierte Steuerung der „*Nearness-Diagram*“-Methode, deren Implementierung in diesem Abschnitt beschrieben wird.

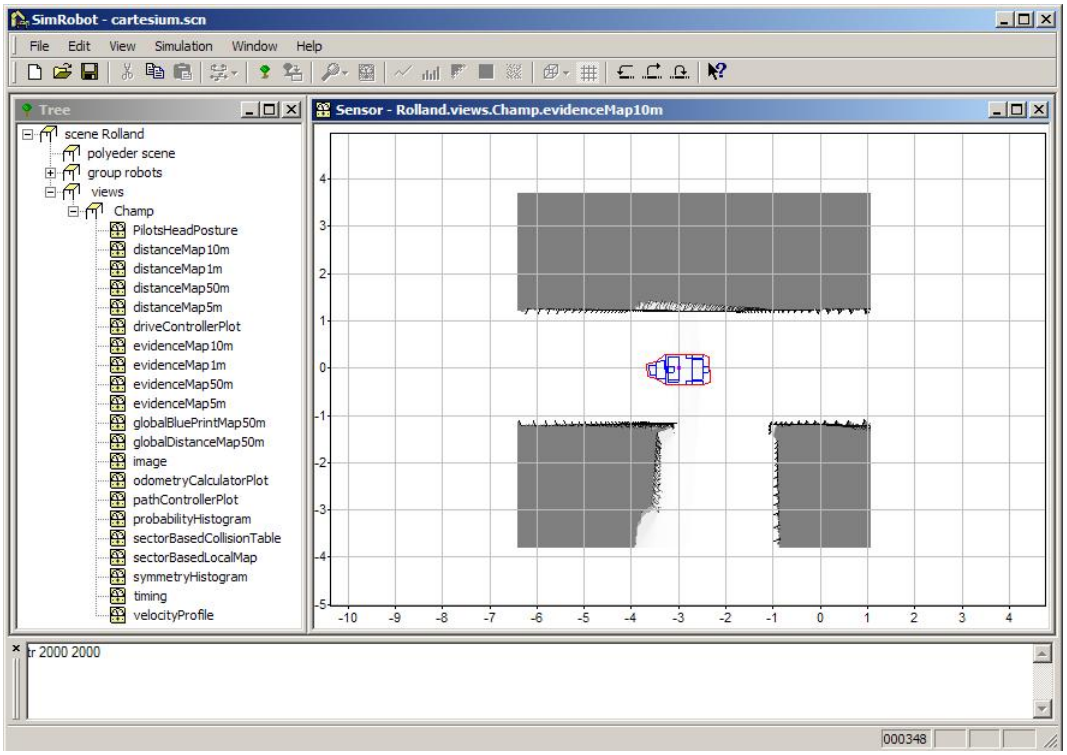

Abbildung 3.2: Beispielabbildung der Entwicklungsumgebung

3.2 Implementierung und Umsetzung

Die Beschreibung des entwickelten Navigationsmoduls gliedert sich entsprechend dem Aufbau von reaktiven Verhaltensmethoden nach [1] (siehe Abbildung 3.3).

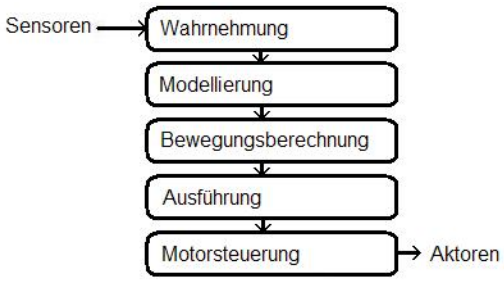

Abbildung 3.3: Erweiterter Zyklus reaktiver Methoden

Somit werden im ersten Abschnitt die Methoden erläutert, die zur Sammlung und Verarbeitung der Sensordaten benutzt werden und so die „Wahrnehmung“ des Systems dar-

stellen. Im zweiten Schritt wird dargestellt, wie jene Daten dazu verwendet werden, das interne Weltmodell zu modellieren und die Umgebung zu repräsentieren. Anschließend wird dargelegt, wie dieses Modell zur Situationsfindung dient und auf welche Weise die Berechnung der kollisionsfreien Bewegung stattfindet. Diese Berechnungsphase schließt eine detaillierte Beschreibung der veränderten Mechanik ein, die notwendig geworden ist, um die bestehende „*Nearness-Diagram*“-Methode auf die spezielle Form des Bremer Autonomen Rollstuhls *Rolland* umzusetzen. Ein Vergleich der Original-Methode mit der Erweiterung wird in diesem Abschnitt aufgenommen und erläutert.
Im letzten Schritt erfolgt eine Beschreibung der direkten Bewegungsbefehle an die Entwicklungsumgebung, die die kollisionsfreie Navigation schließlich durchführen.

3.2.1 Wahrnehmung der Umgebung

In der Wahrnehmungsphase werden alle nötigen Informationen, die durch die Sensoren geliefert werden, zusammengetragen und zur weiteren Verarbeitung aufbereitet.
Das System verfügt über eine Odometrie-Sensorik und über zwei in Bodennähe montierte Laserscanner, welche sowohl frontseitig als auch rückseitig die Umgebung wahrnehmen. Diese Sensoren genügen bereits, um eine lokale Hindernisvermeidung durchführen zu können.
Wie in Kapitel 2.3.1 beschrieben ist es zur Durchführung der Navigation mit Hilfe der „*Nearness-Diagram*“-Methode nötig, folgende Informationen in jedem Berechnungszyklus zur Verfügung zu haben:

- die Position des Rollstuhls
- die Zielposition relativ zum Rollstuhl
- die Sensorinformationen, welche die Umgebung widerspiegeln.

Im Folgenden wird auf diese Punkte eingegangen.

Rollstuhlposition

Die Informationen über die Position des Rollstuhls werden anhand der Daten aus dem Odometriesystem ermittelt. Durch Inkrementalgeber an den Fahrwerkrädern des Rollstuhls werden die Radumdrehungen während der Fahrt exakt gemessen und aufgezeichnet. Auf diese Weise ist es möglich, die vom Fahrtbeginn zurückgelegte Strecke und die aktuelle Blickrichtung genauestens zu bestimmen.

Dies geschieht systemintern anhand einer Abbildung der Position in einem zweidimensionalen kartesischen Koordinatensystem. Die Datenstruktur zur Positionserfassung enthält also stets (unter anderem) sowohl die Position des Rollstuhls, abgebildet durch einen Punkt innerhalb dieses Koordinatensystems ($P_{robot} = (x_{robot}, y_{robot})$), als auch die Ausrichtung als Richtungsangabe (aufgezeichnet in rad). Auf diese Weise kann die translationale Verschiebung des Rollstuhls mit Hilfe der Odometriemessdaten von Aufzeichnungsbeginn an bestimmt werden.

Wie jedes Odometriesystem zur Positionsbestimmung hat auch das System des Bremer Rollstuhls *Rolland* mit den der Messung inhärenten Fehleranfälligkeiten zu kämpfen. So kann es geschehen, dass die Sensoren, welche die Radumdrehungen messen, fehlerhafte Werte aufzeichnen, wenn eines der Räder aufgrund von sehr glatten oder besonders unebenen Fahrtoberflächen oder aber starken Drehbewegungen auf dem Boden „gleitet“ oder „durchdreht“. So bewegt sich zwar der Rollstuhl, ohne allerdings dass sich das Rad bewegt und somit auch der Inkrementalgeber keine Bewegung aufzeichnet, oder aber das Rad dreht auf der Stelle, ohne dass sich die Position des Rollstuhles verändert. Eine fehlerhafte Positionsberechnung ist die Folge.
Zu dieser Fehlerquelle addieren sich falsche Messwerte aufgrund ungenauer Architektur des Fahrwerks, beispielsweise unterschiedlicher Radaufhängung oder unterschiedlichen Radabstandes, ungleicher Raddurchmesser (auch bedingt durch ungleichen Luftdruck innerhalb der Räder) und verschieden starker Belastung der einzelnen Räder.

Besonders bei längeren Fahrten akkumulieren sich die kleinen Fehler in den Odometriemessungen und schwellen so leicht zu einem beträchtlichen Fehler in der Positionsbestimmung an. Das zusätzliche Auswerten weiterer Sensoren bezüglich der Position ist unabdingbar.

Zielvorgabe

Durch die Abbildung des Rollstuhls in einem zweidimensionalen Koordinatensystem lassen sich weitere Punkte innerhalb dieser Abbildung erfassen, deren Entfernung und Position relativ zur Rollstuhlposition berechnet werden können. Diesen Vorteil verwendet das System zur Zielvorgabe für die Navigation des Rollstuhls.
So lässt sich auf einfache Weise ein sogenannter „*targetrequest*“ an das Navigationsmodul übergeben, der einem Punkt $p_{goal} = (x_{goal}, y_{goal})$ in demselben Koordinationsystem entspricht. Es ist anschließend möglich, sowohl die Entfernung als auch die Ausrichtung zur Rollstuhlposition zu bestimmen.

Sensordaten

Auf eine ähnliche Weise wie die Abbildung von Rollstuhl- und Zielposition ist es auch möglich, die von den Laserscannern wahrgenommenen Hindernisse als Punktinformationen im System zu hinterlegen. Durch die bekannte Länge und Ausrichtung der Scanner und der einzelnen Scanstrahlen ist eine Berechnung der Punktkoordinaten im Koordinatensystem jedes wahrgenommenen Hindernisses möglich.
Mit diesem Hintergedanken wird zur Verwendung der „*Nearness-Diagram*“-Navigation auf das im System bereits implementierte *Evidence-Grid* zurückgegriffen. Diese Datenstruktur unterteilt die lokale Umgebung des Rollstuhls in Zellen und bildet so ein Gitter um ihn herum. Die einzelnen Zellen werden während des Systemablaufs mit den Sensordaten gefüllt und in jedem Schritt aktualisiert.
Dabei enthält jede Zelle Informationen darüber, ob ein Hindernis an dieser Stelle wahrgenommen wurde und wie lange diese Information zurückliegt, dargestellt durch eine Wertigkeit innerhalb dieser Zelle. Jener Wert gibt an, wie wahrscheinlich es ist, dass sich ein

Hindernis an dieser Stelle befindet. Mit fortlaufender Systemzeit werden die Wertigkeiten jeder Zelle aktualisiert. So wird der Wert einer belegten Zelle erniedrigt, sofern kein exakter Laserscan ein Hindernis in entsprechender Zelle detektiert. Auf der anderen Seite werden ehemals freie Zellen, die nicht mehr im Scannerbereich liegen, in ihrer Wertigkeit erhöht. Auf diese Weise ist es möglich, für jede Zelle in der lokalen Umgebung des Rollstuhls zu bestimmen, ob an der Position dieser Zelle sicherlich ein Hindernis ist, die Zelle frei von jeglichen Objekten ist oder es mehr oder weniger wahrscheinlich ist. Abbildung 3.4 stellt ein beispielhaftes *Evidence-Grid* in der lokalen Umgebung um eine Rollstuhlposition dar.

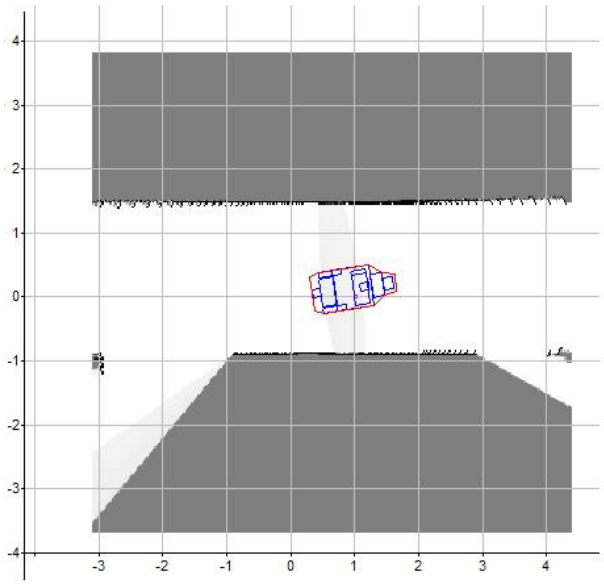

Abbildung 3.4: Das „Evidence-Grid“

Hier ist ein *Evidence-Grid* mit einer Größe von $7.5m \times 7.5m$ um die Rollstuhlposition zu erkennen. Es ist unterteilt in 300×300 Zellen. Die Wertigkeit dieser Zellen ist als Grauwert dargestellt und gliedert sich wie folgt:

- Wertigkeit 255: besetzt. An dieser Stelle befindet sich mit Sicherheit ein Hindernis. Es wurde vor sehr kurzer Zeit der Scan eines Hindernisses auf diese Zelle abgebildet.
- Wertigkeit 0: frei. Innerhalb dieser Zelle ist mit Sicherheit kein Hindernis, da die Scanner vor sehr kurzer Zeit in diesem Bereich kein Objekt wahrgenommen haben.

- Wertigkeit 128: unbekannt. Es ist nicht bekannt, ob sich an dieser Stelle ein Hindernis innerhalb der Zelle befindet, da dieser Bereich seit geraumer Zeit nicht im Sichtfeld einer der Laserscanner liegt.
- Wertigkeit 129 - 254: wahrscheinlich besetzt. An dieser Stelle hat sich vor einiger Zeit ein Hindernis befunden. Je höher die Wertigkeit, desto kürzer ist der Zeitpunkt dieser Messung her und desto wahrscheinlicher ist es, dass sich ein Hindernis an dieser Stelle befindet.
- Wertigkeit 1 - 127: wahrscheinlich unbesetzt. Hier wurde vor geraumer Zeit eine freie Fläche wahrgenommen, diese liegt aber nicht mehr im Sichtbereich der Laserscanner. Je höher der Wert, desto länger liegt die Aufnahme zurück.

Es ist lediglich nötig in jedem Systemzyklus zum einen die aktuellen Scanpunkte auf die entsprechenden Zellen abzubilden und den Wert dort zu aktualisieren und zum anderen die Wertigkeiten aller anderen Zellen dementsprechend zu verringern beziehungsweise zu erhöhen. Das *Evidence-Grid* wird in jedem Zyklus um die Rollstuhlposition erstellt, das Zentrum ist also jeweils das Achsenzentrum des Rollstuhls. So ist es möglich, bei fortlaufender Fahrt auch jene Bereiche zu modellieren, welche nicht im Sichtbereich der Laserscanner liegen (die toten Zonen an der Rollstuhlseite) und Informationen über vergangene Messungen beizubehalten.

3.2.2 Modellierung des Weltmodells

In der Modellierungsphase werden die vom Wahrnehmungsmodul bereitgestellten Daten dazu verwendet, ein internes Weltmodell abzubilden. Dieses Modell spiegelt die lokale Umgebung des Rollstuhls wider, wobei alle Information über Zielposition, Rollstuhlposition und Hindernisverteilung abgebildet werden.
Dies geschieht mit Hilfe einer speziellen Entität, dem „*Nearness-Diagram*“. Dieses enthält die oben genannten Informationen.

Aufbau der „Nearness-Diagrams“

Die Definition der „*Nearness-Diagrams*“ wurde bereits in Kapitel 2.3.3 dargelegt. Dieser Abschnitt dient zur Beschreibung der Vorgehensweise, wie die aufbereiteten Sensordaten aus der Wahrnehmungsphase dazu verwendet werden, die beiden Diagramme PND und RND aufzubauen.

In der grundlegenden Beschreibung der „*Nearness-Diagrams*“ ist bereits darauf hingewiesen worden, dass die Diagramme auf einem zweidimensionalen Arbeitsbereich W basieren, in denen die Hindernisinformationen als Punktinformationen in der Form $p_{obstacle} = (x_{obstacle}, y_{obstacle})$ abgelegt sind. Diese Grundlage entspricht dem „*Evidence-Grid*“, welches exakt diese Daten in der lokalen Umgebung des Rollstuhls beinhaltet. Da es in jedem Rechenzyklus aktualisiert wird, kann auch die Berechnung der „*Nearness-Diagrams*“ stets aktuell gehalten werden. Es muss keine weitere Anpassung erfolgen.

Der Aufbau der Diagramme geschieht somit direkt aus den Daten des *Evidence-Grid*. Da die Methode der „*Nearness-Diagram*“-Navigation die lokale Umgebung des Rollstuhls in Sektoren einteilt, ist hierfür weiterhin die Definition der Sektorenanzahl n nötig.
Basierend auf den Erfahrungen der Entwickler der Methode ([18]) ist eine Einteilung der Diagramme PND und RND in $n = 144$ Sektoren sinnvoll, wobei jeder Sektor einen Bereich von 2,5 Grad abdecken wird ($360°/n = 2,5°$, mit $n = 144$). Dies sorgt für eine ausreichend detaillierte Abbildung der Umgebung in den Diagrammen.

Anschließend ist es lediglich nötig, die wahrgenommenen Hindernispunkte auf die Sektoren abzubilden. Durch die aus den Odometriedaten bekannte Ausrichtung des Rollstuhls ist dies möglich, da somit für jede einzelne Zelle des *Evidence-Grid* die Ausrichtung relativ zur Ausrichtung des Rollstuhls bestimmt werden kann.
Abbildung 3.5 zeigt sowohl das lokale *Evidence-Grid* um die Rollstuhlposition (links), als auch die sich daraus ergebende Sektoreinteilung mit $n = 144$ (rechts).

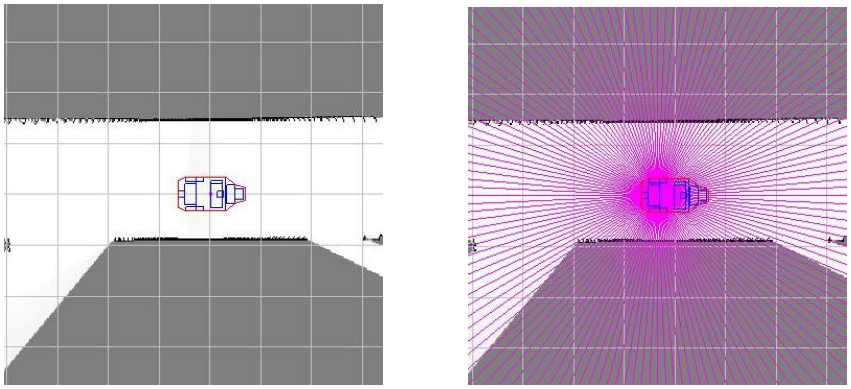

Abbildung 3.5: Links: Lokales Evidence-Grid, Rechts: Grid mit Sektoreinteilung

Anschließend lässt sich, entsprechend der Funktionsweise des Diagrammaufbaus, nun derjenige Punkt in jedem Sektor ermitteln, der die kürzeste Distanz zum Rollstuhl aufweist.

