TU

Technische Universität Wien

Diplomarbeit zur Erlangung des akademischen Grades
Diplomingenieur

In der Studienrichtung
Informatik

Konzeption und Simulation eines semiautonomen Rollstuhls

eingereicht am

Institut für Computersprachen
der Technischen Universität Wien

betreut durch das

Institut für Industrielle Elektronik und Materialwissenschaften
Forschungsgruppe FORTEC – Rehabilitationstechnik

eingereicht bei Ao.Univ.Prof. Mag.rer.nat. Dipl.-Ing. Dr.techn. Rudolf Freund
betreuender Assistent: Ass.Prof. Dipl.-Ing. Dr.techn. Wolfgang Zagler

erstellt von

Michael Bucek

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Wien, 09.10.2003

Danksagung

An dieser Stelle möchte ich all jenen danken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Diplomarbeit beigetragen haben.

Besonderen Dank gilt Herrn Ass.Prof. Dipl.-Ing. Dr.techn. Wolfgang Zagler, Leiter der Forschungsgruppe für Rehabilitationstechnik am Institut für Industrielle Elektronik und Materialwissenschaften, der diese Arbeit ermöglicht hat und mich bei der Durchführung betreut und stets durch seine wertvollen Anregungen unterstützt hat.

Des Weiteren möchte ich mich bei meinen Eltern Erika und Lothar Bucek bedanken, die mir mein Studium ermöglicht haben sowie bei Sylvia Stummer, die mir durch ihre Geduld und persönliche Unterstützung stets zur Seite steht.

Inhaltsverzeichnis

1. Einleitung 1
   1.1. Ziel und Umfang dieser Arbeit 1
   1.2. Definition: Semiautonom versus autonom 1
   1.3. Zweck und Nutzen eines semiautonomen Rollstuhls 2
2. Systembeschreibung 6
   2.1. Einführung in die Simulation 6
      2.1.1. Vorteile einer Simulation 7
      2.1.2. Nachteile einer Simulation 9
   2.2. Softwaremodule 9
      2.2.1. Leitrechner – Programm Rollrob 11
      2.2.2. Rollstuhl und Simulation – Programm Simroll 11
      2.2.3. Softwarekonzept für integrierten Leitrechner im Rollstuhl 12
   2.3. Konzeptüberblick 14
      2.3.1. Vorbereitung der Umgebung 14
      2.3.2. Ausstattung des Rollstuhls 15
         2.3.2.1. Hindernisdetektierung 16
         2.3.2.2. Leitliniensensoren 16
         2.3.2.3. Landmarkdetektor 17
         2.3.2.4. Shaft-Encoder – Bewegungsmessung 17
         2.3.2.5. Zusammenfassung der Rollstuhlausstattung 18
3. Selbstlokalisation 20
   3.1. Odometrie - Positionsverfolgung 21
      3.1.1. Grundlagen 21
      3.1.2. Kinematik eines differentialen Steuerungssystems 21
      3.1.3. Bestimmung der Odometriedaten 23
      3.1.4. Koppelnavigation – Aufrechnung der Odometrie im Leitrechner 25
   3.2. Globale Lokalisierung 30
4. Aufnahme der Umgebung 32
   4.1. Einlernen von Landmarken und Leitlinien - Teaching 32
      4.1.1. Konzept des Teachings 32
      4.1.2. Technischer Ablauf des Teachings 33
         4.1.2.1. Erreichen der ersten Landmarke 34
         4.1.2.2. Erreichen der zweiten Landmarke 34
         4.1.2.3. Bestimmung der Absolutausrichtung des Rollstuhls 35
         4.1.2.4. Erreichen der dritten Landmarke 37
   4.2. Korrektur der eingelernten Umgebung 39
   4.3. Korrektur der Rollstuhlposition und Orientierungsverlust 40
      4.3.1. Orientierungsverlust („Kidnapped Robot Problem“) 42
5. Hinderniserkennung 44
   5.1. Konzept der Hinderniserkennung 44
      5.1.1. Funkübertragung der Hindernisinformation 45
      5.1.2. Kollisionsvermeidung in der Rollstuhlssoftware 48
   5.2. Hindernisrepräsentation im Leitrechner 49
      5.2.1. Hindernisse eintragen 51
      5.2.2. Hindernisse austragen 52
6. Navigation 55

6.1. Erläuterung der umgesetzten Navigationsmethode 55
6.1.1. Neue Zieleingabe 57
6.1.2. Zur Leitlinie finden 57
6.1.3. Linienverfolgung 58
6.1.4. Hindernisbehandlung bei der Navigation 61
6.1.5. Weitere Navigationsverhalten 66
6.1.6. Übersicht über die Navigationskommandos an den Rollstuhl 67
6.2. Sicherheitsaspekte bei der Navigation 69
6.3. Alternative Navigationsmöglichkeit 71
6.3.1. Navigation nach dem Voronoi-Algorithmus 72
7. Simulation 76
7.1. Wiederholungsrate der Sensorberechnung 77
7.1. Shaft-Encoder 78
7.2. Hindernissensoren 80
7.3. Leitlinien-Sensoren 83
7.4. Landmarken-Detektor 86
8. Programm-Anleitung 88
8.1. Simulationsprogramm Simroll 88
8.1.1. Visuelle Darstellung im Simulator 89
8.1.2. Editieren der simulierten Umgebung 91
8.1.2.1. Leitlinien eingeben 92
8.1.2.2. Landmarken eingeben 93
8.1.2.3. Hindernisse eingeben 93
8.2. Leitrechnerprogramm Rollrob 95
8.2.1. User-Interface im Leitrechnerprogramm 96
8.2.2. Anleitung zum Teaching 98
8.2.3. Navigation und Zusatzfunktionen 101
9. Zusammenfassung und Ausblick 104
Literatur 107

1. Einleitung

1.1. Ziel und Umfang dieser Arbeit

In dieser Diplomarbeit wurde ein Systemkonzept erstellt und implementiert, mit welchem ein elektrischer Rollstuhl möglichst autonom in dafür vorbereiteten Räumlichkeiten navigieren kann. Der Mensch gibt dabei das gewünschte Ziel vor und der Rollstuhl soll dieses selbstständig, möglichst direkt und intelligent erreichen. Ein spezielles Augenmerk bei dem Systemdesign wurde auf die Robustheit und den Kostenfaktor der zu verwendenden Rollstuhlausstattung gelegt, damit dieses System auch wirtschaftlich realistisch umgesetzt werden kann.

Im Zuge dieser Arbeit wurde die Software für den Rollstuhl erstellt und eine Simulation entwickelt, um das entsprechende Verhalten und die räumliche Umgebung zu simulieren und um dadurch die Software testen zu können.

Die meisten in dieser Arbeit enthaltenen Teilgebiete entstammen dem Gebiet der Robotersteuerung, da ein semiautonomer Rollstuhl in seiner Funktion einem semiautonomen Roboter in einer speziellen Form entspricht.

1.2. Definition: Semiautonom versus autonom

Für die Begriffe „semiautonom“ und „autonom“ findet man eine Vielzahl von Definitionen. Für dieses Projekt wurde die Bezeichnung „semiautonom“ ausgewählt. Es soll damit ausgedrückt werden, dass der Rollstuhl im Gegensatz zu herkömmlichen mechanischen oder auch elektrischen Rollstühlen, nicht durch ständige Beeinflussung durch den Benutzer oder die Benutzerin gesteuert wird, sondern dass nur Ziele vorgegeben werden und der Rollstuhl diese durch Handlungen, die er selbst je nach äußeren Gegebenheiten setzt, erreicht.

Warum also semiautonom und nicht autonom? Wie bereits kurz erwähnt wurde, muss die Umgebung, in der sich der Rollstuhl bewegen soll, speziell vorbereitet werden. D.h. der Rollstuhl kann sich nicht in beliebigen Umgebungen zurechtfinden. Schon deshalb, weil die Benutzer und Benutzerinnen Probleme haben würden, in einer neuen Umgebung ihrem Wunsch Ausdruck zu verleihen, wenn dieser ein spezielles Ziel bezeichnen soll. Selbst wenn konkrete Befehle des Benutzers oder der Benutzerin erfolgen würden (z.B. Fahre ins Wohnzimmer), so müsste der Rollstuhl in einer völlig unvorbereiteten Umgebung zunächst erkennen können, was hier vermutlich das Wohnzimmer ist. Details zur Vorbereitung der Umgebung werden später genauer erläutert.

Die notwendige Anpassung der Umgebung und die dadurch entstehende Beschränktheit des Einsatzgebietes des Rollstuhls ist Hauptgrund für die Wahl der Bezeichnung „semiautonom“.

1.3. Zweck und Nutzen eines semiautonomen Rollstuhls

Warum einen semiautonomen Rollstuhl entwickeln und wer soll einen Nutzen aus diesem ziehen können?

Durch die Entwicklung der Mikroprozessortechnik wurden auch auf dem Gebiet der Steuerungsinterfaces von elektrischen Rollstühlen wesentliche Fortschritte gemacht. Zusätzlich zu den herkömmlichen Steuerungsmöglichkeiten über Joystick, Kinn- oder Kopfbewegungen haben Rollstuhlbenutzer(innen) mittlerweile die Möglichkeit, denselben zum Beispiel durch die Zunge oder durch die Bewegung der Augen zu steuern.

Doch selbst mit diesen modernen Steuerungstechniken ist es für eine Vielzahl von Menschen, die einen Rollstuhl benötigen, abhängig von der Art und der Schwere ihrer Beeinträchtigungen sehr schwer bzw. unmöglich, einen elektrischen Rollstuhl in einer typischen Umgebung wie z.B. der eigenen Wohnung zu steuern, da in dieser

die Bewegungsfreiheiten durch Möbel etc. oftmals deutlich eingeschränkt sind. Die Bewegungen des Rollstuhls müssen beim Ausweichen von Hindernissen oder beim Passieren von Türen sehr präzise sein.

Diese Tatsache hat dazu geführt, dass sich in den vergangenen zehn bis fünfzehn Jahren bereits einiges im Bereich der Forschung an sensorunterstützten Rollstuhlsteuerungen fortbewegt hat [GOM 98]. Ziel dabei ist es, durch die Zusammenarbeit von Sensoren und Mikroprozessoren, die Eingaben des Menschen zu interpretieren und je nach Gegebenheit der Umgebung, die Bewegungen des Rollstuhls anzupassen. Zum Beispiel lenkt der Mensch in Richtung einer Tür. Die feinen Steuerungsbewegungen, die erforderlich sind, damit sich der Rollstuhl durch dieselbe ohne anzustoßen hindurch manövrieren kann, werden jedoch von der Steuerungslogik automatisch berechnet und ausgeführt. [LEV 99, SIM 99]. Abbildung 1.1 zeigt als Beispiel eines sensorunterstützten Rollstuhls, das an der Universität von Michigan entwickelte Modell NavChair [GOM 98].

Abbildung 1.1: NavChair; Universität von Michigan

Um die Existenz von Personen die von solchen semiautonomen Systemen profitieren könnten besser dokumentieren zu können, wurde vom „Journal of Rehabilitation Research and Development“ eine Umfrage unter 200 Krankenhausbediensteten durchgeführt [FEH 00].

Die wesentlichen Ergebnisse dieser Umfrage waren die Folgenden:

- Die Befragten gaben an, dass neun bis zehn Prozent ihrer Patienten/innen, die ein Training für elektrische Rollstühle absolvierten, es extrem schwer bzw. unmöglich fanden, diesen Rollstuhl für Aktivitäten im täglichen Leben zu benutzen.
- Als die Patienten/innen speziell über Steuerung und Manövrierung befragt wurden, erhöhte sich der Prozentsatz derjenigen, die dieses sehr schwer oder unmöglich fanden, auf 40 %.
- Laut dieser Studie könnten fast die Hälfte aller Patienten/innen, die aufgrund von fehlenden motorischen oder visuellen Fähigkeiten nicht in der Lage sind, einen konventionellen elektrischen Rollstuhl zu bedienen, von einer autonomen (automatischen) Navigationsmethode profitieren.