Nach der folgenden Formel lassen sich nun die Einträge im PND („*Nearness-Diagram*“ zur Rollstuhlmitte) und im RND („*Nearness-Diagram*“ zur Rollstuhlaußenseite) ermitteln.

$$
\begin{aligned}
PND : \mathbb{R}^2 \times D &\rightarrow \{\mathbb{R}^+ \cup \{0\}\}^n \\
(x, d_i) &\rightarrow \{PND_i(x, d_i)\}^n \\
if\ d_i > 0,\ PND_i(x, d_i) &= d_{max} + 2R - d_i \\
else\ PND_i(x, d_i) &= 0
\end{aligned}
$$

$$
\begin{aligned}
RND : \mathbb{R}^2 \times D &\rightarrow \{\mathbb{R}^+ \cup \{0\}\}^n \\
(x, d_i) &\rightarrow \{RND_i(x, d_i)\}^n \\
if\ d_i > 0,\ RND_i(x, d_i) &= d_{max} + E_i - d_i \\
else\ RND_i(x, d_i) &= 0
\end{aligned}
$$

wobei

- E_i der Funktion entspricht, die den Abstand vom Mittelpunkt des Rollstuhls zu dessen Außenseite im Sektor i darstellt.

- R dem inneren Radius des Rollstuhls entspricht.
- d_{max} der Breite des *Evidence-Grid* entspricht.

Besondere Beachtung fällt ab nun auf die spezielle Architektur eines Rollstuhls. Durch die ausladende Form zur Front (siehe schematische Abbildung 3.6) ist kein einheitlicher Radius zu ermitteln, so wie es bei einem kreisrunden Roboter der Fall wäre.

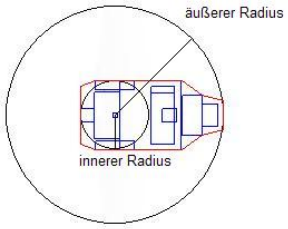

Abbildung 3.6: Innerer und äußerer Radius

Die systeminterne Klasse *robotshape* enthält Informationen über die genauen Maße des Rollstuhls wie die Form, die Achslänge, den inneren und den äußeren Radius. Die Kombination dieser Werte wird dazu verwendet, um die nötigen Informationen zur Berechnung der „*Nearness-Diagrams*“ bereitzustellen.
Das Zentrum der Diagramme liegt hierbei stets auf dem Drehpunkt des Rollstuhls. Dieser befindet sich architekturbedingt in der Mitte der hinteren Achse. Der Radius, welcher zur Berechnung des PND hinzugezogen wird, entspricht dem inneren Radius um das Zentrum des Rollstuhls (siehe Abbildung 3.6). Dies ist in sofern ausreichend, da das PND lediglich dazu verwendet wird, um die Hindernisverteilung um den Rollstuhl herum darzustellen (siehe Kapitel 2.3.3).

Der sicherheitskritische Aspekt wird mit Hilfe des RND ausgewertet, der Nähe der Hindernisse zur Rollstuhlaußenseite.
Durch die Auswertung des *robotshape* lässt sich nun für jeden Sektor der Abstand vom Zentrum des Rollstuhls zur Außenseite ermitteln (siehe Abbildung 3.7). Dies entspricht der Funktion E in der Formel zur Berechnung des RND.

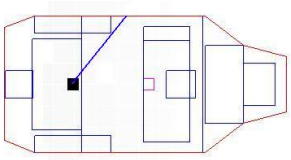

Abbildung 3.7: Distanz zur Außenseite

Anpassung der Sicherheitsdistanz

Aufgrund der speziellen Architektur des Rollstuhls muss auch eine Anpassung der Sicherheitsdistanz erfolgen [17]. Dies liegt vor allem daran, dass Drehbewegungen - besonders durch die nach vorne erweiterte Form - schnell problematisch werden können und hier besondere Vorsicht geboten ist. Abbildung 3.8 zeigt einen Vergleich der Sicherheitszonen eines runden Roboters, für welches die „*Nearness-Diagram*“-Navigation entwickelt wurde, und des Rollstuhls der Universität Bremen.

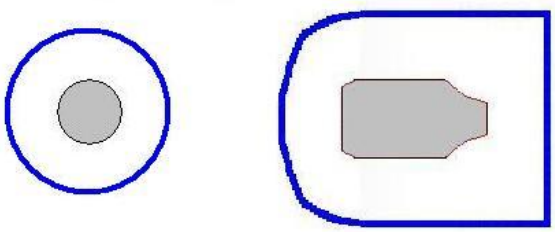

Abbildung 3.8: kreisrunde und angepasste Sicherheitszone

So ist gesichert, dass Hindernisse frühzeitig als kritisch eingestuft werden, und die Drehbewegung entsprechend früh stattfinden kann.
Durch die spezielle Form des Rollstuhls und die daraufhin angepasste Sicherheitszone ergibt sich also an dieser Stelle eine Änderung: Die Sicherheitsnähe wurde für einen runden Roboter definiert als $n_s = d_{max} - d_s$. Hierbei ist von einer kreisrunden Sicherheitszone ausgegangen worden. In Abbildung 3.9 ist zu erkennen, dass die Sicherheitszone um den

Rollstuhl uneinheitlich ist, da sie nach vorne hin erweitert wird, um entsprechend früh auf frontale Hindernisse zu reagieren.
Somit wird die Berechnung der Sicherheitsnähe n_s geändert auf die Form:

$$n_{s_i} = d_{max} - d_{s_i}$$

$$wobei\ i \in R,\ 0 \leq i < n$$

Dies entspricht der angepassten Sicherheitsnähe in Sektor i. In Abbildung 3.8 wird der Unterschied der angepassten Sicherheitsnähe und des daraus resultierenden RNDs im Vergleich zu einem runden Roboter deutlich. In der oberen Abbildung ist die Sicherheitsnähe um den kreisrunden Roboter für jeden Sektor identisch. In der unteren Abbildung ist sie durch die nach vorne erweiterte Form nicht einheitlich, was sich auch in den Werten des RNDs niederschlägt. So sind die Werte zur Front des Rollstuhls um einiges höher, als es beim runden Roboter der Fall ist, da sich hier die Umgebung näher an der Rollstuhlaußenseite befindet.

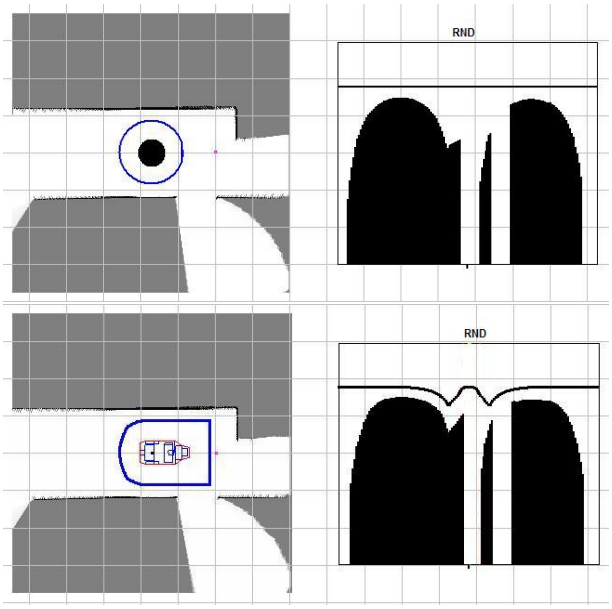

Abbildung 3.9: Angepasste Sicherheitszone mit RND

Mit Hilfe dieser Änderungen stehen am Ende der Modellierungsphase zwei Entitäten, zum einen das PND und zum anderen das RND, welche im Folgenden sowohl zur Situationsfindung als auch zur Aktionsbestimmung dienen. Des Weiteren befinden sich innerhalb dieser Diagramme alle relevanten Informationen über die lokale Umgebung des Rollstuhls. So können die Einträge jederzeit dazu verwendet werden, um genaue Entfernungen zu und zwischen den umgebenden Hindernissen zu berechnen, da ein Zurückrechnen der Informationen aus den Diagrammen in die genauen Hindernispunkte möglich ist.

Folgende Abbildung 3.10 verdeutlicht den gerade erläuterten Sachverhalt mit Hilfe einer beispielhaften Umgebung mitsamt der daraus resultierenden „*Nearness-Diagrams*“ PND und RND.

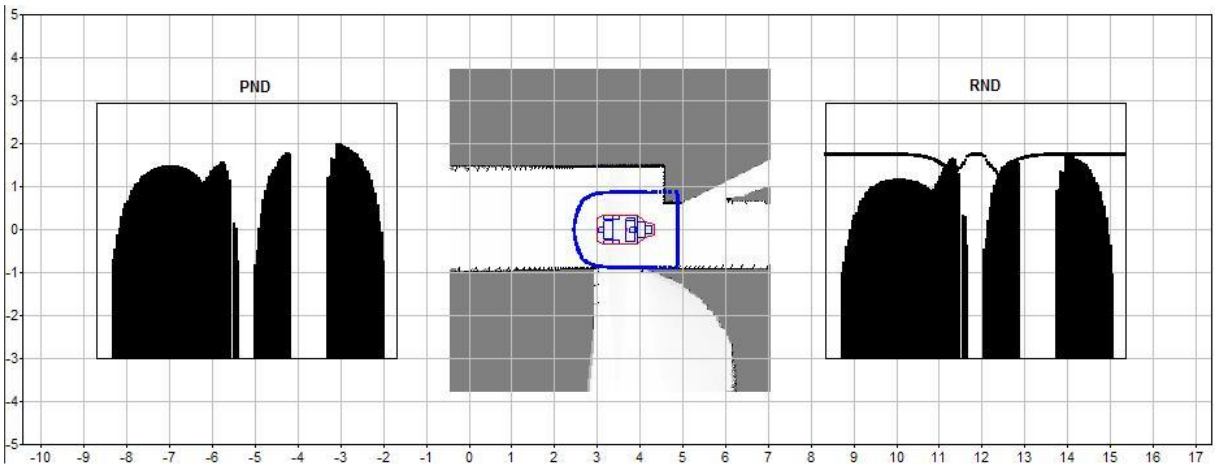

Abbildung 3.10: Beispielumgebung in der Simulation

So ist zu erkennen, dass an den enstprechenden Stellen des PND (links) und des RND (rechts) die Ausschläge die Nähe zu den umgebenden Hindernissen darstellen. Generell gilt: je höher der Ausschlag, desto näher ist das Hindernis. Freie Flächen deuten auf befahrbare Regionen hin. Die Ecke der Wand auf der linken Seite des Rollstuhls befindet sich innerhalb der Sicherheitszone. Dies ist auch dadurch zu erkennen, dass an dieser Stelle die Sicherheitsdistanz im RND überschritten wurde.

3.2.3 Berechnung der kollisionsvermeidenden Bewegung

Das Ziel der Berechnungsphase liegt darin, die Algorithmen zur kollisionsfreien Navigation auf das Weltmodell anzuwenden und auf diese Weise entsprechend auf die jeweilige Umgebung reagieren zu können. Hierzu wird auf die soeben erstellten Diagramme zurückgegriffen.

Im Zuge der „*Nearness-Diagram*“-Navigation geschieht dies in drei Schritten:

1. Aus den Informationen der Sensoren und Zielvorgaben, vorliegend in den „*Nearness-Diagrams*“ PND und RND, wird ein besonderes Element detektiert, die *free walking area*.
2. In einer weiteren Analyse der „*Nearness-Diagrams*“ wird die aktuelle Situation des Systems bestimmt.
3. Je nach Situation wird anschließend der in der Aktionsphase auszuführende Befehl errechnet. Dies schließt sowohl die Planung der Bewegungsrichtung als auch der Geschwindigkeit und der Rotationsgeschwindigkeit ein.

Am Ende dieser Phase steht also ein Bewegungsbefehl, der den Rollstuhl sicher und frei von Kollisionen zum Ziel navigieren soll.

Die ND-Analyse

In Kapitel 2 ist der grundlegende Algorithmus zur Analyse der „*Nearness-Diagrams*“ für kreisrunde Roboter vorgestellt worden. Auf eben diese Weise werden aus dem PND in entsprechender Reihenfolge

- Lücken detektiert,
- jeweils zwei Lücken zu einer Region zusammengefasst,
- eine der Regionen als *free walking area* definiert.

Abbildung 3.11 zeigt, wie sich die Analyse in der Entwicklungsumgebung vollführt.

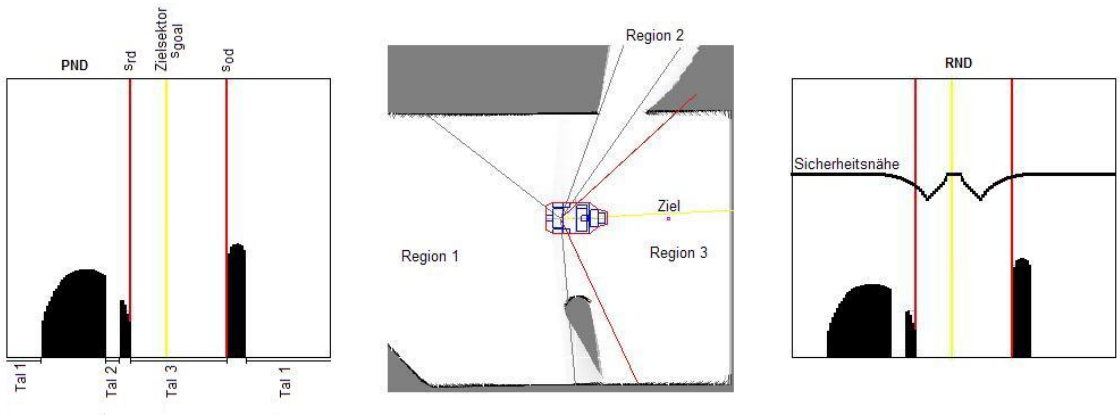

Abbildung 3.11: ND-Analyse in einer Beispielsumgebung

In dieser Abbildung sind drei Regionen zu erkennen, jeweils durch zwei Lücken (Sprünge im PND) gebildet. Schließlich ist Region 3 als *free walking area* ausgewählt worden, da sie sowohl navigierbar ist, als auch eine Seite aufweist, die eine kürzere Distanz zur Zielposition hat als die anderen.

Eine Anpassung der Funktionsweise aufgrund der speziellen Form des Rollstuhls muss nicht erfolgen. Zur Detektion von Unstetigkeiten in den Diagrammen wird hierbei die Breite des Rollstuhls herangezogen, da nur solche Regionen interessant sind, durch die der Rollstuhl navigieren kann.

Navigierbarkeit einer Region

In diesem Abschnitt soll der Algorithmus erläutert werden, der verwendet wird, um eine Region auf ihre Navigierbarkeit hin zu untersuchen. Es wurde hierbei die Methode nach [18] implementiert.

Der Algorithmus überprüft, ob für eine gegebene Ziel- und Roboterposition (x_{goal} und x_{robot}) ein passierbarer Pfad zwischen umgebenden Objekten (abgelegt als Hindernispunkte in einer Liste L) existiert. Hierzu wird die Umgebung in vier Bereiche unterteilt (FL: Front-links, FR: Front-rechts, BL: Hinten-links, BR: Hinten-rechts), welche von der Linie P zwischen Ziel- und Roboterposition abhängig ist (siehe Abbildung 3.12). Somit kann jeder Hindernispunkt x aus L einem Bereich zugewiesen werden.

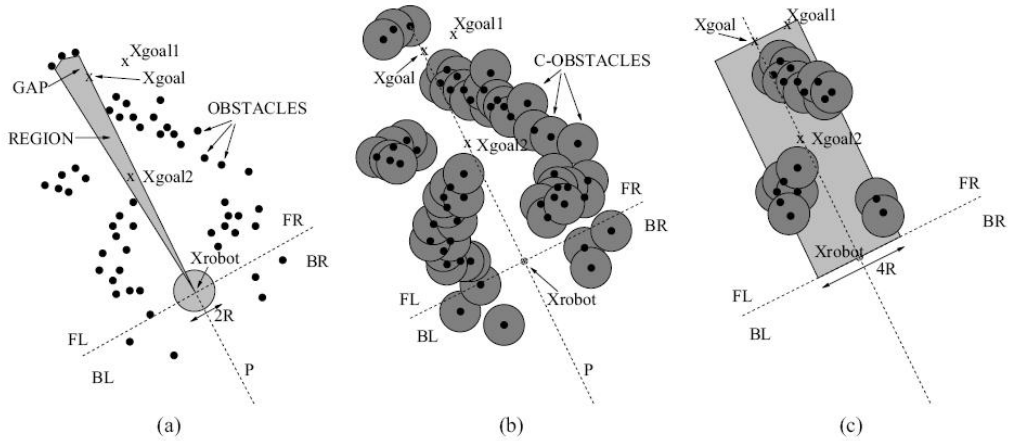

Abbildung 3.12: Algorithmus zur Navigierbarkeit einer Region [18]

In einer ersten Überprüfung wird festgestellt, ob es möglich ist die Zielposition x_{goal} überhaupt anzufahren. Liegt ein Hindernispunkt aus L näher am Zielpunkt, als der Roboter breit (R) ist, kann das Ziel nicht erreicht werden. Wenn also gilt

$$\exists i \mid d(x_{goal}, x_i) < R$$

dann ist die Region nicht navigierbar.
Sollte dies nicht der Fall sein, wo werden nur die Hindernispunkte aus L betrachtet, die folgende Eigenschaften erfüllen:

- Sie liegen zwischen dem Roboter und dem Ziel

$$x_i \in FL \; oder \; x_i \in FR$$

und außerdem

$$d(x_i, x_{robot}) \leq d(x_{goal}, x_{robot})$$

- Sie befinden sich in einem Korridor der Breite $4R$ um die Linie P zwischen Roboter und Ziel (siehe Abbildung 3.12 c)

$$d(x_i, P) \leq 2R$$

Das Ziel ist dann erreichbar, sofern für alle restlichen Punkte mit den obigen Eigenschaften gilt:

$$d(x_i, x_j) > 2R$$

$$wobei\ x_i \in FL\ und\ x_j \in FR$$

So ist gesichert, dass zwischen allen Hindernissen der linken und der rechten Seite jeweils genug Raum ist, damit der Roboter hindurch fahren kann.