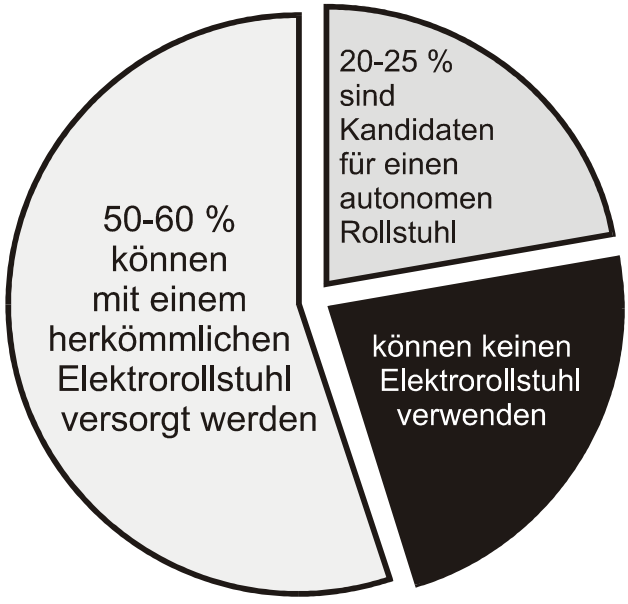


Abbildung 1.2: Kandidaten für einen autonomen Rollstuhl

Diese Ergebnisse zeigen auf, wie hoch der Bedarf von Entwicklungen unterstützender autonomer Navigationsmethoden für Rollstühle ist. Dabei reichen weitere Innovationen in die Steuerungs-Interfaces nicht aus.

2. Systembeschreibung

In diesem Kaptitel soll zunächst kurz auf die technischen Rahmenbedingungen eingegangen werden, um ein Grundverständnis für die folgenden Kapitel aufzubauen. Des Weiteren wird hier ein Gesamtüberblick über das Konzept des semiautonomen Rollstuhls gegeben. Es soll erläutert werden, welche Module über welche Schnittstellen miteinander kommunizieren.

In den späteren Kapiteln werden dann die einzelnen Hauptbereiche genauer erläutert und ihre technischen Umsetzungen aufgezeigt.

2.1. Einführung in die Simulation

Die entwickelte Steuerungssoftware wird auf einem ebenfalls im Zuge dieser Diplomarbeit entwickelten Simulator getestet. Eine detaillierte Funktionsbeschreibung des Simulators ist in Kapitel 7 zu finden. Der Simulator nimmt die Aktuatorenwerte, welche von der Rollstuhlsteuerung bestimmt werden, im Wesentlichen sind dies die Motoransteuerungen für die beiden Antriebsräder, und simuliert dadurch die Bewegungen des Rollstuhls. Diese werden grafisch dargestellt. Zusätzlich errechnet der Simulator entsprechend der Position und der Bewegung des Rollstuhls in der simulierten Umwelt die jeweiligen Sensorwerte und liefert diese wieder an das Rollstuhlsteuerungsprogramm, welche diese auswertet und entsprechend dem Ergebnis seine nächsten Handlungen setzt.

Sämtliche Stellgrößen können bei der Simulation durch einfaches Adaptieren von Konstanten im Programm verändert werden. D.h. Sensorreichweiten, Rollstuhlmaße oder auch die Anzahl von bestimmten Sensoren kann im Programm variiert werden. Dadurch können unterschiedliche Einstellvarianten relativ einfach getestet und verglichen werden.

Der Simulator ersetzt die reale Aktorik und Sensorik (siehe Abb. 2.1).

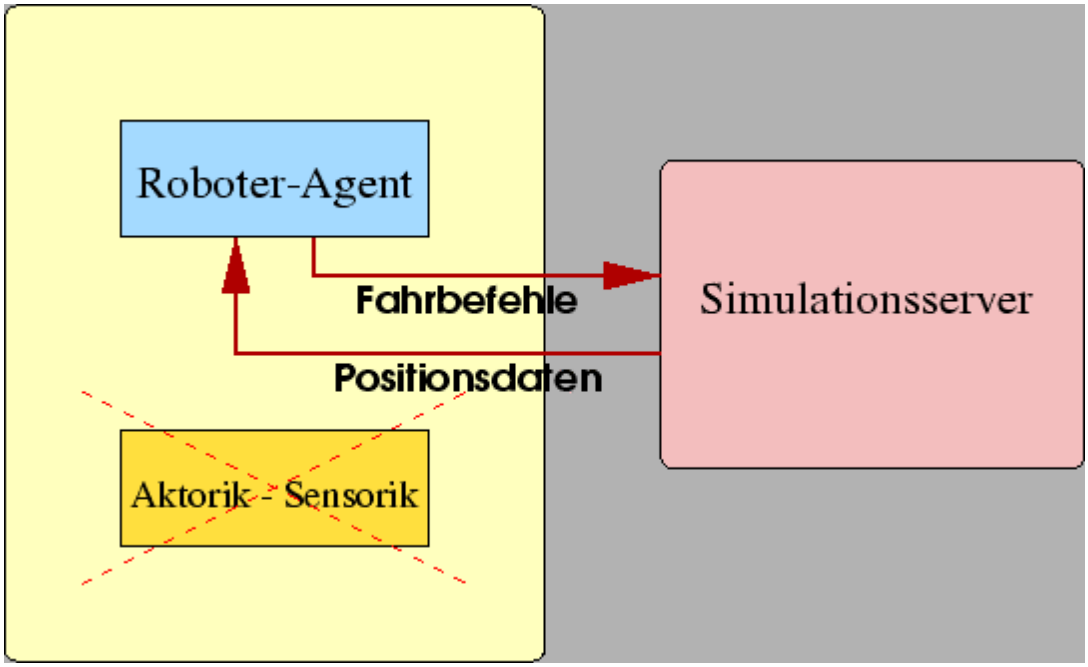


Abbildung 2.1: Gewünschte Situation

Ein spezielles Augenmerk bei der Umsetzung der Simulation wurde darauf gelegt, dass eine Umstellung von der Simulation zur Realität möglichst einfach durchzuführen ist [RÖF 98_2].

2.1.1. Vorteile einer Simulation

Eine Simulation bietet folgende Vorteile:

1.) Die Simulation erlaubt ein schnelles und effizientes Ausführen und Testen von verschiedenen Steuerungsalgorithmen. Das bedeutet, es können verschiedene Lösungswege relativ einfach getestet werden. Selbstverständlich muss dafür die Software erstellt werden, jedoch ist es nicht erforderlich, einen realen Rollstuhl oder ein Modell desselben bei jeder neuer Umsetzungsstrategie mit den entsprechend erforderlichen Sensoren auszustatten. Stattdessen müssen jedoch im Simulationsprogramm

entsprechende Berechnungsmethoden implementiert werden, um die jeweiligen Sensorwerte bestimmen und bereitstellen zu können. Um auch den Aufwand für Letzteres zu senken, wurde der Simulator so umgesetzt, dass die Parameter der einzelnen Sensoren einfach abänderbar sind.

Gleichzeitig ist die Geschwindigkeit eines Testlaufes am Computer selbstverständlich wesentlich schneller als ein Testlauf mit einem realen Rollstuhl in einer realen Umgebung. Außerdem können so auch Testläufe außerhalb von eigens dafür vorbereiteten Testumgebungen durchgeführt werden.

2.) Hardwareeinflüsse, wie zum Beispiel ungenaue Sensordaten, temporär auftretende Defekte oder durch Spannungsschwankungen hervorgerufene Phänomene machen eine Fehlersuche in der Software oftmals zeitaufwändig bzw. schwierig bis unmöglich.

Durch die Simulation können komplexere Softwarealgorithmen, wie zum Beispiel Ausweichverfahren oder Routenplanungen vorab in einer „idealen“ Umgebung, in der gewisse Störfaktoren ausgeschaltet werden können, auf ihre prinzipielle Funktionsfähigkeit und Tauglichkeit getestet werden.

3.) Fehler, die unter Umständen zu selbstzerstörenden Manövern führen könnten, können durch die Simulation oftmals rechtzeitig und ohne tatsächliche Materialverluste erkannt werden. Es können jedoch selten alle Fehler durch die Simulation gefunden werden, da bestimmte Fehler erst unter realistischen Bedingungen zum Tragen kommen (siehe dazu unter 1.3.2. Nachteile einer Simulation).

4.) Durch die programmtechnische Generierung und Simulation der Bewegungen und Ereignisse kann eine nachträgliche Analyse erfolgen. Durch die Auswertung der gespeicherten Informationen kann z.B. beim Auftreten eines ungewöhnlichen oder fehlerhaften Verhaltens einfacher überprüft werden, wodurch dieses ausgelöst wurde. Um diesen Punkt effektiv erfüllen zu

können, muss der Simulator jedoch verschiedene Datenaufbereitungen bzw. Auswertungsalgorithmen aufweisen, um aus der Vielzahl von Daten sinnvolle Ergebnisse aufbereiten zu können.

2.1.2. Nachteile einer Simulation

Der Nachteil einer Simulation ist schnell gefunden. Er liegt in der Abweichung zwischen der Simulation und der Realität, welche wohl auch bei einer noch so detaillierten Simulation niemals gänzlich auszuschalten ist.

Wie groß dieser Nachteil ausfällt, hängt sehr stark vom Typ des zu simulierenden Objektes ab. So wird ein Hubschrauber mit allen seinen aerodynamischen Eigenschaften nur schwierig mit einem Simulator korrekt abgebildet werden können. Eine Steuerung, die in der Simulation einwandfrei funktioniert, kann in der Realität angewendet kläglich scheitern.

Auch wenn die Simulation eines Rollstuhls wesentlich einfacher ist, so wird die exakte Übereinstimmung der Reaktionen in der realen Umgebung und der Umgebung der Simulation nicht erreicht werden können. Die Realität hat sehr viele Einflussgrößen, welche in ihrer Komplexität nicht alle simuliert werden können.

2.2. Softwaremodule

Die gesamte Software wurde in Borland Delphi 5 unter Windows 2000 entwickelt. Die Software besteht aus zwei getrennten Programmen, die jeweils mehrere Tasks beinhalten, welche nebeneinander ablaufen. Das erste Programm stellt den Leitrechner dar, das zweite Programm beinhaltet die Simulation und die Software des Rollstuhls.

Bei der Konzeption und Implementierung der Softwaremodule wurde prinzipiell Augenmerk darauf gelegt, dass die Lösung zunächst auf einem Modell eines

Rollstuhls zur Anwendung gebracht werden soll. Im Falle eines Modells kann die komplexe Software des Leitrechners mit seinem User-Interface nicht direkt auf dem Rollstuhl laufen, da dazu ein PC erforderlich ist. Stattdessen läuft am Modell nur eine minimale Softwarelösung, welche über Funk mit dem Leitrechner kommuniziert (siehe Abb. 2.2).

Aus diesem Grund wurde die Software für Leitrechner und Rollstuhl getrennt. Durch die erforderlichen Funkübertragungen entstehen aber natürlich auch Engpässe bezüglich der Datenübertragungsrate und Bandbreite. Unter dem Punkt 2.2.3. wird kurz angesprochen, wie das Systemkonzept aussehen würde, wenn der Leitrechner direkt auf dem Rollstuhl angebracht wird. Dies ist bei der Umsetzung auf einem realen Rollstuhl anzustreben, da dann direkt auf die Sensorwerte zugegriffen werden kann, ohne dass ein vorheriges Übertragen per Funk erforderlich ist.