Die Methode ist direkt anwendbar auf Situationen, in denen das Ziel innerhalb der untersuchten Region liegt (x_{goal2} in Abbildung 3.12). Dies entspricht den Situationen HSGR, LS2 und LS1GR (letztere wird später eingeführt).
Alle anderen Zustände erfordern eine Änderung, da hier die Zielposition außerhalb der Region liegt (x_{goal1}). So wird in diesem Fall überprüft, ob eine Art Landmarke (x_{goal}) auf dem Weg zum Ziel angesteuert werden kann. x_{goal} entspricht dabei der Mitte der Lücke zwischen den Hindernissen, die diese Grenze der Region (s_{rd}) definieren (siehe Abbildung 3.12).

Für nicht kreisrunde Roboter, wie einen Rollstuhl, ist dieser Algorithmus nur bedingt aussagekräftig, da nicht jede Region passierbar ist, durch die ein Rollstuhl mit seiner Breite navigieren kann. Aber es lassen sich solche Regionen ausschließen, die ganz sicher nicht befahren werden können, was an dieser Stelle als ausreichend angenommen wird.

Situationsfindung auf Basis der „Nearness-Diagrams“

Der zweite Schritt innerhalb der Planungsphase, nachdem die Diagramme auf das Vorhandensein einer *free walking area* überprüft wurden, ist die Bestimmung des eindeutigen Systemzustandes, der Situation, auf die entsprechend reagiert werden muss.

In Kapitel 2 sind die Grundlagen zur Situationsfindung auf Basis der „*Nearness-Diagram*“-Methode dargelegt worden. Die folgende Abbildung 3.13 zeigt den damit verbundenen Entscheidungsbaum in der Originalentwicklung. Im Anschluss daran erfolgt eine Betrachtung der einzelnen Situationen in Bezug auf kritische Planungsabläufe, welche durch die Form des Rollstuhls einer Anpassung bedürfen.

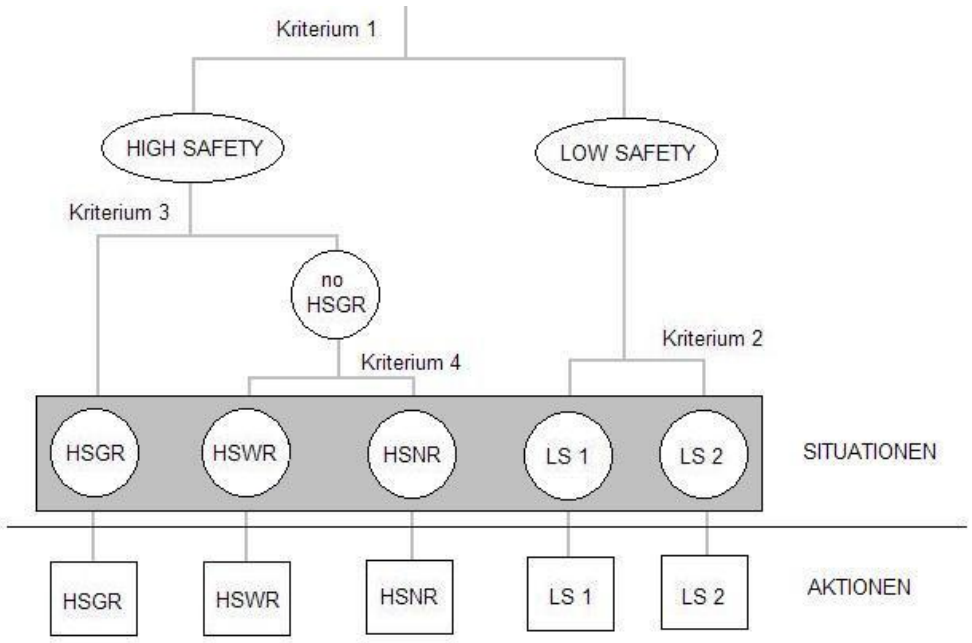

Abbildung 3.13: ND-Entscheidungsbaum

Kriterium 1: Sicherheit der Umgebung

Die Auswertung des ersten Kriteriums erfolgt nach der im Grundlagenteil erläuterte Methode (siehe Abschnitt 2.3.6). So werden die Einträge des RNDs daraufhin überprüft, ob

sie die Sicherheitsdistanz überschreiten.
Sollten ein oder mehrere Sektoren die Sicherheitsnähe überschreiten, so befindet sich das System im Zustand der niedrigen Sicherheit. Anderenfalls ergibt sich eine hohe Sicherheit als Situation.

Kriterium 2 - Hindernislage in der Sicherheitszone

Eine Überprüfung der Sektoren, welche die Sicherheitsnähe im RND überschreiten, führt zur Auswahl der Situationen Low Safety 1 (auf einer Seite der Zielseite der *free walking area* befindet sich ein Hindernis) oder Low Safety 2 (auf beiden Seiten befinden sich Hindernisse innerhalb der Sicherheitszone).
In Abbildung 3.14 (oben) führt dies zur Detektion des Zustandes LS 1, in Abbildung 3.14 (unten) entsprechend zur Situation LS 2.

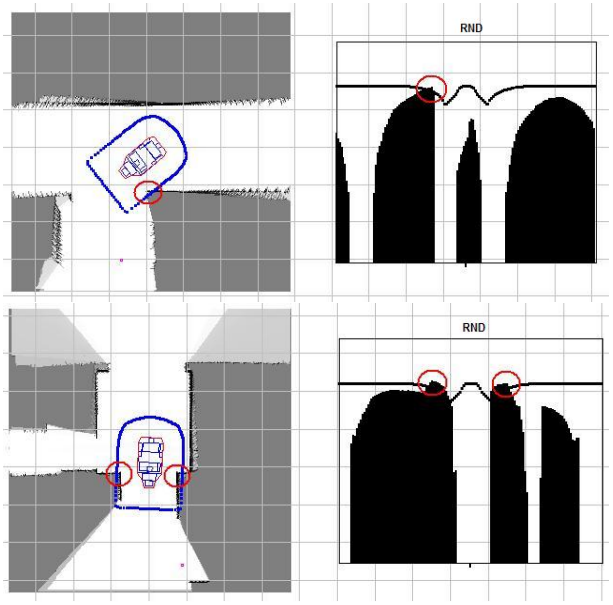

Abbildung 3.14: RND-Analyse der Hindernislage - LS1 (oben), LS2 (unten)

Kriterium 3 - Ziellage zur free walking area

Die Lage des Zielsektors s_{goal} gibt Auskunft darüber, ob sich das Ziel innerhalb der aus-

gewählten *free walking area* befindet (HSGR - High Safety Goal in Region, siehe Abbildung 3.15) oder ob das letzte Kriterium ausgewertet werden muss.

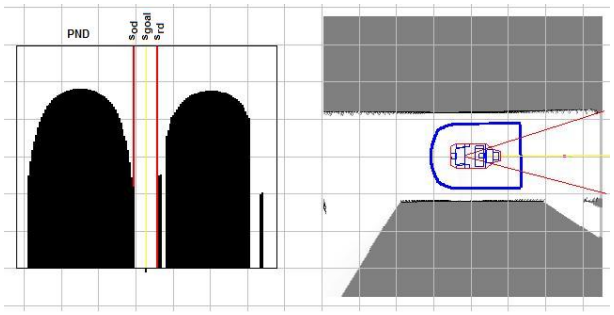

Abbildung 3.15: PND-Analyse der Zielposition

Beschaffenheit der free walking area

Eine Berechnung der Breite der *free walking area* muss erfolgen, wenn das vorgegebene Ziel nicht innerhalb dieser Region liegt. Da die Anzahl der regionbildenden Sektoren direkten Aufschluss über ihre Breite liefert, wird auf diese Weise bestimmt, ob die Situation als HSWR (weite *free walking area*) oder HSNR (enge *free walking area*) definiert wird. Abbildung 3.16 verdeutlicht diesen Sachverhalt.

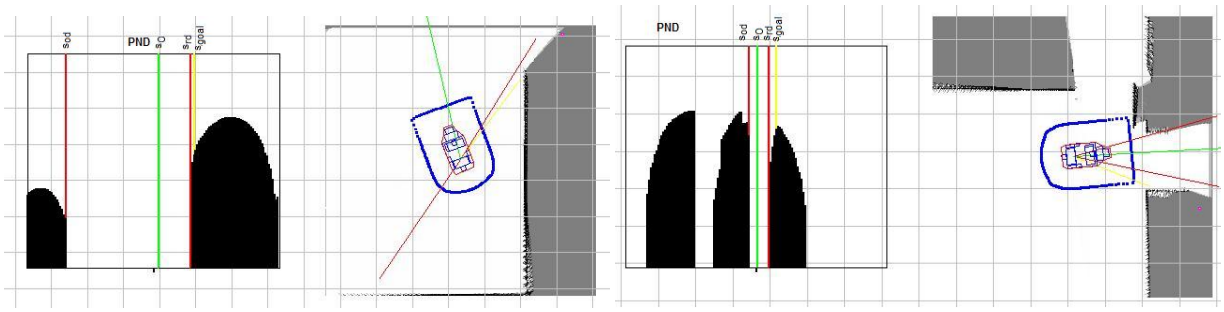

Abbildung 3.16: PND-Analyse: Beschaffenheit der *free walking area* - HSWR (links), HSNR (rechts)

Am Ende der Auswertung der soeben beschriebenen Kriterien steht also der Systemzustand fest und kann als einer der fünf Situationen HSGR, HSWR, HSNR, LS1 oder LS2 benannt werden.

Anpassungen der Planungsphase

An die Situationsbestimmung schließt sich nach [18] die Berechnung der kollisionsvermeidenden Bewegung an, das heißt die Bestimmung der entsprechenden Aktion auf Basis der detektierten Situation (siehe auch Kapitel 2).

Durch die spezielle Form des Rollstuhls ist es allerdings nötig, die Situationen differenzierter zu betrachten als es in der grundlegenden Entwicklung der Fall ist. Die Änderungen innerhalb dieser Phase lassen sich in folgenden Punkten zusammenfassen:

- Differenzieren der „Low Safety 1“-Situationen
- Erzwingen einer Ausschwenkbewegung beim Hineinfahren in enge Bereiche
- Überprüfen der *free walking area* und ihrer Umgebung auf kritische Sektoren
- Überprüfen der errechneten Bewegungsrichtung auf Machbarkeit und Einfügen von Rückwärtsfahrverhalten

Diese Änderungen beziehen sich auf die Planung der Bewegungsrichtung und adaptieren die der originalen „*Nearness-Diagram*“-Navigation zugrunde liegende Mechanik, da dort weder explizites Ausschwenken noch rückwärtige Navigation implementiert wurde.

Differenzieren der Low Safety 1 - Situationen

Die erste Anpassung innerhalb der Planungsphase betrifft die Funktionsweise der Low Safety 1-Situation. Abschnitt 2.3.3 erläutert, dass der Plan dieser Situation vorsieht, die Sicherheitszone von Hindernissen zu befreien und dabei auf das Ziel (impliziert vorgegeben durch die „Zielseite“ der *free walking area*, s_{rd}) zuzusteuern. Die Befreiung der Sicherheitszone von Objekten ist allerdings Hauptziel dieser Situation.
In Bezug auf die Rollstuhlform ist es allerdings nötig, vorausschauender zu fahren, beispielsweise die Vorbereitung einer Ausschwenkbewegung.
Eine differenziertere Betrachtungsweise dieser Situation erscheint somit sinnvoll, um zum einen keine unnötigen Hindernisvermeidungen durchzuführen und zum anderen die Be-

schaffenheit der Umgebung näher zu betrachten, als lediglich auf Hindernisposition und Lage des s_{rd} einzugehen.

Abbildung 3.17 erweitert die detektierte LS1-Situation, indem im anschließenden Schritt eine Analyse ähnlich der High-Safety-Situationen durchgeführt wird.

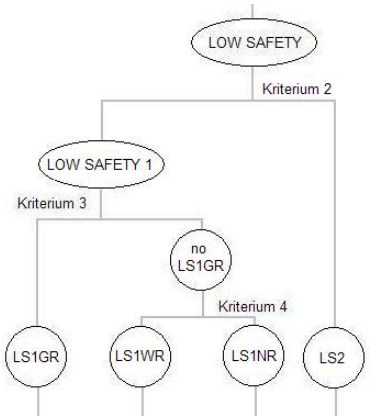

Abbildung 3.17: Differenzirung der LS1-Situation

Dies beinhaltet die Einführung dreier neuen Situationen:

- Low Safety 1 Goal in Region (LS1GR) : Es befindet sich ein Hindernis auf einer Seite der Zielseite der *free walking area* innerhalb der Sicherheitszone, gleichzeitig ist die Zielposition auch innerhalb dieser Region.
- Low Safety 1 Narrow Region (LS1NR) : Die anzusteuernde Region ist eng, das heißt ihre Breite unterschreitet eine bestimmte Grenze, während sich ein Hindernis in der Sicherheitszone befindet.
- Low Safety 1 Wide Region (LS1WR) : Auch hier befindet sich ein Hindernis nah an der Rollstuhlaußenseite, die *free walking area* ist allerdings breit.

Mit Hilfe dieser Aufteilung kann nun ein Bewegungsbefehl analog der Situationen HSGR, HSWR und HSNR berechnet werden. Eine Hindernisvermeidung muss lediglich durchgeführt werden, wenn der daraus resultierende Befehl eine Navigation vorsehen würde, die

in Richtung des Hindernisses erfolgt. Der Vorteil dieser Methode wird bei der Betrachtung folgenden Szenarios deutlich: Während der Fahrt wird der Rollstuhl von einer Person begleitet, die sich stets im linken hinteren Teil der Sicherheitszone befindet, somit immer ein „Hindernis“ darstellt. In Abbildung 3.18 wird dies deutlich. Die Original-Umsetzung der „*Nearness-Diagram*“-Navigation würde die Person immer in die Berechnung der Bewegungsrichtung einfließen lassen. Da sie sich allerdings nicht „im Weg“ befindet, muss dieses Verhalten nicht sein.

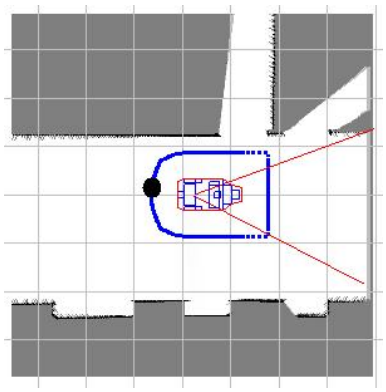

Abbildung 3.18: Beispiel eines unkritischen Hindernisses innerhalb der Sicherheitszone

Eine Ausweichbewegung muss erst erfolgen, wenn diese Person aufgrund der berechneten Bewegung zu einem Hindernis in der Bewegungsrichtung wird.

Erzwingen einer Ausschwenkbewegung in enge Bereiche

Die mit Abstand signifikanteste Veränderung der Bewegungsmechanik ist die Einführung einer speziellen Ausschwenkbewegung. Die folgende Abbildung verdeutlicht, warum ein Roboter, dessen Form nicht kreisrund ist, nicht auf die gleiche Weise navigieren kann wie ein runder Roboter und somit eine Ausschwenkbewegung nötig ist.

Obwohl beide Roboter in Abbildung 3.19 die gleiche Breite haben, wird eine unterschiedliche Fahrmechanik verlangt. Die Lösung der abgebildeten Problematik ist bei der Navigati-

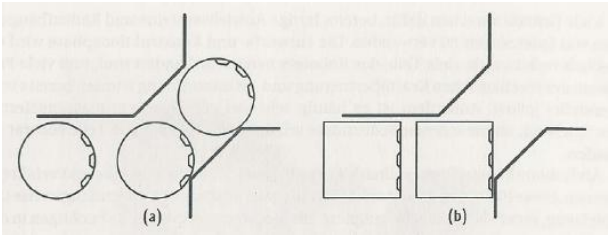

Abbildung 3.19: Vergleich Fahrverhalten: kreisrund / rechteckig, [8]

on eines Rollstuhls nur durch eine Ausschwenkbewegung zu erreichen, damit das Fahrzeug frontal in die Verengung hineinfahren kann. Folgende Abbildungen (3.20) zeigen beispielhaft, in welchen Fällen eine Ausschwenkbewegung angebracht beziehungsweise nötig ist.

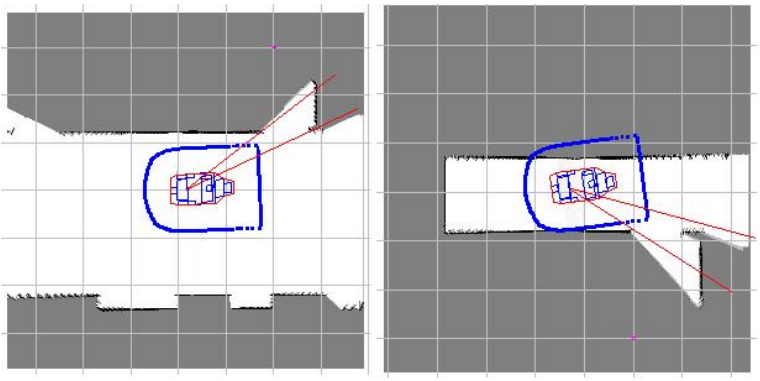

Abbildung 3.20: Situationen mit nötiger Ausschwenkbewegung

Eine Anpassung der Mechanik muss also immer dann erfolgen, wenn in enge Bereiche navigiert wird. Dies wird innerhalb der „*Nearness-Diagram*“-Mechanik immer dann der Fall sein, wenn die *free walking area* eine bestimmte Breite unterschreitet. Es betrifft die Situationen: HSGR, HSNR, LS1GR, LS1NR und LS2. Lediglich die Definition der Situationen HSWR und der soeben eingeführten Situation LS1WR weist implizit daraufhin, dass die *free walking area* nicht eng ist (siehe Abschnitt 2.3.6). Somit ist die Einführung eines weiteren Kriteriums innerhalb des Entscheidungsbaumes nötig. Ein in diesem Zusammenhang geänderter Entscheidungsbaum ist in Abbildung 3.21 zu erkennen. Hier wurde nach jeder der soeben genannten Situationen ein weiteres Kriterium eingeführt, mit welchem

die Notwendigkeit einer Ausschwenkbewegung überprüft werden soll.