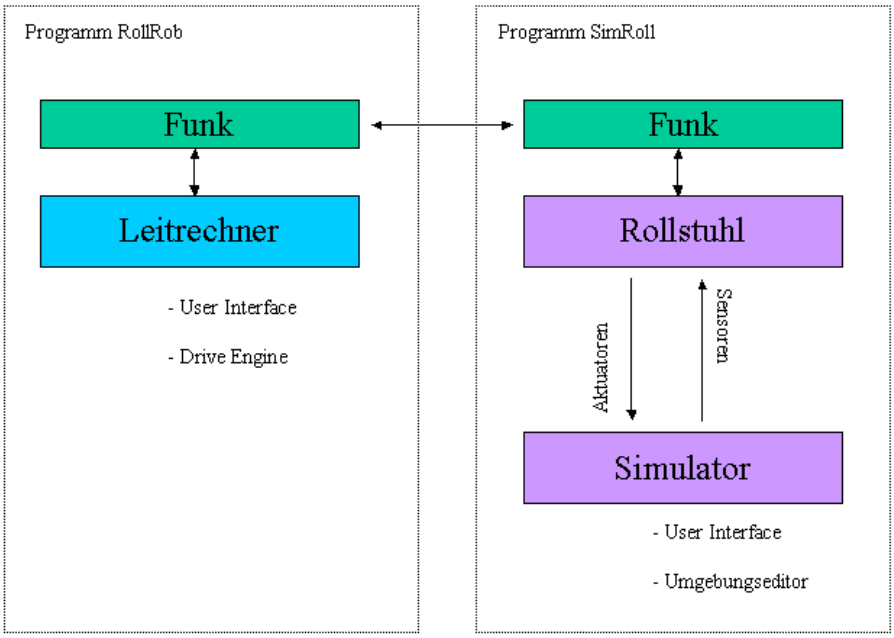


Abbildung 2.2: Modulaufbau

2.2.1. Leitrechner – Programm Rollrob

Das Programm Rollrob beinhaltet den Kern des autonomen Rollstuhls und besteht aus mehreren Prozessen, die nebeneinander ablaufen. Ein Prozess ist für die Funkkommunikation mit dem Rollstuhl zuständig. Die „Drive Engine“ als weiterer Prozess ist für sämtliche Berechnungen zuständig. Hier werden die empfangenen Daten des Rollstuhls ausgewertet und entsprechende Handlungen bestimmt. Die daraus resultierenden Steuerungskommandos werden dann wiederum über die Funkschnittstelle an die Rollstuhlsoftware gesendet. Sämtliche Aspekte der Navigation und der Umgebungswahrnehmung finden im Leitrechner statt.

Da der Rollstuhl nur simuliert wird, findet in der Funkschnittstelle momentan keine reale Funkübertragung statt. Stattdessen findet die Kommunikation zwischen den Programmen Rollrob und Simroll über Interprozesskommunikation statt. D.h. die Daten werden zwischen den Programmen hin und her gesendet. Die Schnittstellen wurden jedoch so implementiert, dass diese auch bei Umstellung auf realen Funkverkehr genauso verwendet werden können. Nur die unterste Ebene der Übertragung muss dazu ausgetauscht werden.

Zuletzt befindet sich in diesem Programm die Benutzerschnittstelle ebenfalls als eigenständiger Prozess, mit der Darstellung der momentanen Position, der erkannten Hindernisse und der Eingabe der gewünschten Ziele durch den Benutzer oder die Benutzerin.

2.2.2. Rollstuhl und Simulation – Programm Simroll

In diesem Programm läuft die Software des Rollstuhls ab. Wenn anstatt der Simulation ein realer Rollstuhl bzw. ein Modell verwendet werden soll, so ist dies derjenige Teil, der direkt am Rollstuhl ablaufen muss. Hier werden Bewegungsbefehle vom Leitrechner empfangen und entsprechend die Aktuatoren (Motoren) angesteuert. Des Weiteren finden hier grundsätzliche Sicherheitsmechanismen statt, um zum Beispiel zu verhindern, dass der Rollstuhl an

ein Hindernis anfährt. Die Sensordaten werden vorverarbeitet und an den Leitrechner übertragen.

Der Rollstuhl muss ein Fail-Safe Verhalten aufweisen, d.h. in kurzen Worten erklärt: Sollte der Leitrechner ausfallen oder fehlerhafte Befehle senden, so muss der Rollstuhl trotzdem in einen sicheren Zustand geführt werden. Im konkreten Fall wird er automatisch anhalten.

Das Simulationsmodul ist ebenfalls in diesem Programm enthalten. Es wird eine Sensor/Aktuator-Schnittstelle zur Verfügung gestellt. Der Rollstuhlprozess schreibt die Aktuatorenwerte in diese Schnittstelle und liest daraus die Sensorwerte, welche die Simulation ständig berechnet und dort bereitstellt.

Zusätzlich ist das User-Interface der Simulation im Programm SimRoll integriert. Zum Einen kann hier das Design der Umwelt vorgenommen werden, d.h. Räume können erstellt werden, Hindernisse können platziert werden sowie Leitlinien und Landmarken können positioniert werden. Des Weiteren wird während der laufenden Simulation jeweils die Position des Rollstuhls in der Umgebung dargestellt. Dies entspricht der tatsächlichen Position des Rollstuhls. Demgegenüber wird im User-Interface des Leitrechners die errechnete Position des Rollstuhls dargestellt. D.h. hier kann ein direkter Vergleich gezogen werden.

2.2.3. Softwarekonzept für integrierten Leitrechner im Rollstuhl

Wie bereits erwähnt wurde, ist das hier umgesetzte Systemkonzept dafür ausgelegt, dass sich der Leitrechner nicht direkt „onboard“ am Rollstuhl befindet. Dies ist für die Umsetzung auf einem Modellrollstuhl mit beschränkter Größe zwingend erforderlich. Schon alleine deswegen, weil am Modell keine Person mitfahren kann und die Person daher dem User-Interface nachlaufen müsste. Deshalb

kommunizieren Rollstuhl und Leitrechner per Funk bzw. über eine simulierte Funkschnittstelle miteinander.

Bei der Umsetzung auf einem realen Rollstuhl ist jedoch diese Funkkommunikation nicht erforderlich, da der Leitrechner inklusive User-Interface direkt am Rollstuhl angebracht sein wird. Durch das Einsparen der Funkschnittstelle würden dann gewisse Vorteile aufgrund der Möglichkeit des direkten Zugriffs auf die Sensorwerte entstehen.

So könnten dann die in dieser Arbeit umgesetzten Strategien zur Kompensation der Zeitspanne zwischen Aufnahme des Sensorwertes durch den Rollstuhls bis zum Eintreffen desselben Wertes beim Leitrechner entfallen. Des Weiteren könnten größere Datenmengen vom Leitrechner ausgewertet werden, da diese nicht per Funk übertragen werden müssen. Dies würde vor allem beim Auswerten der Hindernisinformationen zu Vorteilen führen, wie unter dem Punkt 5.1.1.: „Funkübertragung der Hindernisinformation“ ersichtlich werden wird.

Die Zusammenfassung von Leitrechner und Rollstuhlsteuerung würde somit einerseits einige Softwarestrukturen überflüssig machen, als auch teilweise verbesserte Möglichkeiten bieten, um auf bestimmte Situationen reagieren zu können.

2.3. Konzeptüberblick

Bevor in den folgenden Kapiteln die einzelnen Teilbereiche genauer erläutert werden, soll hier zunächst ein grober Überblick über das in dieser Arbeit umgesetzte Systemkonzept eines semiautonomen Rollstuhls gegeben werden.

2.3.1. Vorbereitung der Umgebung

In der Umgebung, in der sich der Rollstuhl bewegen soll, sind gewisse Vorbereitungen zu tätigen. Zunächst werden Leitlinien am Boden der Räumlichkeiten verlegt. Bei jedem Kreuzungspunkt einer Leitlinie und bei jedem Ziel, das erreichbar sein soll, wird dann eine sogenannte Landmarke, zum Beispiel in Form eines Barcodes, angebracht. Die Leitlinien dürfen ausschließlich aus geraden Linien bestehen, da wie später erläutert wird, mit deren Hilfe die Ausrichtung des Rollstuhls durch den Leitrechner berechnet wird.

Jeder Landmarke kann eine Bezeichnung zugeordnet werden, mit deren Hilfe der Mensch schließlich angeben kann, wohin er gerne fahren möchte. Der Rollstuhl wird dann selbständig versuchen, dieses Ziel zu erreichen. Eines der Grundkonzepte ist es, dass es nicht erforderlich ist, den detaillierten Aufbau der Umgebung mit den genauen Maßen und Winkeln in das System einzugeben. Stattdessen nimmt der Rollstuhl die Umgebung während der Bewegung durch diese aktiv auf. Wie die Landmarken und die Leitlinien dem System eingelernt werden, wird unter dem Punkt 4.1. Teaching genauer erläutert.

Die Leitlinien sollen so gelegt werden, dass der Rollstuhl diese ungehindert entlangfahren kann, wenn sich der Mittelpunkt seiner Hinterachse direkt über der Linie befindet.

Dadurch, dass die Linien nur aus Geraden bestehen, kann Letzteres eventuell etwas aufwändig sein. Wie man später sehen wird, stellt es aber auch kein wesentliches Problem dar, wenn der Rollstuhl nicht ungehindert einer Linie folgen kann, da

entsprechende Algorithmen umgesetzt wurden, welche es dem Rollstuhl ermöglichen, eine Linie selbstständig zu verlassen, um Hindernissen auszuweichen.

Trotz allem sollten nach Möglichkeit beim ersten Design, z.B. einer Wohnung, zunächst freie Linien verlegt werden. Hindernisse entstehen in späterer Folge zumeist von alleine (zum Beispiel ein neuer Schuhkasten im Vorraum, etc.).

Eine mögliche Vorbereitung einer Wohnung wird in Abbildung 2.3 dargestellt.

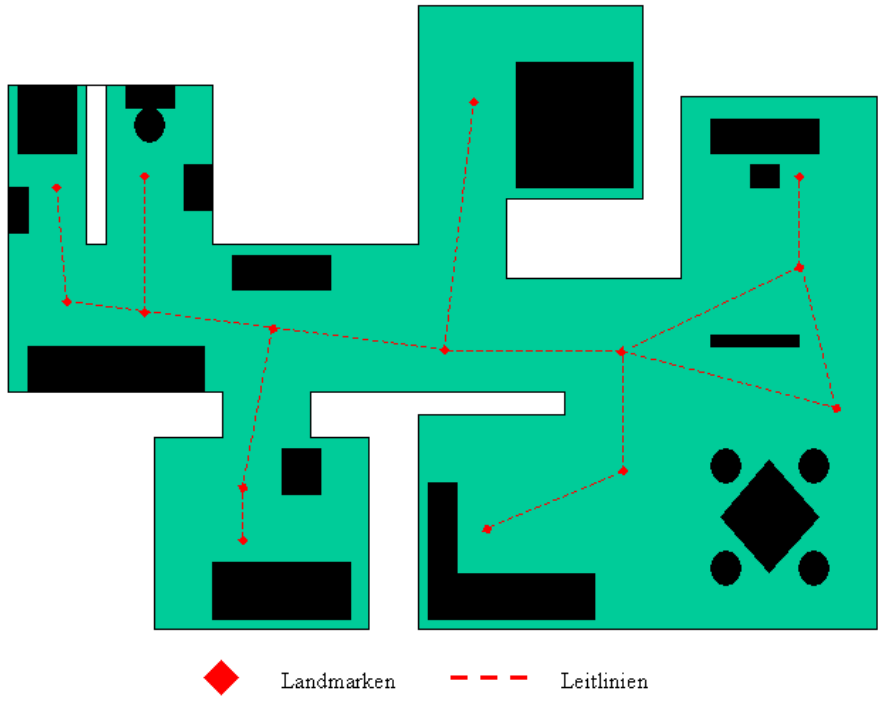


Abbildung 2.3: Vorbereitung einer Wohnung mit Leitlinien und Landmarken

2.3.2. Ausstattung des Rollstuhls

Das Konzept ist auf einen Rollstuhl mit hinteren Antriebsrädern, die getrennt voneinander angetrieben werden, ausgelegt. Die Lenkbewegungen des Rollstuhls

werden durch das Setzen von unterschiedliche Geschwindigkeiten der beiden Antriebsräder erreicht.

Die Sensorenausstattung des Rollstuhls ist in Abbildung 2.4 ersichtlich und wird im Folgenden erklärt.

2.3.2.1. Hindernisdetektierung

Der Rollstuhl besitzt an der Vorderseite einen Sensor zur Hindernisdetektierung, welcher von der Mitte ausgehend, in der Form eines Kreissegments nach Hindernissen scannt. Bei der Umsetzung auf einem realen Rollstuhl muss aus Sicherheitsgründen auch nach hinten und zur Seite nach Hindernissen gescannt werden. Für die Konzeption des autonomen Verhaltens sind diese Messungen jedoch nicht erforderlich.

2.3.2.2. Leitliniensensoren

An der Unterseite des Rollstuhls befinden sich in der Mitte der Hinterachse zwei Sensoren zur Detektierung der Leitlinie.

Bei der Simulation wurden gute Ergebnisse bei der Linienverfolgung mit folgenden geometrischen Verhältnissen erzielt:
Die Sensoren werden ausgehend vom Hinterachsenmittelpunkt jeweils um die halbe Breite der Leitlinie nach links und nach rechts versetzt angebracht. Der Durchmesser der Sensoraufnahmefläche entspricht der Breite der Leitlinie.

Bei der Umsetzung wurde davon ausgegangen, dass mittels analoger Sensoren am Rollstuhl messbar ist, wie weit sich der Sensor über einer Linie befindet. D.h. es muss unterscheidbar sein, ob zum Beispiel der gesamte Sensor über der Linie positioniert ist, oder ob sich nur eine Hälfte des Sensors über und die andere Hälfte neben der Linie befindet. Diesen Anforderungen würde eine lichtreflektierende

Linie, also zum Beispiel weißes Klebeband mit dazupassenden Infrarotsensoren am Rollstuhl entsprechen. Es können bei der realen Umsetzung aber natürlich auch andere detektierbare Materialien verwendet werden, die diese Anforderungen erfüllen. Eine solche Alternative ist in [WAK 92] beschrieben. Hierbei besteht die Leitlinie aus magnetischem Material und anstatt der Infrarotsensoren werden zwei analoge Magnetdetektoren angebracht. Der Vorteil einer magnetischen Leitlinie liegt darin, dass typische Störfaktoren der Infrarotlösung wie zum Beispiel Umgebungslicht oder Verschmutzung der Leitlinie durch diese Methode ausgeschaltet werden.

2.3.2.3. Landmarkdetektor

Unmittelbar vor den Sensoren zur Linienverfolgung befindet sich der Bereich der Landmarkendetektierung.
Die Landmarken müssen eindeutig identifizierbar sein. Diese Anforderung kann zum Beispiel durch das Anbringen von Barcodes (vgl. [JÖR 99]) auf den Leitlinien erfüllt werden.

2.3.2.4. Shaft-Encoder – Bewegungsmessung

Für die Aufnahme und Überwachung der Rollstuhlbewegung genügt es nicht, sich die Bewegung aus den Ansteuerungen der Antriebsräder zu errechnen. Zum Einen wäre dies durch Beschleunigungsverhalten, Motorleistungsschwankungen, etc. viel zu ungenau und des Weiteren kann der Rollstuhl ja auch bewegt werden, ohne dass der Leitrechner eine Bewegung veranlasst hat.

Um die Bewegungen des Rollstuhls genauer zu erfassen und rückzukoppeln, wird daher sowohl für das rechte als auch das linke Antriebsrad ein sogenannter Shaft-Encoder eingesetzt. Die Shaft-Encoder messen die tatsächlichen Bewegungen der Antriebsräder. Dabei werden Impulse über schwarz/weiß kodierte Scheiben oder

Lochmasken durch Reflexionslichtschranken gezählt. Diese Impulse werden in weiterer Folge als Radimpulse bezeichnet.

Um die erforderliche Feinheit der Messungen zu erreichen, werden die Scheiben gewöhnlich direkt am Getriebe und nicht auf der Achse angebracht, da Zahnräder des Getriebes eine wesenlicht höhere Umdrehungszahl aufweisen als die Antriebsachse, welche durch das Getriebe bewegt wird. D.h. bei einer Umdrehung eines Rades werden eine fixe Anzahl von Impulsen gezählt. Aus diesen gelieferten Impulsen kann daher der zurückgelegte Weg des jeweiligen Rades errechnet werden. Diese Impulse stellen die Basis für die unter Punkt 3.1. erläuterten Odometrieberechnungen dar.

Bei der Umsetzung auf einem Modell oder einem lebensgroßen Rollstuhl muss berücksichtigt werden, dass die Impulse alleine nicht ausreichen. Ebenso wie bei der Simulation, muss auch ohne Simulator unterschieden werden können, in welche Richtung sich das jeweilige Rad dreht. Durch das reine Zählen von Impulsen kann die Richtung aber nicht unterschieden werden. Es müssen also pro Antriebsrad/Welle zwei Shaft-Encoder verwendet werden. In diesem Fall kann durch Berücksichtigung der Phasenverschiebung die Bewegungsrichtung bestimmt werden.

2.3.2.5. Zusammenfassung der Rollstuhlausstattung

Zusammengefasst liefern die am Rollstuhl angebrachten Sensoren folgende Informationen:

- Sensormesswert des linken und des rechten Liniensensors (jeweils Werte zw. 0 und 255 je nach Überschneidung einer Leitlinie mit der Sensoraufnahmefläche; 255 steht für keinerlei Detektierung).

- Die ID einer Landmarke, wenn eine solche durch den Landmarkdetektor erkannt wurde.

- Den Winkel und den Abstand von erkannten Hindernispunkten im Scanbereich

- Die jeweiligen Radimpulse der Shaft-Encoder für das linke als und das rechte Antriebsrad.

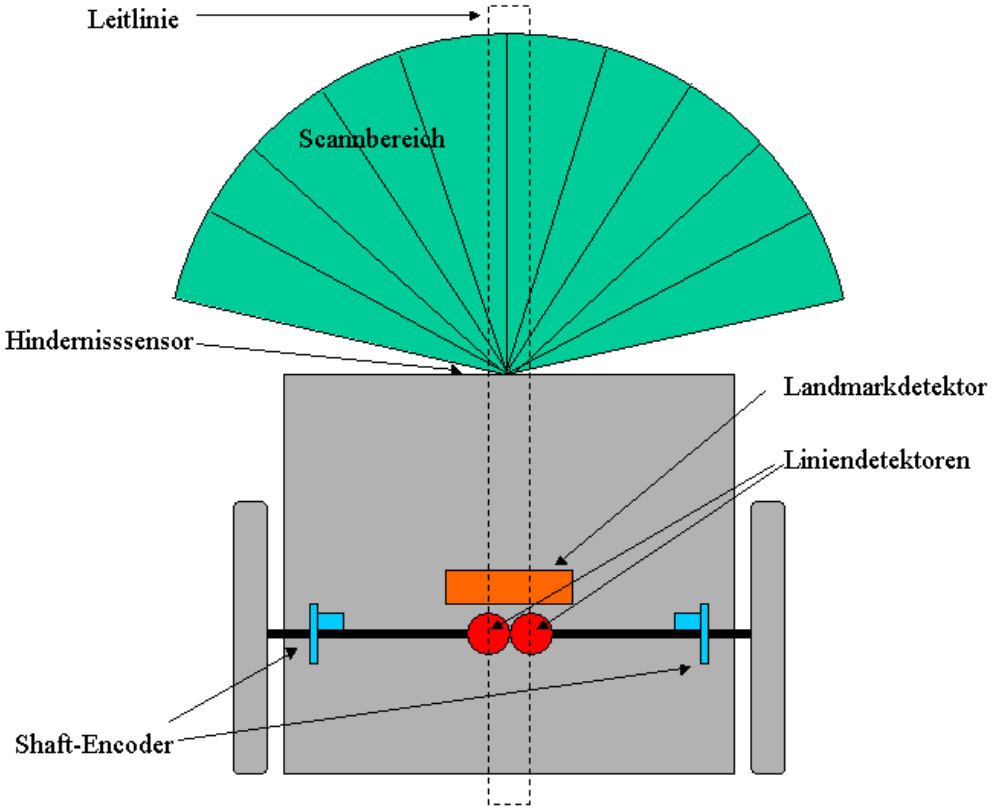


Abbildung 2.4: Ausstattung des Rollstuhls

3. Selbstlokalisation

Die Fähigkeit zur Selbstlokalisation [FOX 98, THR 00, FOX 01] d.h. zur ständigen Bereitstellung einer möglichst akkuraten Schätzung der eigenen Position und Ausrichtung in der Einsatzumgebung, stellt eine Grundlage jeder Navigation dar.

Das einfachste Lokalisierungsproblem, welches in der Literatur die größte Aufmerksamkeit erhalten hat, stellt die lokale Selbstlokalisation dar, welche auch als Positionsverfolgung bezeichnet wird. Dabei wird davon ausgegangen, dass die Startposition des Roboters bzw. des Rollstuhls bekannt ist. Durch Aufzeichnung und Aufrechnung der ausgeführten Bewegungen ist stets die momentane Position bestimmbar, dies wird als Koppelnavigation bezeichnet und wird unter Punkt 3.1.4. näher erläutert.

Etwas komplexer ist das sogenannte globale Lokalisierungsproblem. Dabei ist die Ursprungsposition unbekannt. Der Rollstuhl muss durch geeignete Algorithmen in der Lage sein, die Position selbst zu bestimmen.

Für das in dieser Arbeit umgesetzte Konzept müssen diese beiden Lokalisierungsarten implementiert werden.

Lösungsansatz hierfür stellt ein Verfahren dar, welches zum Einen auf der Positionsverfolgung basiert, und zum Anderen die fixen Landmarken in der Umgebung als Korrektur bzw. Ausrichtungspunkte verwendet.

Unter dem folgenden Punkt 3.1. wird erläutert, wie die Position des Rollstuhls durch Odometrie im Leitrechner aktualisiert wird. Unter dem Punkt 3.2. wird dann die Lösung des globalen Lokalisierungsproblems erklärt.

3.1. Odometrie - Positionsverfolgung

3.1.1. Grundlagen

Unter Odometrie [RÖF 98, LAN 01] versteht man die Orts- und Richtungsangabe eines Fahrzeuges zu einem Referenzpunkt und einer Referenzrichtung. Wurde zum Zeitpunkt X der Referenzpunkt mit 0/0 und die Referenzrichtung mit 0 Grad initialisiert, so kann zu einem späteren Zeitpunkt Y die relative Bewegung seit dem Zeitpunkt X abgefragt werden.

Um nun die Absolutposition und Absolutrichtung des Rollstuhls zum Zeitpunkt Y angeben zu können, müssen zu den Absolutdaten zum Zeitpunkt X die relativen Odometriedaten hinzugerechnet werden.

Die relativen Odometriedaten werden aus den Informationen der beiden Shaft-Encoder an den Antriebsrädern errechnet.

Wie unter 2.3.2.4. erwähnt wurde, können aus den Informationen der Shaft-Encoder die zurückgelegten Wegstrecken des linken und rechten Rades bestimmt werden. Die Rollstuhlsteuerung wertet diese Informationen kontinuierlich aus und errechnet mit Hilfe der Kinematik eines differentialen Steuerungssystems die jeweilige relative Position und Ausrichtung des Rollstuhls.