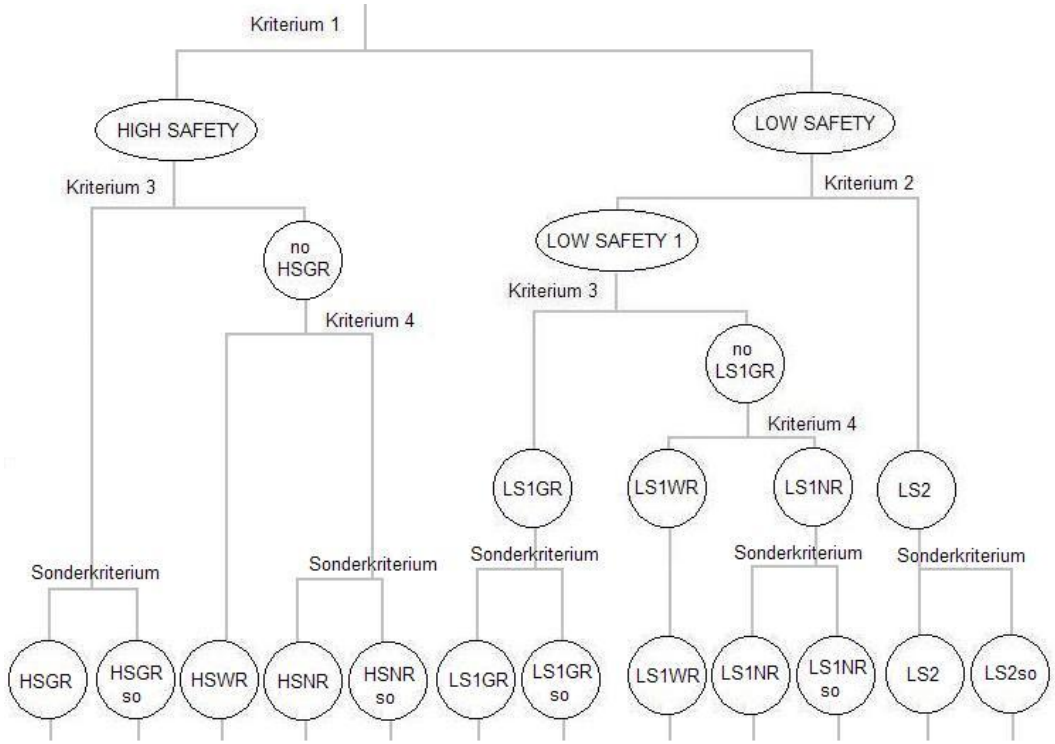

Abbildung 3.21: Erweiterter ND-Entscheidungsbaum

Sonderkriterium: Enge der free walking area

Die Auswertung dieses Kriteriums zeigt an, ob eine Ausschwenkbewegung nötig ist, da in einen schwierigen Bereich navigiert werden soll. Die Grundlage dafür ist zum einen die Breite der *free walking area*, die durch die Anzahl ihrer Sektoren bestimmt werden kann. Zum anderen wird die perspektivische Breite berechnet, um zu überprüfen, ob der Rollstuhl durch die Enge auch ohne eine Ausschwenkbewegung navigieren kann. Ein letztes Kriterium ist die effektive Breite der *free walking area*, also die Strecke, die sich durch die beiden Hindernispunkte ergibt, welche die Region aufspannen. Die folgende Abbildung 3.22 verdeutlicht die drei Kriterien.

In Abbildung 3.22 ist zu erkennen, dass in diesem Fall eine Ausschwenkbewegung erfolgen

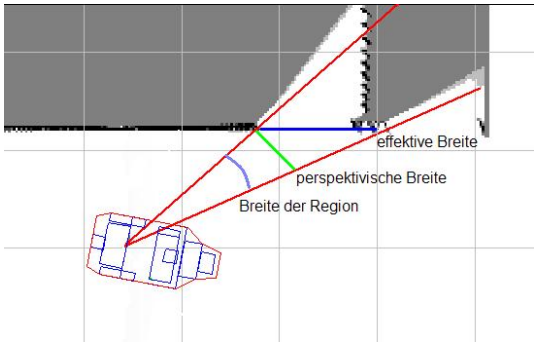

Abbildung 3.22: Ausschwenkkriterien

muss, da die drei Kriterien nicht erfüllt sind. So ist die *free walking area* sowohl in ihrer Sektorbreite als auch in ihrer perspektivischen und effektiven Breite zu schmal, als dass der Rollstuhl ohne spezielle Bewegung hindurch navigieren könnte. Die Originalmechanik der „*Nearness-Diagram*"-Methode (siehe Abschnitt 2.3.6) würde den Roboter zu diesem Zeitpunkt mittig in die Region hineinlenken, da das Fahrverhalten dieses aufgrund der Situation HSNR vorsieht. Die entsprechende Lenkbewegung würde erst erfolgen, wenn Teile der Wand in der Sicherheitszone erscheinen und die Situation von HSNR in die entsprechende LS-Situation wechseln würde. Ein Reagieren zu einem so späten Zeitpunkt macht die Bewegung unnötig kompliziert, beziehungsweise in bestimmten Umgebungen unmöglich und muss bereits früher erfolgen.

In allen Situationen, in denen das Sonderkriterium „Enge der *free walking area*" positiv ausgewertet wird, muss die originale Planung der Bewegung überschrieben werden. Es muss neben dem Umgehen möglicher Hindernisse eine Bewegung erzeugt werden, welche nicht direkt auf die Zielseite der *free walking area* (s_{rd}) steuert, sondern diese in einer Bogenform umfährt, um ein Hereinfahren in diesen engen Bereich zu erleichtern.

Überprüfung der free walking area und ihrer Umgebung auf kritische Sektoren

Im Hinblick auf die Mechanik des Ausschwenkmanövers ist es nötig, bedingt durch die Funktionsweise der „*Nearness-Diagram*"-Methode, die als *free walking area* ausgewähl-

te Region ein weiteres Mal zu überprüfen. Hier wird einer der Vorteile der „*Nearness-Diagram*“-Navigation (siehe Abschnitt 2.3.7) zum Nachteil. So werden die Grenzen der Regionen durch detektierte Sprünge innerhalb des PNDs gebildet. Ist der Sprung nicht groß genug - es wird die Rollstuhlbreite als Parameter verwendet - wird an entsprechender Stelle keine Grenze der Region gefunden. Die folgende Abbildung macht dies deutlich (3.23).

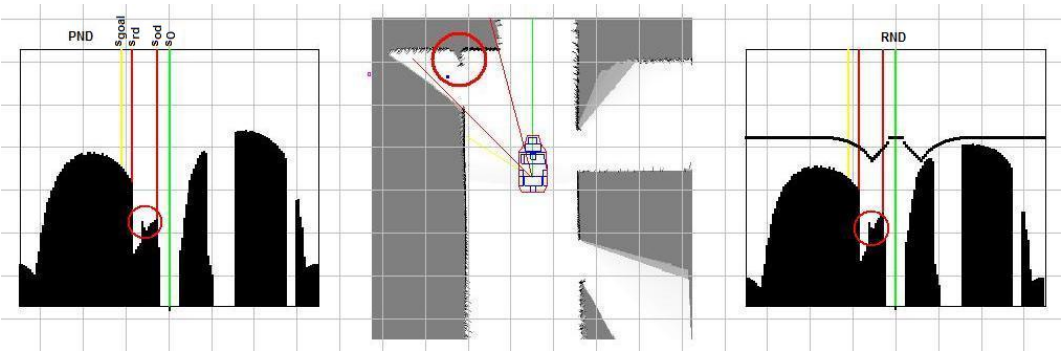

Abbildung 3.23: Kritischer Sektor innerhalb des PND

Hierbei wird die rechte Grenze der Region durch die linke steigende Unstetigkeit im PND (links) ausgelöst, welche durch den Sprung von Türzarge zur nächsten Wand entsteht. Die andere Seite der Tür wird nicht als Grenze erkannt, da es keine weitere Unstetigkeit an dieser Stelle gibt, die groß genug ist, um als Lücke erkannt zu werden - die Werte benachbarter Sektoren des PND in diesem Bereich ist geringer als die Breite des Rollstuhls ($2 * R$). Erst die rechte fallende Unstetigkeit wird als zweite Grenze der Region erkannt. Es ist also nötig, die erkannte *free walking area* auf eben solch ein Kriterium hin zu untersuchen und somit zu erkennen, ob eine Ausschwenkbewegung nötig ist.

Des Weiteren entspricht die *free walking area* nicht immer einer ausreichenden Darstellung der Umgebung des Rollstuhls. In Abbildung 3.24 wird dies deutlich. Hier ist zu erkennen, dass die ausgewählte *free walking area* nicht durch die beiden sich gegenüberliegenden Türzargen definiert wird, sondern durch einen Sprung im PND innerhalb der Tür. Eine

Ausscherbewegung wäre allerdings dennoch sinnvoll und kann nur erkannt werden, wenn auch die andere Seite der Tür mit in die Berechnung einfließt.

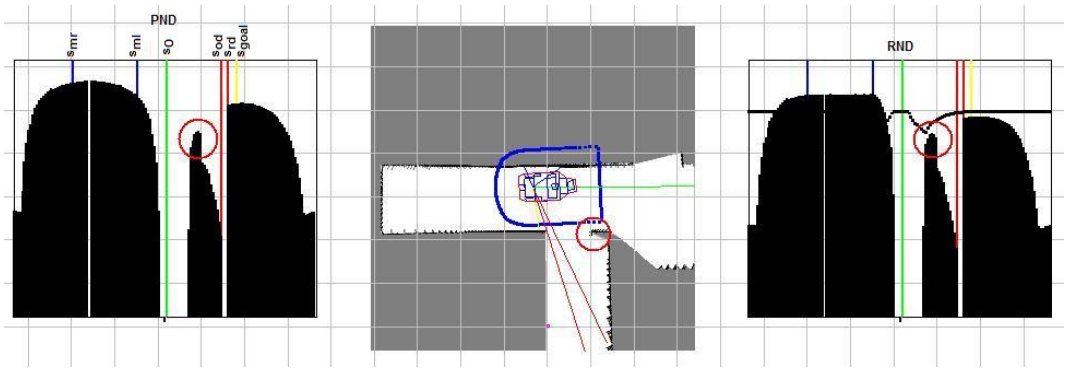

Abbildung 3.24: Kritischer Sektor in der Umgebung der FWA

Die Untersuchung erfolgt auf Basis der im PND abgelegten Hindernis-Informationen. So ist es möglich, die Abstände der Hindernispunkte in den einzelnen Sektoren zu berechnen. Auf diese Weise lässt sich eine „Verengung“ der Umgebung erkennen. In Abbildung 3.23 ist der Hindernispunkt der rechten Türzarge sehr viel näher am Hindernispunkt, welcher die linke Seite der *free walking area* definiert, als die anderen Punkte innerhalb der Region. Eine Prüfung, ob ein Ausscheren nötig ist, muss also auf Grundlage des Sektors dieses Punktes erfolgen und nicht mit der realen anderen Grenze der Region. Somit erhält die *free walking area* in diesem Fall eine neue rechte Grenze, den detektierten kritischen Sektor. Es wird dabei nicht nur der Bereich der *free walking area* untersucht, sondern ein vergrößerter Bereich von 45 Grad, damit auch jene Verengungen außerhalb der Region (siehe Abbildung 3.24) gefunden werden.

Ausschwenkbewegungen in Richtung eines Hindernisses

Aufgrund der geänderten Geometrie in Hinblick auf die Ausscherbewegung beim Hineinfahren in einen engen Bereich (siehe oben), kann es in den Situationen LS1GRso, LS1NRso und LS2so geschehen, dass die Ausschwenkrichtung (s_θ) ebenfalls der Richtung des nächsten Hindernisses innerhalb der Sicherheitszone (s_{ml} beziehungsweise s_{mr}) entspricht oder

in die gleiche Richtung führt. Dies geschieht häufig beim Einlenken in eine Tür innerhalb eines engen Korridors, wie es in Abbildung 3.25 deutlich wird.

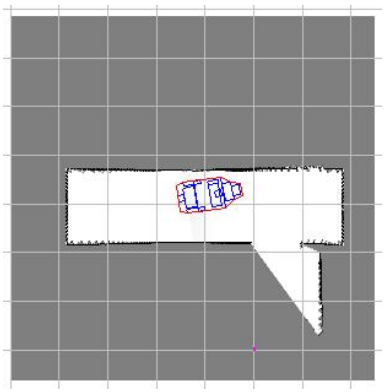

Abbildung 3.25: Beispiel einer Ausscherbewegung in engen Bereichen

Die erweiterte Bewegungsvorschrift verlangt hierbei eine Ausschwenkbewegung, da sonst nicht in die Tür eingelenkt werden kann. Die ursprüngliche Entwicklung der LS2-Situation sieht es allerdings vor, den Roboter auf gleichem Abstand zu den beiden nächsten Hindernissen zu halten (siehe auch Abschnitt 2.3.2).
In solchen Fällen wird als Erweiterung zur Originalentwicklung ein so genannter *safety drive by sector* ermittelt. Dieser Sektor hält die minimal zu tolerierende Distanz zum nächsten Hindernis ein, führt aber die dabei maximal erlaubte Ausschwenkbewegung aus. Dieser Sektor hängt direkt von der Rollstuhlbreite (dargestellt durch seinen Radius R) ab und ermöglicht, dass sehr nah am Hindernis vorbei gesteuert wird, ohne es dabei zu berühren. Er wird wie folgt definiert:

$$s_{safetydriveby} = s_{ml} \pm \alpha$$

$$\alpha = asin(\frac{R + minDist}{D_{ml}})$$

Neben dem inneren Radius des Rollstuhls (R) entspricht hierbei s_{ml} dem Sektor mit dem nächsten Hindernis innerhalb der Sicherheitszone (in dessen Richtung die Ausschwenkbewegung erfolgen soll) und D_{ml} der Entfernung von Rollstuhlposition zum Hindernispunkt

im Sektor s_{ml}. Der Parameter *minDist* entspricht einem minimalen Abstand zur Außenseite des Rollstuhls, um eine Kollision zu vermeiden und ein anschließendes Einlenken in die Verengung zu ermöglichen. Abbildung 3.26 verdeutlicht diese Berechnung.

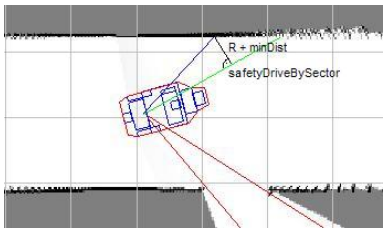

Abbildung 3.26: Berechnung „safety drive by sector“

Berechnung der Bewegungen

Im Anschluss an die Feststellung der Situation, in dem sich das System befindet, muss nun die Berechnung des entsprechenden Bewegungsbefehles erfolgen. In Abschnitt 2.3.6 wurden die Berechnungsvorschriften für die grundlegenden Situationen in Bezug auf einen kreisrunden Roboter nach [18] geliefert. Eine Anpassung dieser Vorschriften in Hinblick auf die Änderungen an der „*Nearness-Diagram*“-Navigation ist nötig und soll in diesem Abschnitt vorgenommen werden.

Am Ende dieser Phase muss die geplante Bewegung feststehen. Diese wird durch einen errechneten Bewegungsbefehl dargestellt, welcher sich in dem Tripel aus Bewegungsrichtung θ, Bewegungsgeschwindigkeit v_m und Rotationsgeschwindigkeit w zusammensetzt.

Bewegungsrichtung θ

Die Bewegungsrichtung wird in jeder Situation durch die Berechnung eines Sektors realisiert, der angesteuert wird. Die Rechenvorschriften werden auf Basis der originalen Vorschriften nach [18] auch für die in diesem Abschnitt neu definierten Situationen entwickelt.

1. *High Safety Goal in Region (HSGR)*: Wie in der originalen Entwicklung entspricht der Richtungssektor hierbei dem Zielsektor.

$$s_\theta = s_{goal}$$

2. *High Safety Goal in Region - sheer out (HSGRso)*: Eine Überprüfung der Region und der Hindernisverteilung ergab, dass ein Hineinfahren in den Bereich nur mit einer Ausschwenkbewegung sinnvoll ist. Aus diesem Grund wird die nähere Grenze (s_{rd} oder s_{od}) umfahren, indem ein fester Winkel auf den Sektor addiert wird. Dieses Verhalten führt dazu, dass nicht direkt auf die Region zugesteuert wird, sondern auf einer Kreisbahn umfahren wird, bis die Schneise groß genug ist, um hineinzulenken.

$$s_\theta = s_{closergap} \pm s_{max}$$

   wobei

   - $s_{closergap}$ ist die näher gelegene Grenze der Region (s_{rd} oder s_{od})
   - s_{max} ist der Ausschwenkwinkel und entspricht 90 Grad ($s_{max} = n/4$). Auf diese Weise wird die nähere Seite der Region auf einer Kreisbahn umfahren.