3.1.2. Kinematik eines differentialen Steuerungssystems

Ein Rollstuhl des hier simulierten Modells folgt den Regeln der Kinematik eines differentialen Steuerungssystems [ALE 88, LUC 01]. Diese erlauben die Berechnung von Position und Ausrichtung aufgrund des zurückgelegten Weges zweier unabhängiger Antriebsräder, die auf einer Achse liegen und sowohl für die Antriebs- als auch Lenkbewegungen zuständig sind.

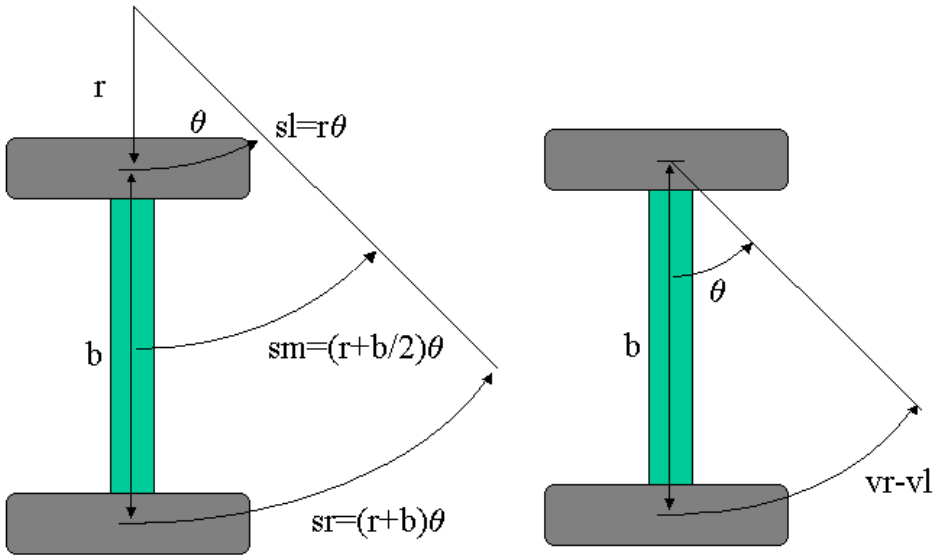


Abbildung 3.1: Kinematik; links der Weg der beiden Räder bei einer Drehung; rechts die Betrachtung des Geschwindigkeitsunterschiedes der Räder

Bewegen sich beide Antriebsräder des Rollstuhls mit der gleichen Geschwindigkeit, so fährt der Rollstuhl eine gerade Linie. Sobald sich ein Rad schneller dreht als das andere, fährt der Rollstuhl eine Kurve auf einer Kreisbahn (bei konstanten Geschwindigkeiten beider Räder).

Abbildung 3.1a zeigt den zurückgelegten Weg der Räder bei einer Drehung des Rollstuhls. Hieraus ergeben sich folgende Formeln, um die erforderlichen Wege und Geschwindigkeiten der beiden Räder zu errechnen, damit eine spezifische Bewegung erreicht wird:

$$sl = r\theta$$
$$sr = (r+b)\theta$$
$$sm = (r+b/2)\theta$$
$$v = s/t$$

r entspricht dabei dem Abstand zum Kreismittelpunkt, um den die Kreisbahn gefahren werden soll. θ entspricht dem zu fahrenden Kreiswinkel im Bogenmaß. b entspricht dem Abstand zwischen den beiden Antriebsrädern.

Für die Berechnung der Odometrie liegt das Interesse nicht an der Berechnung der erforderlichen Geschwindigkeiten der Räder, sondern umgekehrt in der Veränderung der Position und Ausrichtung bei bestimmten Radgeschwindigkeiten, bzw. wenn bekannt ist, welche Wege die einzelnen Räder zurückgelegt haben.

Hierzu wird nur der Geschwindigkeitsunterschied beider Antriebsräder herangezogen, dies wird in Abbildung 3.1b veranschaulicht. Durch das Lösen der sich daraus ergebenden Differentialgleichungen ergeben sich die folgenden Formeln:

ϑ_0, x_0, y_0 sind Ausrichtung sowie x,y Position des Rollstuhls zum Zeitpunkt vor den neuen Radbewegungen.

Für die neue Ausrichtung ergibt sich folgende Formel:

$$\vartheta(t) = (vr - vl)t / b + \vartheta_0$$

Für die neue Position des Achsenmittelpunkt ergibt sich:

$$x(t) = x_o + \frac{b(vr + vl)}{2(vr - vl)}[\sin((vr - vl)t / b + \vartheta_0) - \sin(\vartheta_0)]$$
$$y(t) = y_o + \frac{b(vr + vl)}{2(vr - vl)}[\cos((vr - vl)t / b + \vartheta_0) - \cos(\vartheta_0)]$$

3.1.3. Bestimmung der Odometriedaten

Die obigen Formeln der Kinematik sind nur dann korrekt anwendbar, wenn es sich bei der Bewegung um eine gleichförmige handelt. D.h. linkes und rechtes Rad haben

während der aufzurechnenden Bewegung annähernd konstante Geschwindigkeiten aufgewiesen.

Dies ist nur dadurch zu erreichen, dass die Odometrieberechnungen aufgrund der Radbewegungen in sehr kurzen Zeitabständen (mind. 10x pro Sekunde) durchgeführt werden. Dadurch wird der verursachte Fehler sehr klein.

Beispiel für einen Fehler durch zu große Zeitabstände zwischen den Odometrieberechnungen: Würde sich bei einer Bewegung das linke Rad 10cm bewegen und das rechte Rad stillstehen und im Anschluss daran das rechte Rad 10cm fahren und das linke Rad stillstehen, so wären die zurückgelegten Wege jeweils 10cm. Wenn man erst jetzt die Odometrieposition mit Hilfe der obigen Formeln der Kinematik durchführen würde, so ist das Ergebnis eine gerade Linie nach vorne. In Wahrheit hat sich der Rollstuhl jedoch nach rechts vorne bewegt. Die Ausrichtung ist zwar korrekt, aber die Position im Raum ist eine völlig andere. Die Berechnungen müssen also nach der ersten und der zweiten Teilbewegung durchgeführt und aufgerechnet werden, um zur korrekten Position zu führen.

Durch diese Notwendigkeit der Berechnung in sehr kurzen Zeitabständen ist es auch nicht möglich, die Radbewegungen jeweils an den Leitrechner zu übertragen, damit dieser die Aufrechnung übernimmt. Es würde nämlich durch die eingeschränkte Übertragungsrate und Bandbreite bei der Funkübertragung das Zeitintervall zu groß werden.

Die Berechnung der Odometrie übernimmt daher direkt die Rollstuhlsoftware. Zu Beginn wird eine Startposition von 0/0 und eine Ausrichtung von 0 Grad im Rollstuhl gesetzt. Mehrmals pro Sekunde werden die durch die Shaft-Encoder gelieferten Radbewegungen mittels der Formeln der Kinematik in Positionsänderung und Ausrichtungsänderung umgerechnet und aufaddiert.

Zweimal pro Sekunde sendet der Rollstuhl diese vorverarbeiteten Odometriedaten an den Leitrechner und setzt die Odometriereferenz wieder auf Position 0/0 mit Ausrichtung 0 zurück. Er berechnet die Bewegung also ab diesem Zeitpunkt relativ

zu dieser Übertragung. Dies ist erforderlich, da der Leitrechner die Absolutposition beim Erreichen einer Landmarke korrigieren kann.

Die Funkschnittstelle gewährleistet, dass jede einzelne relative Bewegungsinformation auf jeden Fall beim Leitrechner ankommt, da sobald eine Übertragung verloren gehen würde, die errechnete Position des Rollstuhls in weiterer Folge falsch wäre.

3.1.4. Koppelnavigation – Aufrechnung der Odometrie im Leitrechner

Im Leitrechner werden die relativen Bewegungen, welche über Funk von der Rollstuhlsoftware jede halbe Sekunde empfangen werden, aufaddiert und somit die Absolutposition des Rollstuhls aktualisiert.

Zum besseren Verständnis ist in Abbildung 3.2 die programminterne Repräsentation des Rollstuhls bezogen auf das Koordinatensystem dargestellt. Eine Orientierung von 0 Grad entspricht dabei einer Ausrichtung in die positive x-Achse. Die Position des Rollstuhls wird immer durch den Mittelpunkt zwischen den beiden Antriebsrädern repräsentiert.

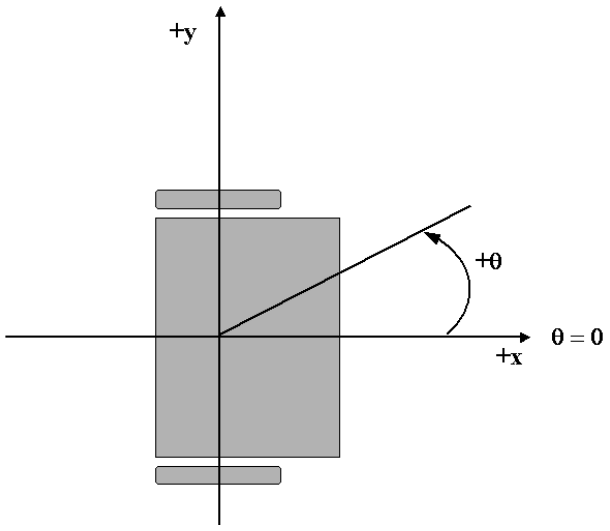


Abbildung 3.2: Interne Repräsentation des Rollstuhls

Es sei an dieser Stelle vorweggenommen, dass im Rollstuhl zwei Positionen des Rollstuhls parallel geführt werden. Die eine Position ist eine relative Position, die zu Programmbeginn mit 0/0 und Ausrichtung 0 initialisiert wird. Hier werden fortan immer die vom Rollstuhl gelieferten Odometriedaten hinzugerechnet. Diese Position wird niemals korrigiert und beinhaltet also auch sämtliche Odometriefehler. Diese Position wird zur Bestimmung und Korrektur von Position und Lage der Leitlinien benötigt; dies wird in Kapitel 4 ausführlich erläutert.

Die zweite Position ist die Absolutposition des Rollstuhls. Diese wird immer dann korrigiert, wenn durch das Erreichen einer Landmarke die tatsächliche Rollstuhlposition bestimmt werden kann. Dies verhindert, dass sich Odometriefehler aufsummieren. Ansonsten werden auch hier die jeweiligen Odometriebewegungen hinzuaddiert, um auch nach dem Verlassen einer Landmarke die Absolutposition bestimmen zu können. Genaueres dazu folgt unter Punkt 4.3.: Korrektur der Rollstuhlposition.

Im Folgenden wird dargestellt, wie eine relative Bewegungsinformation der Rollstuhlsoftware auf die bisherige Absolutposition aufgerechnet wird, um die neue Absolutposition zu bestimmen.