3. *High Safety Wide Region (HSWR)*: Eine Navigation in Richtung des Zieles wird mit dem Umfahren des Hindernisses verbunden, welches die Region definiert. Hierzu wird basierend auf der ursprünglichen Entwicklung nach [18] ein fester Winkel auf den entsprechenden Sektor addiert.

$$s_\theta = s_{rd} \pm \frac{s_{max}}{2}$$

4. *High Safety Narrow Region (HSNR):* Es muss keine besondere Vorsicht vor behindernden Objekten genommen werden und es wird mittig in die Schneise hinein navigiert.

$$s_\theta = \frac{s_{rd} + s_{od}}{2}$$

5. *High Safety Narrow Region - sheer out (HSNRso)*: Durch die Eigenschaften der *free walking area* ist hier eine Ausscherbewegung angebracht. Entsprechend HSGRso wird

auch hier auf die nähere Grenze der Region ein Winkel addiert, um dies zu realisieren.

$$s_\theta = s_{closergap} \pm s_{max}$$

6. *Low Safety 1 Goal in Region (LS1GR)*: Die Berechnung der Bewegungsrichtung entspricht der Situation HSGR. Hierbei wird anschließend überprüft, ob diese Bewegung in Richtung des nächsten Hindernisses führen würde (abgelegt im Sektor s_{ml}) und in eben diesem Fall am Hindernis vorbei navigiert.

   $$s_\theta = s_{goal}$$

   Sollte dies in der Richtung von s_{ml} liegen, regelt der Parameter γ das Umfahren des Hindernisses entsprechend dessen Entfernung:

   $$s_{\theta_{new}} = s_\theta \pm \gamma$$

   $$\gamma = \frac{D_s - D_{s_{ml}}}{D_s} * |(\pi + s_{ml}) - s_\theta|$$

   wobei

   - D_s entspricht der Sicherheitsdistanz
   - s_{ml} ist der Sektor mit dem nächsten Hindernis (welches die LS1-Situation ausgelöst hat)
   - $D_{s_{ml}}$ ist die Distanz zum Hindernis in Sektor s_{ml}

7. *Low Safety 1 Goal in Region - sheer out (LS1GRso)*: Auch in dieser Situation wird eingehend eine Ausschwenkrichtung errechnet. Sollte diese Bewegung in die Richtung des nächsten Hindernisses innerhalb der Sicherheitszone (s_{ml}) führen, so wird hier der weiter oben eingeführte $s_{safetydiveby}$ verwendet, um die maximal erlaubte Ausschwenkbewegung durchzuführen und dabei eine Kollision mit dem Hindernis zu vermeiden.

   $$s_\theta = s_{closergap} \pm s_{max}$$

   Solle dies der Richtung des nächsten Hindernisses entsprechen:

   $$s_\theta = s_{ml} \pm s_{safetydriveby}$$

8. *Low Safety 1 Wide Region (LS1WR)*: Ein Umfahren der Zielseite der *free walking area* realisiert die Navigation auf das Ziel zu. Die Prüfung, ob die Bewegungsrichtung hierbei der Hindernisrichtung entspricht, muss auch hier erfolgen.

$$s_\theta = s_{rd} \pm \frac{s_{max}}{2}$$

Sollte dies in der Richtung von s_{ml} liegen, regelt der Parameter γ das Umfahren des Hindernisses entsprechend dessen Entfernung:

$$s_{\theta_{new}} = s_\theta \pm \gamma$$

$$\gamma = \frac{D_s - D_{s_{ml}}}{D_s} * |(\pi + s_{ml}) - s_\theta|$$

9. *Low Safety 1 Narrow Region (LS1NR)*: Die Berechnung der Bewegungsrichtung geschieht analog zu HSNR, muss anschließend jedoch mit dem Hindernis abgeglichen werden.

$$s_\theta = \frac{s_{rd} + s_{od}}{2}$$

Sollte dies in der Richtung von s_{ml} liegen, regelt der Parameter γ das Umfahren des Hindernisses entsprechend dessen Entfernung:

$$s_{\theta_{new}} = s_\theta \pm \gamma$$

$$\gamma = \frac{D_s - D_{s_{ml}}}{D_s} * |(\pi + s_{ml}) - s_\theta|$$

10. *Low Safety 1 Narrow Region - sheer out (LS1NRso)*: Entsprechend der Ausschwenkvorschrift, wird hier der Sektor berechnet, der nötig ist, um die nächste Grenze der *free walking area* zu umfahren. Sollte dies der Hindernisrichtung entsprechend, muss auch hier umgelenkt werden.

$$s_\theta = s_{closergap} \pm s_{max}$$

Solle dies der Richtung des nächsten Hindernisses entsprechen:

$$s_\theta = s_{ml} \pm s_{safetydriveby}$$

11. *Low Safety 2 (LS2)*: Auf Basis der originalen Entwicklung wird der Sektor berechnet, der zwischen den beiden Hindernissen in den Sektoren s_{ml} und s_{mr} hindurchführt und anschließend mit dem Parameter c abgeglichen, um eine gleiche Entfernung zu beiden Hindernissen zu erreichen.

$$s_{med_1} = \frac{s_{m_l} + s_{m_r}}{2} \quad s_{med2} = \frac{s_{m_l} + s_{m_r} + n}{2}$$

$$if \ \ |s_{rd} - s_{med_1}| < |s_{rd} - s_{med_2}| \, then \ s_\theta = s_{med_1} \pm c$$

$$else \ s_\theta = s_{med_2} \pm c$$

12. *Low Safety 2 - sheer out (LS2so)*: Vorrangig wird hier der Sektor berechnet, der die Ausschwenkbewegung ermöglicht. Erst im zweiten Schritt wird dieser Sektor durch die Hindernisverteilung abgeändert, falls es zu einer Kollision kommen würde.

$$s_\theta = s_{closergap} \pm s_{max}$$

Solle dies der Richtung des nächsten Hindernisses entsprechen:

$$s_\theta = s_{ml} \pm s_{safetydriveby}$$

Überprüfen der Bewegungsrichtung auf Machbarkeit und Einfügen von Rückwärtsbewegungen

Im vorhergehenden Abschnitt wurde die Bewegungsrichtung zur kollisionsfreien Navigation berechnet. Diese Richtung entspricht einem Sektor innerhalb der „*Nearness-Diagrams*“ und kann auf leichte Weise in einen entsprechenden Winkel relativ zur Blickrichtung des Rollstuhles umgerechnet werden (siehe auch Abschnitt 2.3.6):

$$\theta = bisec(s_\theta) = \pi - \frac{2 * \pi}{n} * s_\theta$$

Da die originale Methode der „*Nearness-Diagram*“-Navigation allerdings auf einem kreisrunden und holonomen Roboter basiert, ist eine weitere Änderung in Bezug auf die Bewegungsrichtung und dem damit verbundenen Fahrverhalten vonnöten, was in folgender

Abbildung 3.27: Drehbewegung: runder Roboter / Rollstuhl

Abbildung deutlich wird.

So wird schnell ersichtlich, dass es für einen runden Roboter, welcher befähigt ist, sich auf der Stelle zu drehen, unerheblich ist, ob er sich in dieser Situation nach links oder nach rechts dreht. Der Rollstuhl allerdings wird bei einer Drehung nach links mit seiner Front eine Kollision mit der Wand erleben. Aufgrund der Funktionsweise der „*Nearness-Diagram*"-Navigation würde der Rollstuhl zwar nicht kollidieren, aber „festfahren". Dies liegt daran, dass der berechnete Bewegungsbefehl in der Startsituation den Rollstuhl in Richtung der Wand lenken würde (HSGR), sobald aber die Wand innerhalb der Sicherheitszone auftaucht, würde die Bewegung lauten, vom Hindernis weg zu lenken (LS1), so lange bis es nicht mehr in der Sicherheitszone ist und schließlich eine ähnliche Situation wie die Startposition erreicht wäre (wieder HSGR). Die Folge wäre ein Hin-und-Herlenken auf die Wand zu und von ihr weg. Eine Änderung ist also angebracht.

Hier wird sich der architektonische Vorteil des differentiell angetriebenen Rollstuhls zu Nutze gemacht und auf eine Rückwärtsbewegung ausgewichen. Dies würde auch einem intuitiven Fahrverhalten bei einer Drehbewegung auf ein Hindernis zu entsprechen.
In einem ersten Schritt wird also überprüft, ob die Drehbewegung zu einer Kollision führen würde. Hierzu wird der äußere Radius als Kriterium verwendet. Sollte dies der Fall sein, so wird die Drehbewegung zwar ausgeführt, gleichzeitig allerdings auch rückwärts navigiert, um den Abstand zum Hindernis zu vergrößern und somit die Drehung komplett durchführen zu können. Dieses Verhalten wird lediglich in Situationen niedriger Sicherheit (Low Safety) durchgeführt, da sich in diesen Fällen Hindernisse innerhalb der Sicher-

heitszone befinden (siehe Abbildung 3.28). Somit wird implizit die Sicherheitszone vom Hindernis befreit.

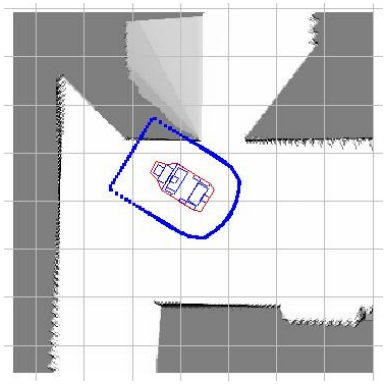

Abbildung 3.28: Drehbewegung auf ein Hindernis zu

In Low Safety 2-Situationen ist es weiterhin nötig zu überprüfen, ob die Drehbewegung nicht zu einer Kollision mit dem zweiten nächsten Hindernis innerhalb der Sicherheitszone führen würde. In solch einem Falle muss die Drehbewegung komplett ignoriert und stattdessen eine vollständige Rückwärtsbewegung eingeleitet werden. Dieses Fahrverhalten entspricht dem rückwärtigen Herausnavigieren aus einem engen Bereich (beispielsweise aus der Mitte einer Tür oder einem sehr engen Korridor), wie es in Abbildung 3.29 zu erkennen ist.

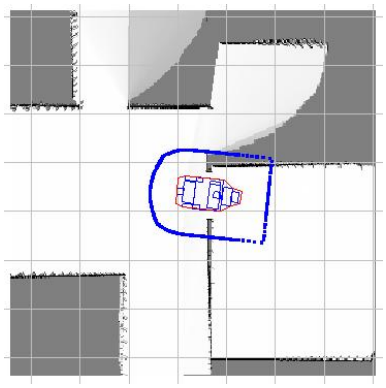

Abbildung 3.29: Rückwärtsbewegung in engen Bereichen

Bewegungsgeschwindigkeit v_m und Rotationsgeschwindigkeit w

Die Berechnung der Geschwindigkeiten von Bewegung und Rotation geschieht auf die gleiche Art und Weise, wie von [18] vorgesehen (siehe auch Abschnitt 2.3.6). So wird mit Hilfe der Funktion

$$w = w_{max} * \frac{\theta}{\frac{\pi}{2}}$$

wobei

- w_{max} die maximale Rotationsgeschwindigkeit darstellt,

basierend auf dem vorher errechneten Zielsektor s_θ und der sich daraus ergebenden Richtung θ die Rotationsgeschwindigkeit w berechnet und anschließend die erwünschte Bewegungsgeschwindigkeit v_m je nach Situation bestimmt:

1. HS-Situationen (HSGR, HSWR, HSNR):

$$v_m = v_{max} * (\frac{\frac{\pi}{2} - |\theta|}{\frac{\pi}{2}})$$

2. LS-Situationen (LS1GR, LS1WR, LS1NR, LS2):

$$v_m = v_{max} * \frac{D_{obs}}{D_s} * (\frac{\frac{\pi}{2} - |\theta|}{\frac{\pi}{2}})$$

   wobei

   - v_{max} die maximale Geschwindigkeit des Roboters sei.
   - D_s die eingehend definierte Sicherheitsdistanz sei.
   - D_{obs} die Distanz zum nächsten Hindernis innerhalb der Sicherheitszone sei.

3.2.4 Ausführung der Bewegung und Motorsteuerung

Im vorhergehenden Abschnitt wurde aufgrund der ermittelten Situation ein Bewegungsbefehl berechnet, welcher eine kollisionsfreie Navigation ermöglichen soll. Diese Berechnung

basiert auf der Entwicklung nach [18] für einen holonomen Roboter und wurde für spezielle Situationen auf die Form und Bewegungsmechanik eines Rollstuhls angepasst (Ausschwenkbewegungen in enge Bereiche, Drehbewegungen auf ein Hindernis, Rückwärtsbewegungen in engen Bereichen).
Da hier teilweise explizite Bewegungen auf ein Hindernis vorgesehen sind, ist es jedoch nötig, eine weitere Instanz als Sicherung einzufügen. Minguez schlägt in [17] die Implementierung eines „*Shape Correctors*“ vor, welcher für das System des Bremer Autonomen Rollstuhls in einer Basis-Version implementiert wurde. Die Funktionsweise soll an dieser Stelle erläutert werden.

Der Shape Corrector

Die Aufgabe des *Shape Correctors* besteht darin, die äußeren Teile des Rollstuhls, die nicht vom inneren Radius abgedeckt sind, vor Kollisionen mit Objekten zu bewahren [17] (siehe Abbildung 3.30).

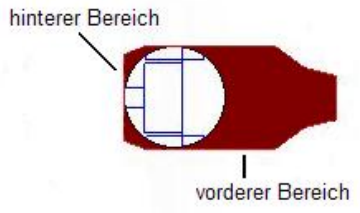

Abbildung 3.30: „Shape Corrector“

Dies wird dadurch erreicht, dass der in der Planungsphase berechnete Bewegungsbefehl überschrieben wird, sofern eine Kollision durch diese Bewegung stattfinden würde.
Es wird dabei unterschieden zwischen den 3 Situationen

- *imminent collision*
- *forward collision danger*
- *backward collision danger*

Der Rollstuhl wird zur Verwendung des *Shape Correctors* mit einer weiteren Sicherheitszone umgeben, deren Größe nur wenige Zentimeter beträgt. Sobald ein Hindernis innerhalb dieser Zone auftaucht, wird der Bewegungsbefehl der „*Nearness-Diagram*“-Navigation überschrieben, um eine etwaige Kollision zu vermeiden.

- *imminent collision*

  Diese Situation tritt immer dann auf, wenn im vorderen Bereich des Rollstuhls ein Hindernis so nah ist, dass jegliche Vorwärtsbewegung oder Drehung zu einer Kollision führen würde (siehe Abbildung 3.31 a). Dieser Bereich wird als *emergency stop area* bezeichnet. Der *Shape Corrector* sorgt in diesem Fall für eine direkte Rückwartsbewegung ($v_m < 0$, $w = 0$), so dass diese Zone von Hindernissen befreit wird.

- *forward collision danger*

  Das System befindet sich in dieser Situation, sobald sich Hindernisse im vorderen Bereich des Rollstuhls innerhalb der Sicherheitszone des *Shape Correctors* befinden. Je nach der Seite, auf welcher sich das Hindernis befindet (rechts oder links), wird ein direkter Drehbefehl vom Hindernis fort gegeben, ohne weitere Vorwärtsbewegung ($v_m = 0$, $w\ != 0$). In Abbildung 3.31 b) wird dies dargestellt. Sollten sich auf beiden Seiten der frontalen Sicherheitszone Hindernisse zu nah am Rollstuhl befinden, wird hier analog zur Situation „imminent collision“ eine direkte Rückwärtsbewegung eingeleitet.

- *backward collision danger*

  In diesem Falle befinden sich im hinteren Bereich des Rollstuhls Hindernisse innerhalb der *Shape-Corrector*-Sicherheitszone (Abbildung 3.31 c). Drehbewegungen könnten hierbei eine Kollision verursachen, weshalb der „*Nearness-Diagram*“-Befehl mit einer reinen Vorwärtsbewegung überschrieben wird.

Befindet sich das System zu irgendeinem Zeitpunkt sowohl in der Situation „forward collision danger“ als auch in der „backward collision danger“ wird sofort jegliche Bewegung

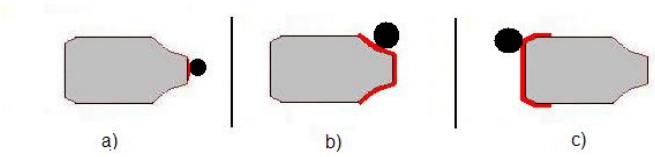

Abbildung 3.31: Situationen des „Shape Correctors“

gestoppt ($v_m = 0$, $w = 0$). In diesem Fall ist es nötig, den Rollstuhl auf andere Weise aus der Situation zu befreien (beispielsweise durch Manipulation der Umgebung). Eine Navigation mit Hilfe der „*Nearness-Diagram*“-Navigation ist nicht mehr möglich.

Rollstuhlsteuerung

Am Ende dieser Phase steht schließlich der endgültige Bewegungsbefehl in der Form (v_m, w) fest. Dieser wird anschließend an einen *Motion Controller* übergeben, der die eigentliche Bewegungssteuerung des Rollstuhls übernimmt. Mit Hilfe des errechneten Befehles ist sowohl gesichert, dass

- die Steuerung des Rollstuhls auf die Zielposition abgestimmt ist,
- ein Einfahren in enge Bereiche mit einer Ausschwenkbewegung initialisiert wird,
- Hindernisse innerhalb der Sicherheitszone um den Rollstuhl umfahren werden,
- die Berechung der Rotationsgeschwindigkeit auf weiche Bewegungen ausgelegt ist,
- die Berechnungen der Fahrtgeschwindigkeit sowohl auf der Entfernung der Hindernisse, als auch auf der Stärke der Drehbewegung basiert und letztendlich
- der *Shape Corrector* den Rollstuhl vor Kollisionen bewahrt und zur Not die Bewegungen korrigiert oder stoppt.

4 Testläufe und Testergebnisse

Dieses Kapitel dient dazu, das in den Kapiteln 2 und 3 beschriebene kollisionsvermeidende Navigationsverfahren genauestens zu evaluieren, um so Stärken und Schwächen der Methode aufzuzeigen. Gleichzeitig werden die Erweiterungen der originalen „*Nearness-Diagram*“-Navigation in der realen Umgebung auf ihre Tauglichkeit hin untersucht und verifiziert.

Dies geschieht anhand verschiedener Testläufe innerhalb vordefinierter Szenarien. Jedes Szenario testet dabei einen bestimmten Aspekt, welcher besondere Anforderungen an das Navigationsmodul stellt. Dabei wurden die Testsituationen so gewählt, dass sie natürlichen Büro- und Innenraumumgebungen entsprechen, um somit die eingehend definierte Zielumgebung abzudecken.
So wird im ersten Abschnitt die Fahrmechanik im Allgemeinen untersucht. Das Szenario 1 befasst sich mit dem Fahrverhalten in Korridoren, welche in ihrer Breite variieren. Das zweite Szenario betrachtet das Einfahren und Verlassen von Türen und Durchgängen, welche sich wiederum in ihrer Breite unterscheiden. Hierbei wird gleichfalls das Durchfahren von Türen innerhalb von Korridoren untersucht.
Szenario 3 befasst sich mit generellen und speziellen Drehbewegungen um Richtungswechsel während der Navigation zu untersuchen. Hierbei wird besonders auf die Drehfreiheit geachtet und das Verhalten in engen Situationen untersucht. Der letzte Testdurchlauf testet das Navigationssystem auf dynamische Änderungen der Umgebung, dies können Personen sein, die den Weg des Rollstuhls kreuzen oder die Durchfahrt verengen, als auch sich öffnende und schließende Durchgänge (beispielsweise Bürotüren).

Im ersten Abschnitt dieses Kapitels werden die Szenarien definiert und die verwendeten Parameter zur Variation der Umgebung eingeführt. Anschließend erfolgt die Betrachtung der Testdurchläufe. Hier wird die Testumgebung dargestellt und das aufgezeichnete Verhalten des Navigationsmoduls und des Rollstuhls dargelegt. Abschließend erfolgt eine Betrachtung der Testfahrten im Zusammenhang mit den Grundlagen des implementierten System. Es werden Situationen von besonderer Bedeutung oder Komplexität noch einmal betrachtet und analysiert.

4.1 Definition der Szenarien

Die Wahl der „*Nearness-Diagram*“-internen Parameter stützt sich zum einen auf die Erfahrung der Entwickler der Methode ([24]), zum anderen auf die eigenen Erfahrungen aufgrund vieler Testläufe.
So wurden für die Szenarien folgende Parameter des Navigationsmoduls gewählt:

- $v_{max} = 350mm/sek$ stellt die maximale Geschwindigkeit innerhalb von Büroumgebungen, in denen mit Personenverkehr gerechnet wird, dar.
- $w_{max} = 45°/sek$ als maximale Rotationsgeschwindigkeit verhindert zu starke Drehbewegungen.
- $d_s = 550mm$ definiert die Sicherheitszone zur Außenseite des Rollstuhls. Alle Hindernisse, welche eine geringere Entfernung haben, werden als sicherheitskritisch eingestuft.
- $d_{shapecorrector} = 50mm$ stellt die Sicherheitszone des *Shape Correctors* dar. Hindernisse, die diese Entfernung unterschreiten, lösen das Verhalten des Shape Correctors aus und unterbinden somit das Verhalten der „*Nearness-Diagram*“-Navigation.

Szenario 1: Korridorfahrten

Das erste Szenario dient der Untersuchung der Fahrmechanik innerhalb von Korridor- und Flurumgebungen verschiedener Breite. Als Parameter wird hierbei die **Breite des Korridors** variiert, um so eine Vielzahl an natürlichen Umgebungen darzustellen. Das Fahrverhalten soll für folgende Korridorbreiten untersucht werden:

- unbegrenzter Raum (freie Fläche)
- breiter Korridor (3m)
- schmaler Korridor (1,5m - 2m)
- sehr schmale Verengung (90cm)

Szenario 2: Türdurchfahrten

In diesem zweiten Experiment wird in mehreren Testdurchläufen das Verhalten beim Einfahren in Durchgänge und Türen betrachtet und evaluiert. Dies ist eine besondere Anforderung an das Navigationsmodul eines Rollstuhls, da durch die ausladende Länge eine spezielle Ausschwenkbewegung durchgeführt werden muss. Als Parameter werden hierbei sowohl die **Breite des Durchgangs**, als auch die **Breite des Korridors**, in dem sich der Durchgang befindet, variiert. So können besonders schwierige Situationen, wie enge Türen in engen Fluren, aber auch unkritische Umgebungen (weiter Durchgang in breitem Korridor) untersucht werden.
Breite des Durchgangs:

- sehr schmaler Durchgang (85cm)
- normale Tür (1m)
- Doppeltür (2m)
- unbegrenzter Durchgang (keine Begrenzung)

Breite des Korridors:

- schmaler Korridor (1,50m)

- breiter Korridor (2,50m)
- unbegrenzter Raum

Szenario 3: Drehen und Wenden

Szenario Nummer 3 dient der Erforschung des Drehverhaltens der implementierten Steuerung. Dies ist insbesondere sinnvoll, um auf weiche und natürliche Drehungen zu achten, und somit eine hohe Komfortabilität zu erreichen. Außerdem wird in diesem Durchgang untersucht, wie auf Hindernisse in Drehrichtung reagiert wird beziehungsweise wie die Drehbewegung in engen Situationen verläuft. Es wird in diesem Zusammenhang auch auf die implementierte Rückwärtsbewegung geachtet, welche in Situationen der niedrigen Sicherheit (LS1 und LS2) bei Drehungen auf ein Hindernis zu stattfindet. Als Parameter wird hierbei **die Freiheit des Rollstuhls** zu seinen Seiten verändert, deren Variation folgendermaßen festgehalten wird:
Freiheit des Rollstuhls:

- volle Freiheit (Drehung auf freier Fläche)
- zu einer Seite begrenzt (mit den Abständen 1,5m und 80cm zur Rollstuhlposition)
- zu beiden Seiten begrenzt (mit einer Breite des Korridors von 2,5m, 1,5m und 1m)

Szenario 4: Dynamische Änderungen der Umgebung

Im abschließenden Szenario werden dynamische Veränderungen der Umgebung simuliert und damit untersucht, wie das Verfahren auf solche Situationen reagiert. Diese Änderungen beinhalten zum einen das Kreuzen des Pfades von Personen und zum anderen die Variation der Umgebung durch Öffnen und Schließen neuer Durchfahrtmöglichkeiten.

- Personenkreuzung: frontal, hinten
- Durchfahrtsmöglichkeiten: Schließen einer Durchfahrtsmöglichkeit mit Alternative/ohne Alternative; Öffnen einer Durchfahrtsmöglichkeit mit Alternative/ohne Alternative

4.2 Testläufe

Die Szenariendurchführung wurde in realer Umgebung durchgeführt. Es wurden während der Fahrt signifikante Daten aufgezeichnet, welche eine anschließende Untersuchung des Verhaltens erlauben. Dies sind zum einen die vom „*Nearness-Diagram*“-Modul detektierten Situationen und daraus errechneten Geschwindigkeiten (v_m und w), um somit feststellen zu können, wie häufig es Situationstransitionen gibt, da dies auch einhergeht mit Änderungen in der Fahrmechanik. Die Darstellung der Situation beschränkt sich aus Gründen der Übersichtlichkeit auf die in der Original-Methode definierten fünf Situationen (HSGR, HSWR, HSNR, LS1 und LS2). Zum anderen wird die Rollstuhlposition aus den Odometriedaten aufgezeichnet, um im Nachhinein eine Abbildung des zurückgelegten Weges darstellen und untersuchen zu können. Schließlich erfolgt eine Berechnung der Rechenzeit des Moduls, um den zeitkritischen Aspekt der Echtzeitanwendung nachvollziehen zu können.
Zur Veranschaulichung des Verhaltens sind neben der Start- und Zielposition an signifikanten Stellen der Testläufe Anmerkungen eingefügt worden, auf die in der anschließenden Analyse besonderes Augenmerk gelegt wird.

4.2.1 Szenario 1: Korridorfahrten

Die Durchführung dieses Szenarios geschah in der Büroumgebung der 8. Ebene des Mehrzweckhochhauses der Universität Bremen. Dies war in sofern vorteilhaft, da die Flure in dieser Umgebung eine rollstuhlgerechte Breite aufwiesen und an einigen Stellen durch Schränke oder Stühle verengt waren und somit eine gute Testumgebung darstellten. Gleichzeitig wies die gewählte Strecke an verschiedenen Stellen die eingehend definierten Korridorbreiten auf. So gab es Bereiche mit schmaler (1,5m bis 2m) als auch mit weiter (3m und mehr) Korridorbreite. Der Test auf besonders schmale Verengungen konnte aufgrund einiger Türen auf dem Korridorweg erreicht werden.
Es folgt eine Karte mit eingezeichneter zurückgelegter Strecke (4.1 oben). Innerhalb dieser

Karte sind sowohl die Startposition, sowie die Streckenposten (goal1 bis goal5) eingezeichnet, wobei der letzte Streckenposten der Ausgangsposition (S) und gleichzeitig der Zielposition (Z) entspricht. Schließlich werden die Plots der signifikanten Daten aufgeführt und betrachtet (4.1 unten).

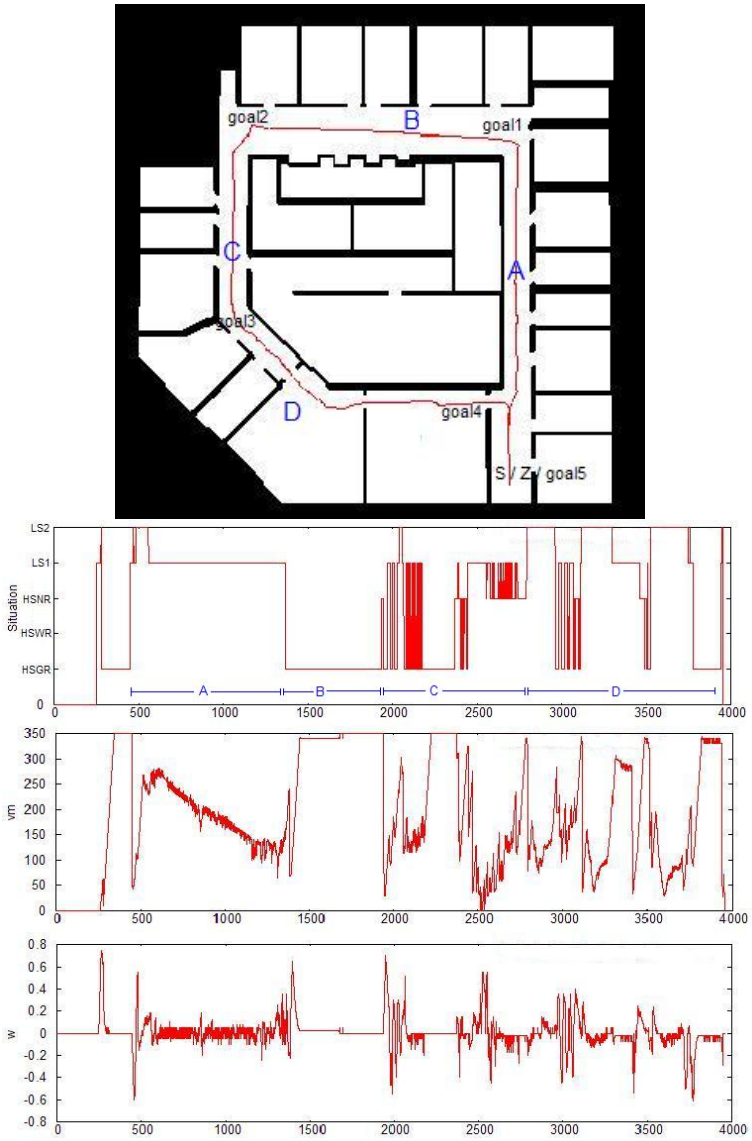

Abbildung 4.1: Szenario 1 - Karte und Logdaten

Die Betrachtung der Streckenplots und der dazugehörigen Karte lässt eine gute Betrachtung des Verhaltens in Korridorumgebungen zu. Streckenumgebung A wies eine Korridorbreite von unter 2 Metern auf. In den Aufzeichungen der Situation ist zu erkennen, dass hier immer in der LS1-Situation navigiert wurde, somit immer eine der beiden Korridorseiten innerhalb der Sicherheitszone lag. Aus diesem Grund wurde die Geschwindigkeit entsprechend der Entfernung zum nächsten Hindernis reduziert. Die Drehbewegungen in engen Korridoren waren nicht immer ruckelfrei. Dieses Verhalten ist darauf zurückzuführen, dass die „*Nearness-Diagram*"-Methode stets vom nächsten Hindernis (enthalten im Sektor s_{ml} oder s_{mr}) weg lenkt. Drehbewegungen von einer Wand in einem engen Korridor führen dazu, dass die Front des Rollstuhls sich zur anderen Wand bewegt. Die Unterschiede der Sektoren s_{ml} und s_{mr} können also in kurzer Zeit sehr groß sein, was ein ruckelfreies Fahren verhindert.

Beim Einbiegen in den breiteren Flur (Streckenabschnitt B) und Erreichen des ersten Streckenpostens ($goal1$) wird anschließend für einige Zeit im *High Safety*-Bereich navigiert, weswegen die Geschwindigkeit hier stets nah an v_{max} liegt.

Beim Einbiegen und Fahren im engeren Flur (C) schaltet die Situation oft von HSGR nach LS1. Dies liegt daran, dass der Flur geringfügig breiter ist, als der erste enge Flur. Somit ist nicht immer eine der beiden Seiten innerhalb der Sicherheitszone. Kurz darauf wird Streckenposten $goal3$ erreicht.

Im anschließenden Abschnitt D sind drei Verengungen in Form von Türen zu erkennen. In den Plots ist dieses durch das dreimalige Erreichen der Situation LS2 zu sehen. In diesem Bereich sind vor und während dem Einlenken in die Verengungen kleinere Ausschläge der Rotationsgeschwindigkeit zu erkennen. Dies liegt daran, dass das Verhalten versucht, die Türzargen auf gleicher Distanz zu den Seiten des Rollstuhls zu halten. Des Weiteren sind Schwankungen in der Rotationsgeschwindigkeit bei den Übergängen der Streckenabschnitte zu erkennen, da diese jeweils mit einer Kurve verbunden sind. Auch hier geht die Fahrgeschwindigkeit entsprechend herunter, um ein Umkippen oder schaukelndes Verhalten des Rollstuhls während der Drehung zu verhindern.

Die Zielposition (Z) ist ohne Kollision mit einem Hindernis erreicht worden.

Die Berechnungszeit des „*Nearness-Diagram*“-Moduls von der Akquisition der Sensordaten bis hin zum Senden des Bewegungsbefehls an den *Motion Controller* erreichte über den gesamten Testlauf einen Maximalwert von 70msek, lag ansonsten stets im Bereich zwischen 30 und 40msek. Dieser Wert schließt die Ausgabe von Debug-Informationen und visuellen Darstellungen ein und kann somit noch verringert werden.

4.2.2 Szenario 2: Türdurchfahrten

Das Ziel dieses Szenarios ist, das Verhalten beim Einfahren in Türen und Durchgänge verschiedener Breite in mehr oder weniger engen Korridorumgebungen zu untersuchen. Aus diesem Grund wurde eine variable Umgebung mit Hilfe von Umzugskartons aufgebaut und es konnte durch einfaches Verschieben der Kartons jede der eingehend definierten Durchgangsbreiten simuliert werden.
Die folgende Abbildung zeigt den Versuchsaufbau mit den entsprechenden Parametern.

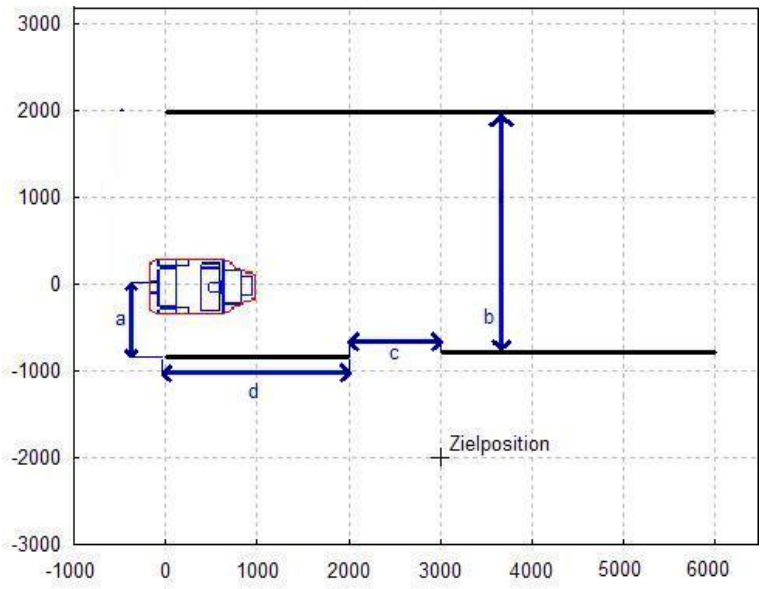

Abbildung 4.2: Versuchsaufbau Szenario 2

Bei diesem Aufbau variiert der Parameter a (Abstand der Wand zum Rollstuhlzentrum) nicht, er bleibt konstant bei 80cm. Auch der Abstand zum Durchgang (d) von 2m wird nicht verändert. Lediglich die Parameter b und c ändern sich entsprechend dem Versuchsdurchlauf in folgenden Abständen:

- Breite des Korridors b - Durchlauf 1: unbegrenzt; Durchlauf 2: 2,50m; Durchlauf 3: 1,50m
- Breite des Durchgangs c - Änderung in jedem Durchlauf: unbegrenzt, 2m, 1m, 85cm

In den unterschiedlichen Zusammenstellungen der Parameter sind in diesem Szenario 12 Durchläufe ausgeführt worden, die folgende Tabelle fast diese zusammen.

Versuchslauf	Parameter a	Parameter b	Parameter c	Parameter d
1	80cm	∞	∞	2m
2	80cm	∞	2m	2m
3	80cm	∞	1m	2m
4	80cm	∞	85cm	2m
5	80cm	2,50m	∞	2m
6	80cm	2,50m	2m	2m
7	80cm	2,50m	1m	2m
8	80cm	2,50m	85cm	2m
9	80cm	1,50m	∞	2m
10	80cm	1,50m	2m	2m
11	80cm	1,50m	1m	2m
12	80cm	1,50m	85cm	2m

Tabelle 4.1: Szenario 2 - Versuchsgruppen

Anhand dieser Tabelle wurden die 12 Durchläufe mit jeweils drei Testfahrten in der Testumgebung durchgeführt und die jeweiligen Odometriedaten aufgezeichnet. In den folgenden Abbildungen werden die Ergebnisse deutlich, sie sind nach der entsprechenden

Versuchsgruppe zusammengefasst. Im Anschluss an die Abbildung erfolgt eine Betrachtung der Ergebnisse.