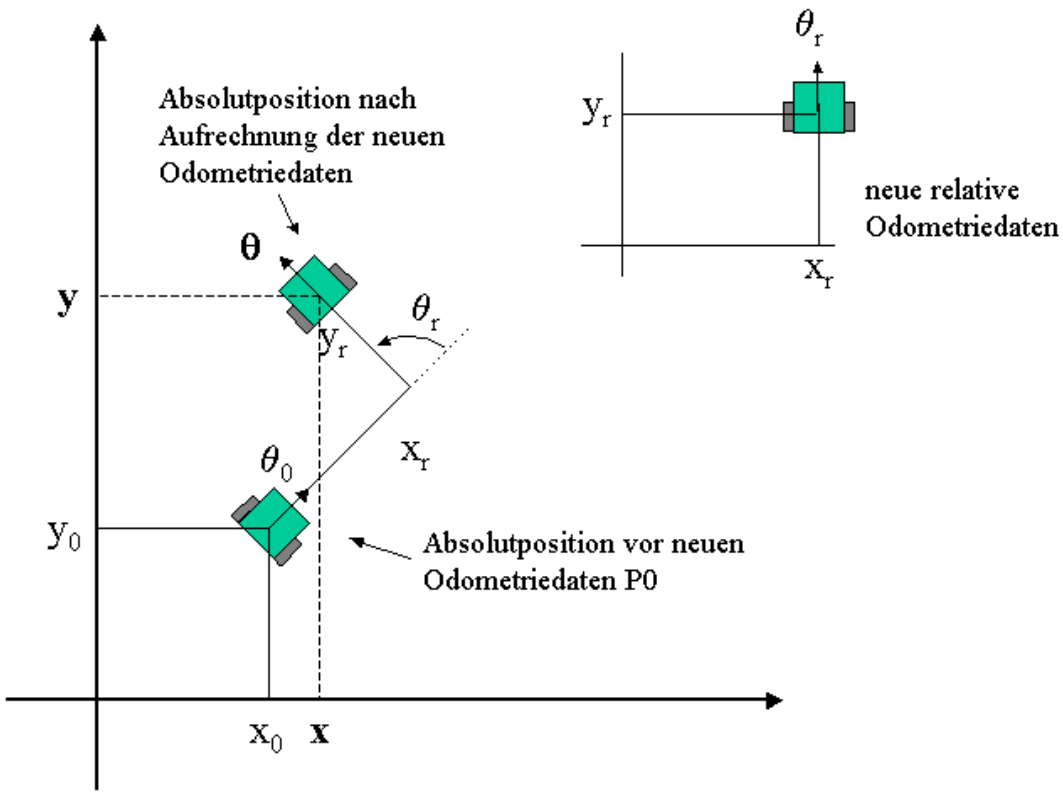


Abbildung 3.3: Aufrechnung der Odometrieinformation zur Absolutposition

x_r,y_r: Odometrieposition vom Rollstuhl

θ_r: Odometriewinkel vom Rollstuhl

Da alle Odometriewerte im Rollstuhlprogramm nach der jeweils letzten Übertragung zum Leitrechner wieder mit 0/0 und Ausrichtung 0 initialisiert werden, stellen die Odometriedaten immer die relativen Positionsveränderungen seit der letzten Übermittlung dar.

In Abbildung 3.3 ist eine beispielhafte Situation dargestellt. Im rechten oberen Bereich ist das relative Koordinatensystem dargestellt, welches vom Rollstuhl zweimal pro Sekunde an den Leitrechner übertragen wird und die relative Positionsveränderung seit der letzten Übertragung beinhaltet.

Links unten ist das Absolutkoordinatensystem mit der bisherigen Absolutposition (x_0,y_0) und Absolutausrichtung (θ_0) dargestellt. Um nun die relative Positionsveränderung auf die bisherige Rollstuhlabsolutposition aufzurechnen, wird zunächst das relative Koordinatensystem um die vorherige Absolutausrichtung des Rollstuhls rotiert.

$$x_r' = x_r \cos(-\theta_0) + y_r \sin(-\theta_0)$$
$$y_r' = -x_r \sin(-\theta_0) + y_r \cos(-\theta_0)$$

Die negativen Vorzeichen vor den Winkeln ergeben sich daraus, dass eine Drehung mit positiven Winkelwerten einer Drehung im Uhrzeigersinn entspricht. Wie bei Abbildung 3.2 dargestellt wurde, entspricht jedoch ein positives θ in der von uns gewählten internen Repräsentation der Informationen einer Drehung des Rollstuhls gegen den Uhrzeigersinn. Deshalb muss der Winkel negativ gesetzt werden.

Diese rotierten relativen Koordinaten werden auf die bisherige Absolutposition aufaddiert und ergeben damit die neue Absolutposition:

$$x = x_0 + x_r'$$
$$y = y_0 + y_r'$$

Die neue Absolutorientierung des Rollstuhls ergibt sich durch ein einfaches Aufaddieren des relativen Odometriewinkels zur bisherigen Ausrichtung:

$$\theta = \theta_0 + \theta_r$$

Beispiel:

Folgendes Beispiel, welches die Darstellung von Abbildung 3.3 wiederspiegelt, zeigt die Anwendung obiger Formeln:

Laut Leitrechner befindet sich der Rollstuhl momentan an der Position (10/10) mit einer Ausrichtung von 45°. Es wird nun vom Rollstuhl eine relative Bewegung empfangen mit den Informationen, der Rollstuhl habe sich um 15 cm nach vorne und 10 cm nach links bewegt und dabei seine Ausrichtung um 90° nach links geändert. Dies entspricht der relativen Bewegung von (0/0) Ausrichtung 0 nach (15/10) und Ausrichtung 90. Zu bestimmen ist die neue Position und Ausrichtung.

Lösung:

Eingangsdaten:

$$x_0 = 10$$
$$y_0 = 10$$
$$\theta_0 = 45°$$
$$x_r = 15$$
$$y_r = 10$$
$$\theta_r = 90°$$

Zunächst die relativen Koordinaten drehen:

$$x_r' = 15\cos(-45) + 10\sin(-45) = 3{,}535$$
$$y_r' = -15\sin(-45) + 10\cos(-45) = 17{,}677$$

Die neue Absolutposition ergibt sich durch Aufaddierung der gedrehten relativen Koordinaten auf die bisherige Absolutposition. Der neue Winkel ergibt sich ebenfalls durch Summierung des bisherigen Absolutwinkels mit der Winkeländerung:

$$x = 10 + 3{,}535 = 13{,}535$$
$$y = 10 + 17{,}677 = 27{,}677$$
$$\theta = 45 + 90 = 135°$$

3.2. Globale Lokalisierung

Durch die ständige schrittweise Berechnung der Odometrie-Werte, rutschende Räder (Schlupf) und Reibungsunterschiede wird die Abweichung zwischen der durch Odometrie bestimmten Position und der tatsächlichen Position mit zunehmender Fahrtstrecke immer größer. Die Messfehler addieren sich. Deshalb ist die Odometrie für die Positionsbestimmung auf längeren Distanzen ohne Korrekturmaßnahmen nicht geeignet.

Des Weiteren kann durch die Odometrie zwar die Bewegung nachvollzogen werden, doch ohne die Startposition zu kennen, kann auch die aktuelle Position in einer Umgebung nicht bestimmt werden. D.h. eine globale Lokalisierung ist durch die Odometrie alleine nicht möglich.

Um die beiden genannten Probleme zu lösen, bedient man sich der auf den Leitlinien angebrachten Landmarken als Referenzpositionen. Beim Passieren einer solchen Landmarke kann auf die aktuelle Position des Rollstuhls geschlossen werden. Zwischen diesen Landmarken wird mit Hilfe der Odometrie die Position verfolgt.

Wie bei der Konzepterklärung bereits kurz angesprochen wurde, werden dem System nach der Ausstattung der Umgebung mit Leitlinien und Landmarken, weder die genauen Abstände und Winkel der Linien noch die Position von Wänden und Hindernissen eingegeben. Dies würde detaillierte Pläne der Wohnung erfordern und als noch aufwändiger würde sich das Ausmessen der verlegten Linien und angebrachten Landmarken erweisen. Für eine einfache Nutzung dieses Systems wäre dies also nicht geeignet.

Aus diesem Grund kennt der Rollstuhl bei seiner ersten Inbetriebnahme in einer neuen bereits vorbereiteten Umgebung keine einzige Linie und Landmarke. Er kann daher auch beim Passieren einer Landmarke noch nicht auf eine Position ausgerichtet werden, da ja die Position der Landmarke noch nicht bekannt ist. In den folgenden Punkten 4.1. und 4.2. wird deshalb zunächst erklärt, wie der Rollstuhl seine Umgebung einlernt und korrigiert, um darin navigieren zu können. Im Punkt 4.3.

wird dann auf das Korrekturverfahren, bezogen auf Rollstuhlsposition und Ausrichtung beim Erreichen von bekannten Landmarken, detaillierter eingegangen.

4. Aufnahme der Umgebung

In diesem Kapitel wird das Verfahren zum Einlernen der Umgebung, im Speziellen der Positionen von Landmarken und Linien, erläutert. Sobald dem System diese Positionen bekannt sind, können diese Informationen zum Bestimmen der Absolutpositionen und zum Korrigieren von Odometriefehlern genutzt werden. Dies wird unter den Punkten 4.2. und 4.3. genauer erläutert.
Des Weiteren stellen die gelernten Linien und Landmarken die Grundlage für die Navigation dar.

4.1. Einlernen von Landmarken und Leitlinien - Teaching

4.1.1. Konzept des Teachings

Beim ersten Start des Systems ist dem Leitrechner die Existenz von Linien und Landmarken sowie deren Position unbekannt. Es ist daher noch nicht möglich, ein Ziel einzugeben, welches der Rollstuhl erreichen soll, da schließlich auch noch keine Ziele im System definiert wurden.

Das Konzept sieht für diesen Zweck den sogenannten Teaching-Modus vor. Es wurde versucht, auch diesen Modus so einfach wie möglich zu gestalten, damit das Erlernen der Umgebung von keinem speziell ausgebildeten Fachpersonal durchzuführen ist, sondern nach Möglichkeit dies von einem oder einer Angehörigen, Bekannten oder betreuenden Person des Rollstuhlbenutzers oder der Benutzerin durchgeführt werden kann.

Beim Teaching muss jede Landmarke und jede Leitlinie, die in der Umgebung vorhanden ist, mindestens einmal abgefahren werden. Dies geschieht durch Zusammenarbeit von Mensch und Leitrechner. Der Mensch gibt bei jeder Landmarke

die Richtung, in der die nächste Linie verfolgt werden soll, vor. Der Leitrechner verfolgt dann diese Linie bis zum Erreichen der nächsten Landmarke.

Bei jeder Landmarke wird eine Bezeichnung der Landmarke eingegeben. Mit dieser Bezeichnung wird in Zukunft die Landmarke dargestellt oder kann als Ziel ausgewählt werden.

Nachdem eine neue Linie zwischen zwei Landmarken festgestellt wurde, stellt der Leitrechner dem Benutzer oder der Benutzerin die Frage, ob diese Linie zwischen diesen Landmarken korrekt ist. Dies ist erforderlich, da es selten aber doch möglich sein kann, dass der Leitrechner im Zuge der Linienverfolgung etwas von dieser abgekommen ist und eine Landmarke ausgelassen hat, d.h. die Sensoren haben diese nicht erkannt. In diesem Fall existiert keine direkte Linie zwischen der jetzt erreichten und der zuletzt passierten Landmarke.

Das Teaching kann auch abgebrochen und zu einem späteren Zeitpunkt fortgesetzt werden.

4.1.2. Technischer Ablauf des Teachings

Nach dem Start des Systems wird im Leitrechner eine Odometrieposition mit (0/0) und Richtung 0 initialisiert. Diese Position wird im Gegensatz zur Absolutposition (die bisher noch nicht bekannt ist) niemals korrigiert. Sie spiegelt also die Bewegung wieder, die durch reine Odometriemessungen des Rollstuhls erfasst wurde und wird im Leitrechner aufgerechnet (siehe 3.1.4). Diese Positionsangabe findet, wie im Folgenden erläutert wird, für das Erlernen und Korrigieren der Umgebungskarte im Leitrechner Verwendung. Im weiteren Verlauf wird dieser Positionszähler der Einfachheit halber Odometrieposition genannt.

Sämtliche Ein- und Ausgaben werden stets über das User-Interface des Leitrechners getätigt. Wenn der Teaching-Modus gestartet wird, so wird man zunächst

aufgefordert, den Rollstuhl auf eine Linie zu bringen. Für diesen Zweck ist die manuelle Steuerung über Eingabe am Leitrechner aktiviert.
Sobald der Rollstuhl auf eine Linie gebracht wurde, kann mittels eines Knopfes auf dem User-Interface des Leitrechners die automatische Linienverfolgung gestartet werden.