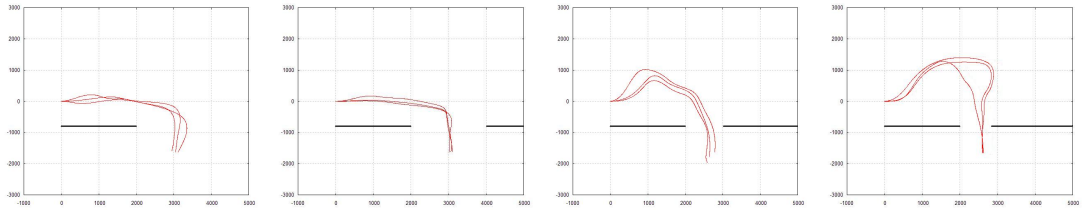

Abbildung 4.3: Szenario 2 - Versuchsgruppe 1

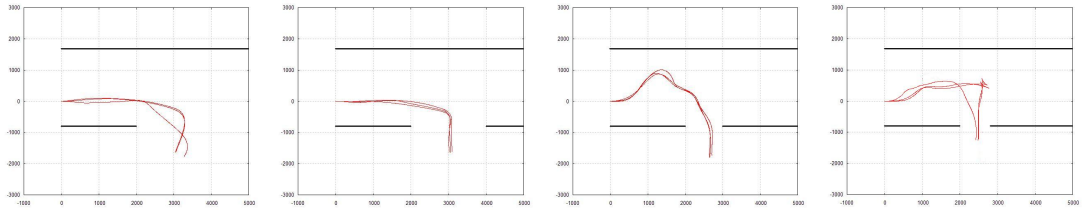

Abbildung 4.4: Szenario 2 - Versuchsgruppe 2

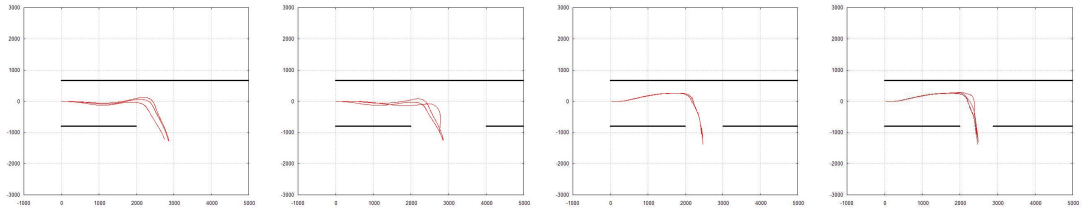

Abbildung 4.5: Szenario 2 - Versuchsgruppe 3

Bei den Versuchen 1 bis 4 weist der Korridor keine Begrenzung auf. In Abbildung 4.3 ist zu erkennen, dass eine Ausschwenkbewegungen bei Durchgängen ab einer Breite von 1,5m und darunter ausgeführt wird. Nur auf diese Weise ist es dem Rollstuhl möglich, in die entsprechende Tür einzufahren. Hierbei kann die Ausschwenkbewegung sehr ausgeweitet sein, da keine Begrenzung zur linken Seite gegeben ist. Der besonders schmale Durchgang (85cm) erfordert eine sehr lange Bewegung auf einer Kreisbahn um die Tür herum, es wird erst spät eingelenkt um direkt mit der Breite des Rollstuhls in den Durchgang einzufahren. Die Durchgänge mit eine Breite von 2m und mehr können ohne besondere Bewegungen

passiert werden.
In der nächsten Versuchsgruppe (5-8) ist der Raum zum Ausschwenken durch eine linke Wand beschränkt (siehe Abbildung 4.4), der Korridor weist eine Breite von 2,50m auf. Die Bewegung wird dennoch ausgeführt, der Rollstuhl bewegt sich - absichtlich - auf das Hindernis zu, um die Ausschwenkbewegung komplett durchführen zu können. Breite Durchgänge zeigen ein analoges Verhalten zum ersten Versuchslauf, ein Ausscheren ist nicht nötig. Bei dem sehr engen Durchgang ist es vorgekommen, dass vor der Drehung in die Tür hinein eine Rückwärtsbewegung ausgeführt werden musste, da sich die linke Seite der Tür zu nah an der Front des Rollstuhls befand. Dies entspricht einem Fahrverhalten, welches auch menschliche Rollstuhlfahrer oft in solchen Situationen anwenden.
Bei den letzten Versuchen 9 bis 12 ist der Korridor noch enger, der Abstand der Wände beträgt lediglich 1,50m. Hier ist es durch die Geometrie der Ausscherbewegung nötig, sehr nah an die gegenüberliegende Wand heranzufahren, damit in die Tür eingelenkt werden kann. In Abbildung 4.5 ist die Bewegung im engen Korridor zu erkennen.

4.2.3 Szenario 3: Drehen und Wenden

Die Testreihe zur Untersuchung der Drehbewegungen gestaltete sich auf ähnliche Weise, wie der Aufbau des zweiten Szenarios. In der Testumgebung wurden Umzugskartons verwendet, um die Hindernisse darzustellen und die einzelnen Versuche auf einfache Weise umzugestalten. Der Aufbau gestaltet sich entsprechend Abbildung 4.6.

Auch dieses Szenario gliedert sich in drei Testreihen. In der ersten Phase gibt es keine Einschränkungen hinsichtlich der Umgebung (Parameter a und b unbegrenzt). Versuchsgruppe 2 testet das Verhalten auf Drehbewegungen gegen eine Wand. Hierbei variiert der Parameter a zwischen einer Drehung, die ohne Rückwärtsbewegung möglich ist, und einer Drehung, die genau dieses verlangt. In den abschließenden Versuchen wird das Wendeverhalten in Korridoren untersucht. Die Zielposition ist so gewählt, dass eine Drehbewegung

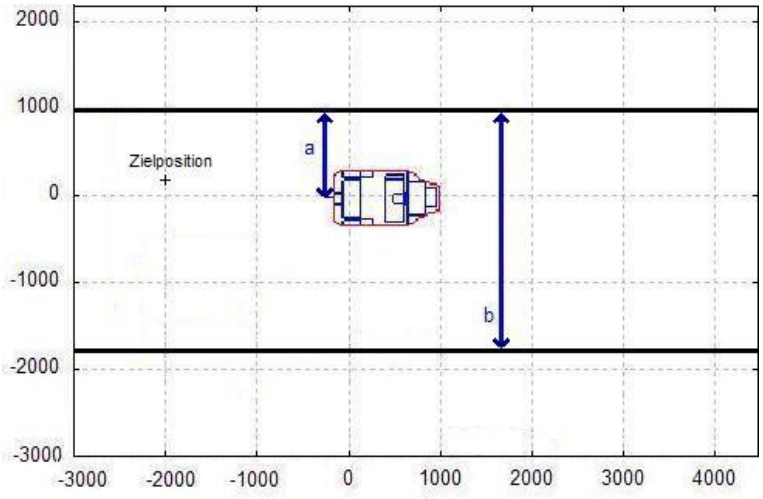

Abbildung 4.6: Versuchsaufbau Szenario 3

in Richtung der Wand auf jeden Fall durchgeführt wird. Dies wird dadurch ausgelöst, dass sich das System in einer *goal in region*-Situation befindet, der ausschlaggebende Richtungssektor der Bewegung also dem Zielsektor entspricht (siehe Abschnitt 3.2.3).
Die folgende Tabelle fasst den Versuchsablauf zusammen.

Versuchslauf	Parameter a	Parameter b
1	∞	∞
2	1,5m	∞
3	80cm	∞
4	80cm	2,50m
5	80cm	1,50m
6	80cm	1m

Tabelle 4.2: Szenario 3 - Versuchsgruppen

Eine Erläuterung der aufgezeichneten Daten schließt sich den folgenden Abbildungen der Versuchsgruppen an.

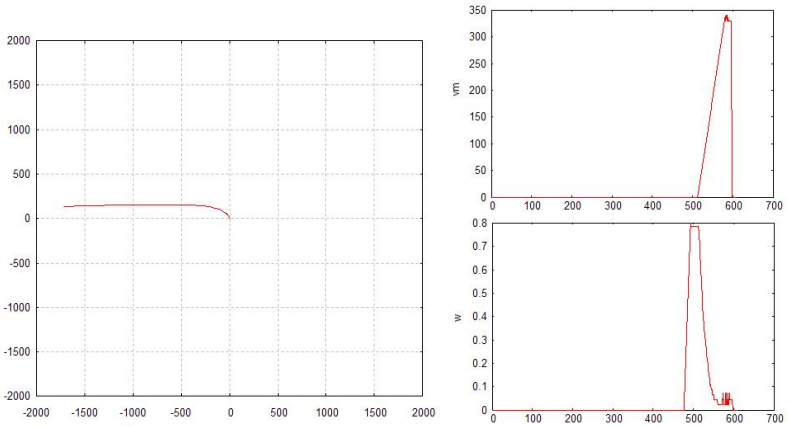

Abbildung 4.7: Szenario 3 - Versuch 1

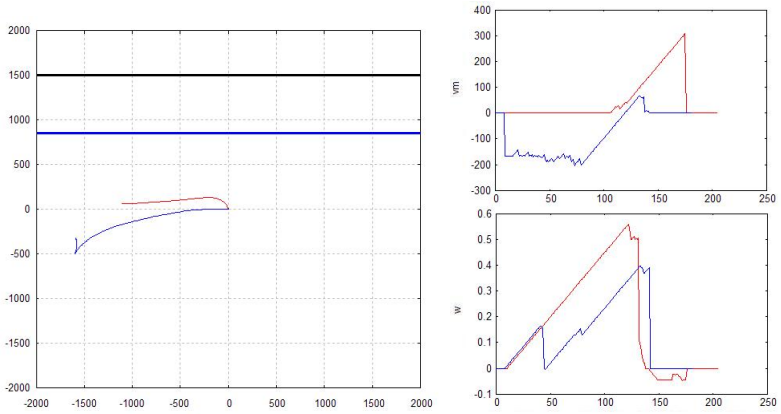

Abbildung 4.8: Szenario 3 - Versuche 2-3

Die Ergebnisse der ersten Versuchsreihe (siehe Abbildung 4.7) zeigen, dass Wendeverhalten auf weiter Fläche einer intuitiven Lenkung entsprechen. So wird zuerst mit maximaler Rotationsgeschwindigkeit gelenkt, die dann reduziert wird, je näher Zielsektor und Blickrichtung einander sind.

Versuchsgruppe 2 zeigt ein analoges Verhalten (siehe Abbildung 4.8). Allerdings wird hierbei im zweiten Versuch - blau gekennzeichnet - deutlich, dass eine Rückwärtsbewegung eingeleitet wird, da der Freiraum nicht für ein vollständiges Wendemanöver ausreichend ist.

Schließlich zeigen die Versuche 4 und 5 der nächsten Testreihe (Abbildung 4.9) ein ähnli-

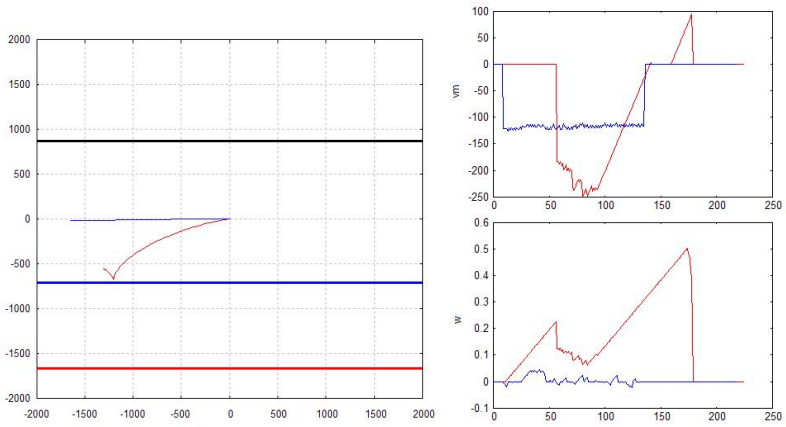

Abbildung 4.9: Szenario 3 - Versuche 4-5

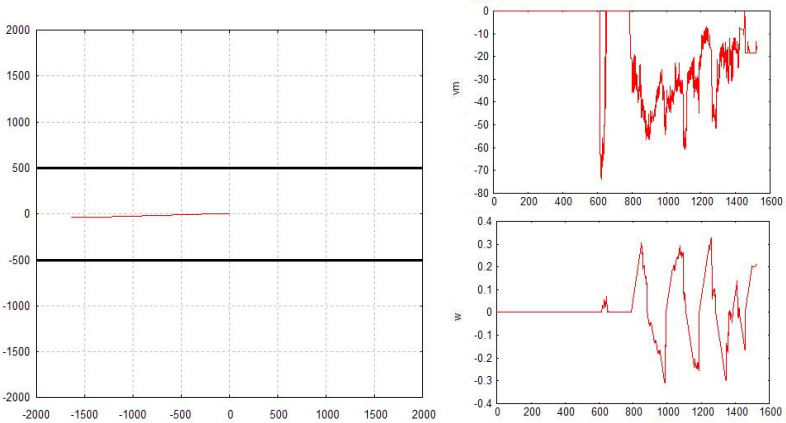

Abbildung 4.10: Szenario 3 - Versuch 6

ches Verhalten. Hier wird aufgrund mangelnden Raumes in Drehrichtung auf die Rückwärtsbewegung ausgewichen, um so das Wenden zu ermöglichen. Im zweiten Versuch (blau) wird deutlich, dass hier keine Drehbewegung eingeleitet wird. Der Korridor ist zu schmal, weswegen komplett rückwärts navigiert wird. Dies zeigt sich auch in der Aufzeichnung der Rotationsgeschwindigkeit, in der keine größeren Ausschläge erkennbar sind.
Hier ergibt sich ein Vorteil der „*Nearness-Diagram*“-Navigation durch die Fallunterscheidungen. Während beispielsweise andere Verfahren versuchen würden, die Drehbewegung auch in zu engen Korridoren durchzuführen, bis schließlich festegestellt wird, dass eine weitere Bewegung in diese Richtung nicht möglich ist, so wird die geänderte Variante der

„*Nearness-Diagram*“-Methode dies bereits vor dem Beginn der Drehung erkennen, diese aussetzen und stattdessen auf eine Rückwärtsbewegung ausweichen.
Im letzten Versuch 6 (Abbildung 4.10) ist ebenfalls nicht genug Freiraum für ein Wendemanöver vorhanden, die beiden nächsten Hindernisse innerhalb der Sicherheitszone liegen zu nah beieinander. Aus diesem Grund wird rückwärts gefahren, um aus dieser Situation zu entkommen. Die Fahrtgeschwindigkeit ist niedriger als im vorhergehenden Versuch, da sich die Hindernisse sehr viel näher an der Rollstuhlaussenseite befinden und vorsichtiger navigiert werden muss. Auch gibt es stärke Unterschiede in der Rotationsgeschwindigkeit, weil bereits kleine Drehungen zur jeweils anderen Richtung sehr nah an die Wänder heran führen, weshalb öfters die Richtung korrigiert werden muss.

4.2.4 Szenario 4: Dynamische Änderungen der Umgebung

In der abschließenden Testphase wird das Verhalten auf Veränderungen der Umgebung evaluiert. Auch dieses Szenario gliedert sich in zwei Versuchsreihen. Die ersten Untersuchungen betrachten das Verhalten der „*Nearness-Diagram*“-Navigation bei Personenverkehr im Sichtbereich. So wird eine Person den Weg zur Zielposition verdecken, indem sie von links in den Sicherheitsbereich des Rollstuhls eintritt. In einem zweiten Versuch kreuzt die Person den hinteren Bereich der Sicherheitszone.
Abbildung 4.11 zeigt das Erscheinen der Person frontal und die daraufhin eingeleitete Ausweichbewegung, sobald diese die Sicherheitszone betritt. Das Passieren einer Person der hinteren Sicherheitszone verändert die Fahrtrichtung des Rollstuhls nicht. Lediglich die Geschwindigkeit wird reduziert (ohne Abbildung).

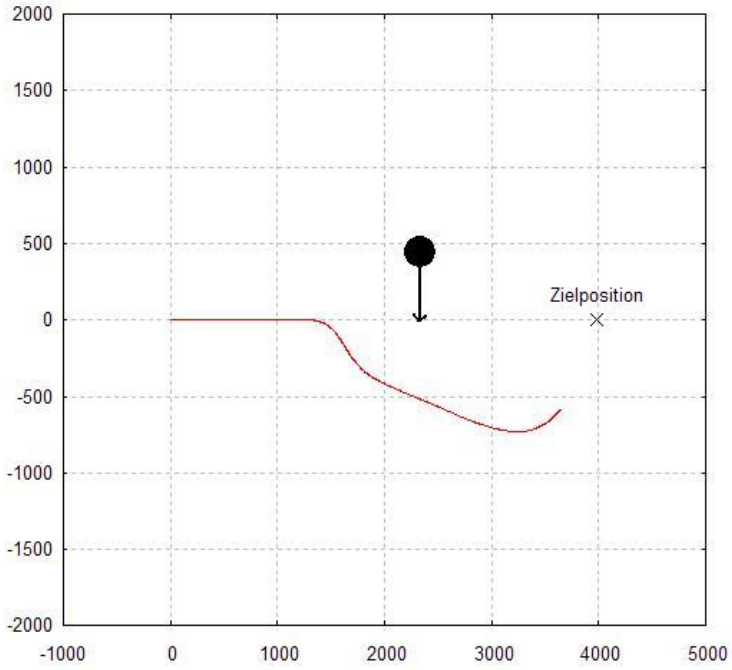

Abbildung 4.11: Szenario 4 - Versuch 1

In einer zweiten Testreihe werden mit Hilfe von Umzugskartons Türen simuliert, die sich öffnen und schließen und somit neue Wege frei geben oder aber geplante Routen verhindern.

In Abbildung 4.12 ist erkennen, dass bei Fahrtpunkt A der Durchgang zum Ziel geschlossen wurde. Daraufhin wurde auf die Ausweichroute rechts umgelenkt. In einem weiteren Versuch wurde die einzige Tür zum Raum, in dem sich der Rollstuhl befandt, geschlossen. In folgender Abbildung 4.13 ist der schematische Aufbau dieses abschließenden Versuchs mit den dazugehörigen Logdaten abgebildet.