Der Rollstuhl verfolgt dabei die Leitlinie bis zu einer Landmarke. Sobald er diese erreicht, stoppt er und es wird am Leitrechner die Eingabe einer Bezeichnung für diese Landmarke verlangt.

4.1.2.1. Erreichen der ersten Landmarke

Programmintern wird die erste gefundene Landmarke auf die Absolutposition 0/0 gesetzt. Somit könnte man eigentlich die Absolutposition des Rollstuhls ebenfalls auf 0/0 setzten. Da jedoch noch keine Information über die Ausrichtung des Rollstuhls verfügbar ist wird dies vorerst nicht durchgeführt.

Die momentane Odometrieposition, welche bereits seit der ersten Bewegung des Rollstuhls aktualisiert wurde, wird nun in einer Liste zusammen mit der erreichten Landmarke gesichert.

Der Rollstuhl kann im Anschluss wiederum mittels manueller Steuerung in die Richtung ausgerichtet werden, in welche die nächste Linie zur nächsten Landmarke verfolgt werden soll. Dann startet der Mensch erneut die Linienverfolgung über Knopfdruck.

4.1.2.2. Erreichen der zweiten Landmarke

Wird nun die zweite Landmarke erreicht, so können intern bereits die ersten konkreten Informationen über die Umgebung errechnet werden.
Die momentane Odometrieposition wird wiederum in der Liste der passierten Landmarken inklusive der Identifikation der Landmarke abgelegt.

Aus den abgelegten Odometriepositionen der letzten und der jetzt erreichten Landmarke kann die Entfernung beider Landmarken bestimmt werden. Da Leitlinien nur Geraden entsprechen, ist diese Entfernung zugleich die Länge der passierten Leitlinie.
Die gefahrene Distanz des Rollstuhls wird immer etwas größer sein als die tatsächliche Linienlänge, da bei der Linienverfolgung Korrekturbewegungen erfolgen. Durch die Odometrie kann jedoch die Länge von Punkt zu Punkt bestimmt werden.

Hinweis: Da Odometriemessungen fehlerbehaftet sind, werden alle Messergebnisse pro Linie in einer Statistik gespeichert und die Mittelwerte berechnet. Dadurch wird durch oftmaliges Abfahren ein besseres Ergebnis erzielt.

Die erste Leitlinie wird in der programminternen Karte immer in Richtung 0 Grad, also gerade nach rechts schauend repräsentiert. Somit kann bereits die Absolutposition der zweiten Landmarke festgelegt werden. Diese entspricht einer x Koordinate, die der Länge der Linie entspricht und y=0.

4.1.2.3. Bestimmung der Absolutausrichtung des Rollstuhls

Sobald zwei Landmarken passiert wurden, kann zusätzlich zur Absolutposition des Rollstuhls auch die Ausrichtung bestimmt werden. Dies findet nicht nur beim Teaching Verwendung, sondern das gleiche Verfahren wird auch beim Normalbetrieb zur Korrektur der Absolutausrichtung verwendet:

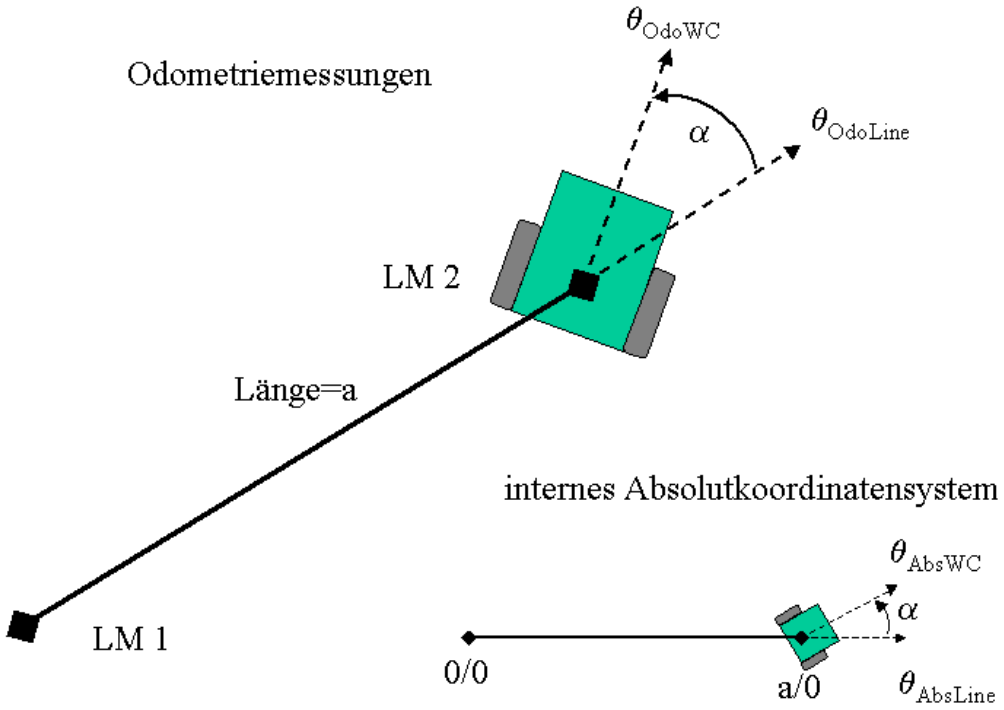


Abbildung 4.1: Bestimmung der Absolutausrichtung des Rollstuhls

Dazu wird zunächst $\theta_{OdoLine}$ der Winkel der gerade abgefahren Linie aus den Odometriepositionen der Landmarken berechnet. Dieser Winkel ist abhängig von der Ausrichtung des Rollstuhls beim Systemstart.

Zur Berechnung der Absolutrichtung des Rollstuhls wird zur Absolutausrichtung der abgefahrenen Linie die Differenz zwischen der momentanen Ausrichtung des Rollstuhls laut Odometriedaten θ_{OdoWC} und $\theta_{OdoLine}$ hinzugezählt:

$$\theta_{AbsWC} = \theta_{AbsLine} + (\theta_{OdoWC} - \theta_{OdoLine})$$

Position und Ausrichtung des Rollstuhls werden ab diesen Zeitpunkt auch am User-Interface dargestellt.

Im Anschluss wird das Teaching fortgesetzt und die nächste Leitlinie zur dritten Landmarke abgefahren.

4.1.2.4. Erreichen der dritten Landmarke

Beim Erreichen der dritten Landmarke werden wie bei allen weiteren Landmarken dieselben Schritte wie bei den Punkten 4.1.2.2. und 4.1.2.3. durchgeführt. D.h. die Länge der Leitlinien wird bestimmt und die Absolutposition des Rollstuhls wird gesetzt.

Sobald jedoch drei Landmarken hintereinander abgefahren wurden und dadurch die jeweiligen Odometriepositionen bestimmt sind, kann zusätzlich der Winkel zwischen den beiden abgefahrenen Leitlinien ermittelt werden.

Da dieser Winkel ebenso wie die Linienlängen rein aus den Odometriedaten bestimmt wurde, wird er wiederum in eine Statistik abgelegt, um jeweils den Mittelwert aus allen Messungen bilden zu können, um diesen dann für die Berechnungen heranzuziehen. Der Winkel zwischen jeweils zwei aneinander grenzenden Linien wird als Kreuzung (Crossroad CR) bezeichnet. Programmintern werden zu jeder Landmarke die dortigen Kreuzungen abgelegt.

Beispiel:
Führt von der Landmarke A jeweils eine Linie zu den Landmarken B, C und D, so werden bei der Landmarke A drei Kreuzungen abgespeichert:

CR 1: Winkel zwischen Linie A-B und Linie A-C
CR 2: Winkel zwischen Linie A-B und Linie A-D
CR 3: Winkel zwischen Linie A-C und Linie A-D

Dadurch, dass die Absolutpositionen von den ersten beiden Landmarken bereits festgesetzt wurden, kann durch die Länge der neu erkannten Leitlinie und dem Winkel zwischen dieser Linie und der zuletzt passierten, die Absolutposition der

dritten Landmarke sowie die Absolutausrichtung der zweiten Leitlinie bestimmt werden. (Siehe Abbildung 4.2)

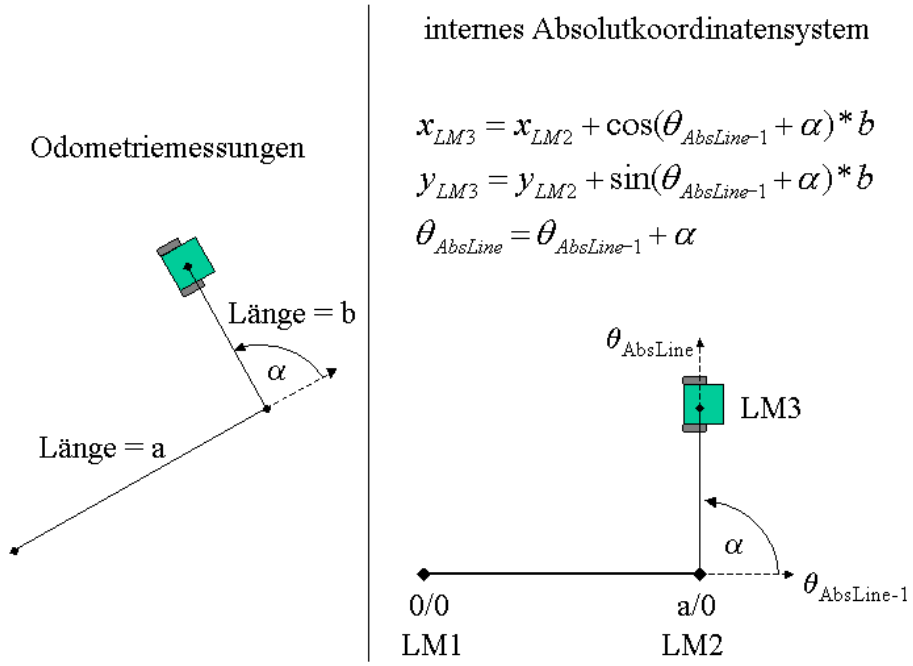


Abbildung 4.2: Bestimmung der Absolutposition der dritten Landmarke

In weiterer Folge werden die Linien zu allen anderen Landmarken abgefahren und jeweils dieser Vorgang wiederholt. Dadurch lernt das System nach und nach die Position der Landmarken und der Leitlinien ein.

Hinweis: Wenn der Rollstuhlsensor eine Landmarke erkennt, muss er dies an den Leitrechner melden. Dieser errechnet, wie eben dargestellt wurde, aus der momentanen Rollstuhlposition die Absolutposition der Landmarke. Da der Rollstuhl standardmäßig nur zweimal pro Sekunde die zurückgelegte Bewegung an den Leitrechner übermittelt, könnte es sein, dass die letzte Odometrieinformation bereits beinahe 0.5 Sekunden zurückliegt. Um zu verhindern, dass dadurch die Position der Landmarke auf eine ungenaue Position gesetzt wird, wird auch beim Erkennen einer Landmarke durch den Rollstuhl sofort die Bewegungsinformation an den Leitrechner mitgesendet.

Zusammengefasst bedeutet dies, dass die Bewegungsinformation vom Rollstuhl zweimal pro Sekunde und zusätzlich beim Erreichen einer Landmarke an den Leitrechner übertragen wird.

4.2. Korrektur der eingelernten Umgebung

Im letzten Punkt wurde erläutert, wie der Rollstuhl seine Umgebung über die Odometrieinformationen erlernt. Da diese Informationen jedoch fehlerbehaftet sein können, wäre es keine zufriedenstellende Lösung, wenn der Rollstuhl nachdem er die Landmarken und Linien einmal abgefahren hat, diese für die Zukunft immer in der ersten wahrgenommenen Art und Weise auf der Karte der Umgebung repräsentieren würde.