Es wird deutlich, dass nach dem Schließen des Durchgangs (zum Zeitpunkt B) keine weitere Möglichkeit existiert, das Ziel anzusteuern. Es wird im Zuge der „*Nearness-Diagram*“-Berechnungen keine *free walking area* detektiert, da keine Regionen existieren. Eine Navigation ohne *free walking area* ist nicht möglich, weshalb in diesem Falle keine Situation

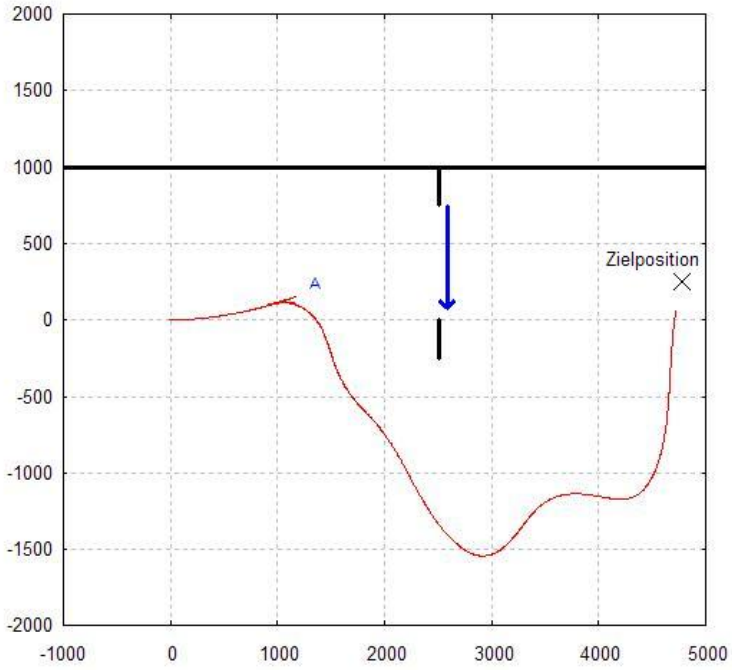

Abbildung 4.12: Szenario 4 - Versuch 2

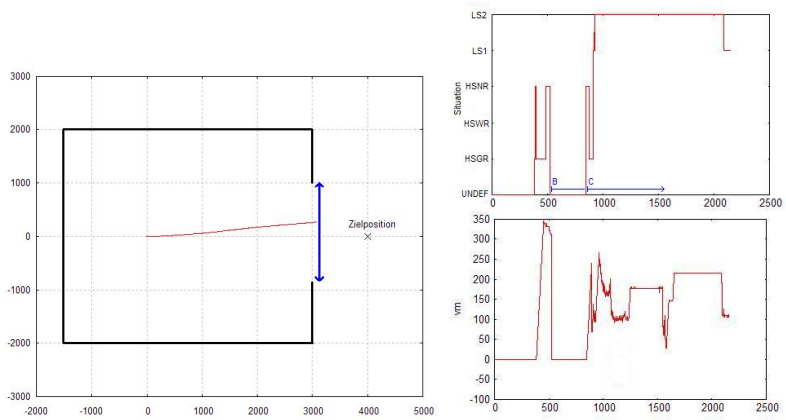

Abbildung 4.13: Szenario 4 - Versuch 3

erkannt wird (siehe Abschnitt B der Situationsaufzeichnung). Sobald der Durchgang wieder geöffnet wird (Zeitpunkt C), entsteht eine *free walking area* und die Fahrt wird erneut aufgenommen.

Dies ist in sofern ein positiver Effekt der „*Nearness-Diagram*“-Navigation, da solche Fälle

nicht zu einem Versagen des Systems führen, sondern als eigener Fall behandelt werden. Die „*Elastic-Band*“-Steuerung beispielsweise kommt mit solchen starken Änderungen der Umgebung nur schwer zurecht (siehe auch Abschnitt 2.2.3), während das Stoppen und die Wiederaufnahme der Fahrt keine Probleme für die Navigation mit Hilfe der „*Nearness-Diagram*“-Methode darstellt.

4.3 Testergebnisse

Abschließend sollen in diesem Abschnitt die Testergebnisse zusammenfassend betrachtet werden.
Die im ersten Szenario dargestellten Situationen wurden in natürlicher Umgebung evaluiert und haben gute Ergebnisse erzielt. So war es möglich mit Hilfe der „*Nearness-Diagram*“-Methode auch durch enge Bereiche zu navigieren, wenn dies verlangt wurde. In schmalen Korridorumgebungen ist aufgrund der Definition der „Low Safety 2“-Situationen immer dafür gesorgt worden, dass ausreichend Raum zu den Seiten des Rollstuhls gehalten wurde. Gleichzeitig ist die Geschwindigkeit in solchen kritischen Bereichen auf ein Minimum reduziert worden (teilweise unter 5cm pro Sekunde), um diese Situationen mit einem hohen Grad an Sicherheit zu meistern.
An einigen Stellen konnten häufige Zustandsübergänge aufgezeichnet werden, wodurch in diesen Situationen starke Änderungen der Bewegungs- und Rotationsgeschwindigkeiten die Folge waren. Besonders beim Umfahren einer Kurve und gleichzeitigem Einfahren in einen schmalen Korridor war dies der Fall. In Situationen hoher Sicherheit konnte aber zumeist ein stabiles Fahrverhalten erreicht werden. In Korridorumgebungen waren die Lenkbewegungen nicht immer ruckelfrei. Dies ist auf die schnellen Wechsel der Sektoren mit den nächsten Hindernissen zurückzuführen, da sich die Front des Rollstuhls auch bei einer leichten Drehung zügig der Wand näherte.

Die zweite Testreihe - das Einfahren in Durchgänge verschiedener Breite - stellte eine be-

sondere Anforderung an das Navigationsmodul. Das Passieren einer engen Tür ist kein trivialer Bewegungsablauf, sondern verlangt sehr viel Vorsicht und vorausschauendes Fahren. Innerhalb der Versuchsabläufe konnten schmale Türen mit einer Breite von unter 90cm sicher durchfahren werden. Dies entsprach einem Freiraum zu den Seiten des Rollstuhls von jeweils nur etwa 8 bis 10 Zentimetern. Die Einfahrt in äußerst enge Durchgänge (unter 80cm, 5-6 Zentimeter Freiraum zu den Seiten des Rollstuhls) konnte nur in wenigen Fällen erfolgreich durchgeführt werden. Hier kamen mehrere negative Einflüsse zusammen. Zum einen war die Ausschwenkbewegung nicht exakt genug, um schließlich mittig in die Tür zu lenken. Zum anderen standen die Gleiträder nach dem Ausscheren schräg zur Türeinfahrt, während die Bewegung den Rollstuhl nah an der Türzarge stoppte. In diesen Situationen greift die Methode des *Shape Corrector* sehr häufig, da zu der Außenseite des Rollstuhls nur wenige Zentimeter Raum zum nächsten Hindernis ist. Durch die Schrägstellung der Vorderräder musste eine Art „Ruckbewegung“ durchgeführt werden, um sie wieder in eine gerade Position zu bringen. Der *Shape Corrector* weist allerdings nur minimale Bewegungsbefehle an, gerade weil sich ein Hindernis sehr nah am Rollstuhl befindet. Aus diesem Grund konnte die Einfahrt nicht komplettiert werden.
Alle anderen Durchgänge konnten zufriedenstellend gemeistert werden. Sogar in schmalen Korridorumgebungen, in denen die Auschwenkbewegung teilweise sehr nah an die gegenüberliegende Wand führte, wurden durch die Berechnung des *safety drive by*-Sektors Kollisionen vermieden.

Durch die geänderte Geometrie bei Wendemanövern, welches in der dritten Szenariodefinition evaluiert wurde, war es möglich, selbst in engen Bereichen durch rückwärtige Steuerung eine Drehung auszuführen. Diese Methode ist bei der Originalentwicklung der „*Nearness-Diagram*“-Navigation nicht vorgesehen, stellt sich aber als unabdingbar heraus. Drehbewegungen innerhalb von Situationen, in denen zu den Seiten des Rollstuhls nicht genug Platz vorhanden ist, wurden korrekt ausgesetzt und stattdessen eine komplette Rückwärtsnavigation eingeleitet, um aus solchen Umgebungen zu entkommen.

Die Reaktionen der „*Nearness-Diagram*“-Navigation auf dynamische Veränderungen wurde im letzten Szenario untersucht. Sobald Personen den Sichtbereich des Rollstuhls betreten, werden sie als Hindernis erkannt, da sie - wie alle anderen wahrgenommenen Objekte - Auschläge in den Diagrammen verursachen. Somit wird die *free walking area* entsprechend um sie herum aufgebaut und umgelenkt, was einem Ausweichverhalten ähnlich ist. Beim Betreten der Sicherheitszone lösen Personen stets die „Low Safety“-Situationen aus und gehen somit in die Berechnung des Bewegungsbefehles ein. Es wird stets versucht, das Hindernis zu umfahren und die Geschwindigkeit entsprechend zu reduzieren. „*Dead lock*“-Zustände treten erst dann auf, wenn durch Manipulation der Umgebung (das Schließen einer Tür) keine *free walking area* detektiert werden kann. In diesem Fall wird die Bewegung des Rollstuhls gestoppt. Sobald nun wieder eine neue Durchfahrtsmöglichkeit entsteht, wird die Bewegung wieder aufgenommen, was einem natürlichen und intuitiven Navigationsverhalten entspricht.

5 Zusammenfassung und Ausblick

Im Zuge dieser Diplomarbeit wurde für den Bremer Autonomen Rollstuhl *Rolland* ein Steuerungssmodul zur Navigation in Innenräumen entwickelt und implementiert.
Die grundlegende Basis hierfür stellte die an der Universität Zaragoza in Spanien entwickelte Methode der „*Nearness-Diagram*“-Navigation dar. Da dieses Verfahren in seiner Grundfassung lediglich für holonome, kreisrunde Roboter ohne Bewegungseinschränkungen entwickelt wurde, waren mehrere Anpassungen nötig.
So wurde im ersten Schritt die originale Version der „*Nearness-Diagram*“-Methode in das laufende System des *Rolland 3* der Universität Bremen eingepflegt und im Anschluss daran auf die spezielle Form und Kinematik des Rollstuhls angepasst. Diese Anpassungsphase verlangte die Betrachtung und Analyse signifikanter Situationen, die besondere Anforderungen an das Navigationsmodul stellten, wie das Passieren von Bereichen, in denen sich die Lenkmethodik von Rollstühlen zu runden Robotern stark unterscheidet. Die Entwicklung einer Lösungsmechanik und die implementative Umsetzung waren das Ziel des nächsten Schrittes.
In der abschließenden Phase wurde das entwickelte Steuerungsmodul in mehreren Testszenarien auf seine korrekte Funktionsweise untersucht und genauestens getestet.

Die eingehend definierten Ziele lagen im Bereich der Kollisionsvermeidung, ganz gleich ob Objekt oder Personen. In den experimentellen Untersuchungen ist gezeigt worden, dass es mit Hilfe der erweiterten „*Nearness-Diagram*“-Navigation möglich ist, selbst komplexe Situationen zufriedenstellend zu absolvieren. Auch auf dynamische Veränderungen der Umgebung konnte gut reagiert werden.

5.1 Grenzen der „Nearness-Diagram“-Steuerung

Wie jedes lokale Verfahren hat auch auch die „*Nearness-Diagram*“-Navigation mit den Nachteilen der reaktiven Steuerung zu kämpfen. Es wird lediglich ein Ausschnitt der Umwelt betrachtet, der der aktuellen Wahrnehmung der Sensoren entspricht. Vorausschauendes Fahren ist also nur bedingt möglich, beispielsweise lässt sich nicht erkennen, ob in eine Sackgasse navigiert wird, sofern das Ende des Korridors noch nicht sichtbar ist.
Eine weitere Schwachstelle der Navigation mit Hilfe der „*Nearness-Diagrams*“ ist die starke Abhängigkeit von der Verteilung der Sektorwerte. Da die Berechnung der Bewegungsbefehle direkt von der Hindernisverteilung innerhalb der Sektoren abhängig ist, so ändert sich der Bewegungsbefehl bei stark dynamischer Umgebung dementsprechend häufig, da sich in solchen Situationen die *free walking area* oft in Lage und Größe verändert. In Bereichen mit hochfrequentiertem Personenverkehr, wie Bahnhofsplätzen oder Messehallen, könnte dieser Aspekt zu häufigen Transitionen der Zustände führen, womit ständige Wechsel der Bewegungsrichtung einhergehen.
Ein während der Testphase deutlich gewordender Nachteil bezieht sich auf die Architektur des Rollstuhls. Durch die Lage des Differentialantriebs auf der Hinterachse gibt es keine Möglichkeit, die vorderen gleitenden Räder explizit zu lenken. Nach einer Drehung auf der Stelle stehen diese Räder zumeist quer zur Rollstuhlachse. Eine Drehung in die entgegengesetzte Richtung ist anschließend nur mit einer starken Lenkbewegung oder einer kurzen kreisenden Bewegung nach vorne oder nach hinten zu erreichen, da die Gleiträder die Drehung ansonsten blockieren würden. Dieser Aspekt verhinderte in mehreren Situationen das vorsichtige Einlenken in einen engen Bereich, da in Situationen niedriger Sicherheit nur minimale Lenkbewegungen durch die „*Nearness-Diagram*“-Steuerung angewiesen werden.

5.2 Ausblick

Als mögliche Weiterentwicklung lässt sich die lokale „*Nearness-Diagram*“-Navigation gut mit einem globalen Bahnplaner verknüpfen. So wäre es denkbar, einen Pfad zu einem weit entfernten Ziel mit Hilfe von globalen Algorithmen planen zu lassen und lediglich die lokale Hindernisvermeidung an das „*Nearness-Diagram*“-Modul abzugeben. Ein ähnliches Prinzip wird für die Roboter der Universität Zaragoza betrieben. Hier arbeitet die „*Nearness-Diagram*“-Steuerung in einem hybriden System mit einem globalen Wegplaner zusammen.

Eine weitere denkbare Entwicklung der Steuerung mit Hilfe der „*Nearness-Diagram*“-Methode wäre die Verbindung mit weiteren Mitteln zur Zieleingabe. Die forcierte Auswahl einer bestimmten *free walking area* aus der Liste der erkannten Regionen, durch Sprachsteuerung oder ähnlichem, würde dem Benutzer die Möglichkeit geben, den Pfad selbstständig zu wählen. Die Navigation durch einen Korridor entspricht beispielsweise der Navigation einer fortlaufenden Region. Ein sprachlicher Befehl der Form „Folge diesem Flur bis zur nächsten Kreuzung“ wäre also denkbar, da die Steuerung in diesem Falle der Region folgen müsste, bis weitere Regionen in den „*Nearness-Diagrams*“ sichtbar würden.

Literaturverzeichnis

[1] Arkin, R.: *Behavior-Based Robotics.* Cambridge, Massachusetts : MIT Press, 1998

[2] Asensio, J.R. ; Montano, L.: A Kinematik and Dynamic Model-based Motion Controller for Mobile Robots. In: *15th IFAC World Congress* (2002)

[3] Borenstein, J. ; Koren, Y.: The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. In: *IEEE Journal of Robotics and Automation Vol 7, No 3* (1991)

[4] Brock, O. ; Khatib, O.: *High-Speed Navigation Using the Global Dynamic Window Approach.* 1999

[5] Fox, D. ; Burgard, W. ; Thrun, S.: The Dynamic Window Approach to Collision Avoidance. In: *IEEE Robotics and Automation Magazine* 4 (1997), Nr. 1, S. 23 – 33

[6] Gunnarsson, B.: *Laser Guided Vehicle - Java and MATLAB for Control.* 2002

[7] Hwang, J.-H. ; Arkin, R.C. ; Kwon, D.-S.: Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control. In: *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2003

[8] Jones, J.L. ; Flynn, A.M.: *Mobile Roboter: Von der Idee zur Implementation.* Addison-Wesley, 1996

[9] Khatib, M.: *Sensor-based motion control for mobile robots.* 1999

[10] Koren, Y. ; Borenstein, J.: Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991

[11] Lamiraux, F. ; Lammond, J.P.: Smooth motion planning for car-like vehicles. In:

IEEE Transactions on Robotics and Automation 17 (2001), Nr. 4, S. 498 – 501

[12] Mandel, C.: *Rolland III und SimRobot - ein Überblick*. 2005

[13] Minguez, J.: *Robot Shape, Kinematics, and Dynamics in Sensor-Based Motion Planning*, Universität Zaragoza, Spanien, Diss., 2002

[14] Minguez, J.: *Integration of Planning and Reactive Obstacle Avoidance in Autonomous Sensor-Based Navigation*. 2005

[15] Minguez, J.: *Research on Mobile Robotics and Rehabilitation Engineering*. 2005

[16] Minguez, J. ; Montano, L.: Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach. In: *IEEE Int. Conf. on Intelligent Robots and Systems* (2000)

[17] Minguez, J. ; Montano, L.: Robot Navigation in Very Complex, Dense, and Cluttered Indoor/Outdoor Environments. In: *15th IFAC World Congress* (2002)

[18] Minguez, J. ; Montano, L.: Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios. In: *IEEE Transactions on Robotics and Automation (February)* 20(1) (2004), S. 45–59

[19] Minguez, J. ; Montano, L. ; Khatib, O.: Reactive Colliscion Avoidance for Navigation with Dynamic Constraints. In: *IEEE Int. Conf. on Intelligent Robotics and Systems*, 2002

[20] Minguez, J. ; Montano, L. ; Santos-Victor, J.: Reactive Navigation for Nonholonomic Robots using the Ego-Kinematic Space. In: *IEEE Int. Conf. on Robotics and Automation*, 2002

[21] Minguez, J. ; Montano, L. ; Simeon, T. ; Alamni, R.: Global Nearness Diagram Navigation (GND). In: *IEEE Int. Conf. on Robotics and Automation*, 2001

[22] Minguez, J. ; Montesano, L. ; Montano, L.: *Autonomous Motion Generation for a Robotic Wheelchair*. 2006

[23] Minguez, J. ; Montesano, L. ; Montano, Luis: An Architecture for Sensor-Based

Navigation in Realistic Dynamic and Troublesome Scenarios. In: *IEEE Int. Conf. on Intelligenz Robot and Systems* (2004)

[24] Minguez, J. ; Osuna, J. ; Montano, L.: A „Divide and „Conquer“ Strategy based on Situations to achieve Reactive Collision Acoidance in Troublesome Scenarios. In: *15th IFAC World Congress* (2002)

[25] Montesano, L. ; Minguez, J. ; Montano, Luis: *Modeling the Static and the Dynamic Parts of the Environment to Improve Sensor-based Navigation.* 2005

[26] Nehmzow, U.: *Mobile Robotics: A Practical Introduction.* London : Springer Verlag, 2000

[27] Pfeiffer, A.: *Bahnplanung für mobile Roboter mit Hilfe der Potentialfeldmethode in dynamischer Umwelt.* 2005

[28] Prassler, E.: *MAid: Eine intelligente Mobilitätshilfe.* 2002

[29] Quinlan, S. ; Khatib, O.: Elastic Bands: Connecting Path Planning and Control. In: *IEEE Int. Conf. on Robotics and Automation* (1993)

[30] Schlegel, C.: Fast Local Obstacle Avoidance under Kinematic and Dynamic Constraints for a Mobile Robot. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998

[31] Siegwart, R.: Robox at Expo.02: A Large Scale Installation of Personal Robots. In: *Robotics and Autonomous Systems (Special issue on Socially Interactive Robots)* 42 (2003), S. 203–222

[32] Stachniss, C. ; Burgard, W.: An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments. In: *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002