Die beim Teaching berechneten Absolutpunkte gelten daher immer nur für einen begrenzten Zeitraum und werden in gewissen Zeitintervallen jeweils neu berechnet. Für diese Neuberechnung ist glücklicherweise kein erneutes Teaching erforderlich. Stattdessen werden mit Hilfe der Informationen der Linienlängen und der Winkel zwischen den Linien, die Absolutpositionen und Richtungen von Landmarken bzw. Leitlinien neu berechnet. Würden sich die dafür herangezogenen Längen und Winkel im Laufe der Zeit nicht verändern, so wäre das Ergebnis freilich auch immer dasselbe.

Es werden jedoch während des Normalbetriebs, ebenso wie beim Teaching, immer diese Längen und Winkelinformationen aus den Odometriemessungen errechnet und in der Statistik abgelegt. Für eine Neuberechnung der Absolutpositionen werden dann die Mittelwerte dieser Messungen herangezogen. Einzelne Messungenauigkeiten werden so kompensiert.

Wurde eine Linie oft genug abgefahren, so wird sich der Mittelwert der Länge kaum mehr verändern.

Um zu verhindern, dass Ausreißer die Statistik und somit den Mittelwert verfälschen, sollten diese bei der Mittelwertberechnung ausgeschieden werden. Des Weiteren könnte man beim Auftreten bestimmter Kriterien das Eintragen in die Statistik verhindern. So wird die Odometrieungenauigkeit größer sein, wenn der Rollstuhl die Linie nicht ununterbrochen von einer Landmarke zur anderen verfolgt hat, sondern diese verlassen musste, um z.B. einem Hindernis auszuweichen.

Je länger der Rollstuhl in Betrieb ist, desto genauer wird das Abbild der Umgebung im Leitrechner der realen Umgebung entsprechen.

4.3. Korrektur der Rollstuhlposition und Orientierungsverlust

Dieses Kapitel knüpft an die globale Lokalisierung an. Wie dort bereits beschrieben wurde, müssen die sich aufaddierenden Odometriefehler regelmäßig korrigiert werden. Dies geschieht für die Absolutposition bei jedem Erreichen einer Landmarke, indem die Rollstuhlposition im Leitrechner auf die Position der soeben erreichten Landmarke gesetzt wird. Sobald der Rollstuhl die Landmarke wieder verlässt, wird die Position des Rollstuhls im System wieder durch die Odometrie weitergeführt, bis die nächste Landmarke erreicht wird.

Noch wichtiger als die Richtigstellung der Position ist die Korrektur der Ausrichtung des Rollstuhls. Denn wenn die Differenz der Richtung zwischen der tatsächlichen Rollstuhlrichtung und der Richtung laut Leitrechner auch nur 5° beträgt, so würde dies bei einer Strecke von 10 Metern eine Abweichung von 87 cm ergeben. Dieser Fehler wird zwar durch das hier verwendete Konzept der Linienverfolgung kompensiert, sollte aber verständlicherweise trotzdem so gut wie möglich verhindert werden.

Eine Korrektur der Rollstuhlausrichtung kann nicht bei jeder erreichten Landmarke erfolgen. Vorraussetzung ist, dass der Rollstuhl zuvor eine Landmarke passiert hat, die mit einer Leitlinie mit der aktuell erreichten Landmarke verbunden ist.

Wie bei 4.1.2.3. bereits erläutert wurde, wird zur Richtungsbestimmung der Winkel, der sich laut Odometriedaten für die passierte Linie ergibt, abzüglich der Ausrichtung des Rollstuhls laut Odometrie, zur Absolutrichtung der Linie im Leitrechner addiert. Der genaue Beobachter wird erkennen, dass also für die Korrektur der Richtung wiederum Odometriedaten verwendet werden, die wiederum fehlerbehaftet sein könnten. Dies ist zwar korrekt, jedoch kann ohne komplexere Sensorik diese Bestimmung nicht anders erfolgen. Man kann dieses Problem jedoch etwas entschärfen, indem man die Richtungskorrektur nur dann vornimmt, wenn die zurückgelegte Länge für die Linie laut Odometrie nicht wesentlich von der Länge der Linie laut Leitrechner abweicht. Des Weiteren wird durch diese Korrektur der Odometriefehler auf die Distanz zwischen zwei Landmarken beschränkt, dadurch vermeidet man die Addierung der Odometriefehler, welches bekanntlich das Hauptproblem darstellt. Abbildung 4.3 verdeutlicht abschließend wie ein sich addierender Odometriefehler aussehen würde, wenn die Landmarken nicht als Korrekturpunkte verwendet werden würden.

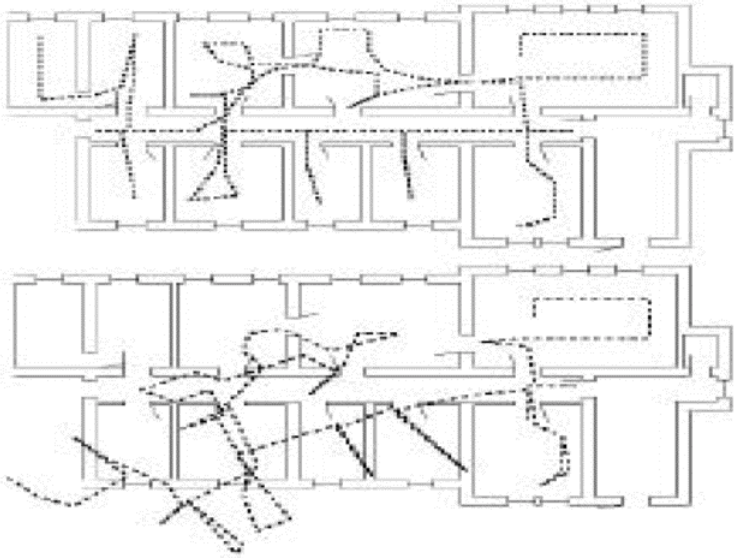


Abbildung 4.3: Addierende Odometriefehler; oben die tatsächlich gefahrene, unten die mit reiner Odometrie abgebildete Strecke

4.3.1. Orientierungsverlust („Kidnapped Robot Problem“)

Ein spezielles Lokalisierungsproblem stellt die in der Literatur als „Kidnapped Robot“ [FOX 01] bezeichnete Situation dar. Darunter versteht man, dass ein Roboter ohne sein Wissen auf eine andere Position gesetzt wird, bzw. anders ausgerichtet wird.

Ohne sein Wissen bedeutet in der konkreten Situation, dass die Sensoren zur Bewegungserfassung, also die Shaft-Encoder, die Bewegung nicht messen konnten. Dies ist zum Beispiel der Fall, wenn der Rollstuhl angehoben und versetzt wird.

Ziel ist es hierbei, dass der Leitrechner einen solchen Fehler erkennt und entsprechend korrigiert, d.h. sich neu orientiert. Um den Fehler zu erkennen, gibt es verschiedene Möglichkeiten. In der hier beschriebenen ersten Variante des Rollstuhls wurde die Fehlererkennung lediglich durch den Vergleich von erwarteter Position und tatsächlich gemessener Position von Landmarken durchgeführt. Für diesen Zweck wird eine Abweichungsdistanz festgesetzt, bei deren Überschreitung der Leitrechner in einen speziellen Modus des Orientierungsverlustes schaltet. D.h. wurde eine Landmarke vom Sensor erkannt, die laut interner Karte eigentlich mindestens der Abweichungsdistanz entsprechend weit entfernt sein müsste, so stimmt die interne Position mit der realen nicht überein.

Ein erweitertes Verfahren könnte auch die Positionen von Hindernissen, welche durch den Abstandssensor bestimmt werden, mit den erwarteten Positionen vergleichen. Dies ist nicht trivial zu lösen, da jederzeit neue Hindernisse in der Umgebung auftauchen können und dadurch der Vergleich mit erwarteten Hindernissen nicht einfach möglich ist. D.h. man müsste zwischen fixen Hindernissen wie Türstöcken bzw. Wänden und dynamischen Hindernissen unterscheiden können.

Sobald der Verlust der Orientierung erkannt wurde, wird dies am User-Interface angezeigt. Ab diesem Zeitpunkt dürfen keinerlei Statistikdaten über Linienlängen, etc. gesammelt werden, da diese Informationen fehlerhaft sein würden.

Es wird muss nun versucht werden, die Orientierung wiederzufinden.
Wie bereits erläutert wurde, genügt es jedoch in diesem Fall nicht, beim Erreichen der ersten Landmarke nur die Absolutposition des Rollstuhls im Leitrechner wieder richtig zu setzen, auch wenn die Position dieser Landmarke bekannt ist, denn es kann noch nichts über die Ausrichtung des Rollstuhls ausgesagt werden.

In die Liste der gemessenen Odometrieposition der zuletzt passierten Landmarken kann zu diesem Zeitpunkt kein Vertrauen gesetzt werden, da auch in diesen Messungen der erkannte Fehler enthalten ist, d.h. es werden auch keinerlei Informationen über passierte Linienlängen etc. aus den Odometriemessungen gewonnen. Um wieder eine korrekte Position und Ausrichtung zu erhalten, muss also zunächst wieder eine Leitlinie zwischen zwei Landmarken abgefahren werden. Dieser Ablauf wurde bereits unter dem Punkt 4.1.2.3. „Bestimmung der Absolutausrichtung des Rollstuhls“ genauer erläutert.

Ausblick: Für die Umsetzung auf einem realen Rollstuhl sollte für das Wiederfinden der Orientierung ein automatisches Verfahren konzipiert werden. Ein problematischer Aspekt hierbei ist es, dass es für den Menschen, der im Rollstuhl sitzt, nicht sonderlich angenehm sein wird, wenn der Rollstuhl, ohne seine Position in der Umgebung zu kennen nach spezifischen Algorithmen herumfährt und versucht, eine Leitlinie zu finden, um zwei beliebige Landmarken zu passieren.

5. Hinderniserkennung

In diesem Kapitel wird darauf eingegangen, wie die Hinderniserkennung umgesetzt wurde, welche Probleme dabei durch die Funkübertragung entstehen und wie diese gelöst wurden. Des Weiteren wird erläutert, wie die erkannten Hindernisse im Leitrechner repräsentiert werden.

5.1. Konzept der Hinderniserkennung

Wie bereits in Abbildung 2.4 verdeutlicht wurde, befindet sich in der hier vorliegenden ersten Ausbaustufe des semiautonomen Rollstuhls der Sensor zur Abstandsmessung in der Mitte der Vorderseite des Rollstuhls. Dieser Sensor sucht jeweils in einem Kreissegment nach dem Vorhandensein von Hindernissen und stellt das Ergebnis dem Rollstuhl zur Verfügung. Hierbei wird jeweils der Winkel relativ zur Ausrichtung des Rollstuhls und die Distanz zu einem erkannten Hindernis geliefert.

Da beim Auftreten eines Hindernisses dieses kontinuierlich im Scanbereich zu Kollisionspunkten führt, werden jeweils nur diskrete Richtungen überprüft. D.h. der Sensor erreicht zum Beispiel einen Bereich von 70° nach rechts und nach links vorne. Dieser Winkel wird in weiterer Folge Scanbereichswinkel bezeichnet. Beim einem Scandurchlauf wird nun beginnend von 70° nach links im Abstand von jeweils z.B. 5° (Scanabstand) die Distanz zu einem Hindernis bestimmt, bis 70° nach rechts erreicht wurde. Würde man diesen Scanabstand nicht festsetzten, so würde eine Wand zu unendlich vielen Kollisionspunkten führen.

Als weitere relevante Größe zum Scanbereichswinkel und zum Scanabstand ist noch die Scanreichweite zu nennen. Diese gibt an, bis zu welcher Distanz ausgehend vom Sensor ein Hindernis erkannt werden kann.

Die Größe jeder dieser Werte verändert verständlicherweise das Ergebnis. Aus diesem Grund wurden diese Parameter als Konstante ausgeführt und können im Konstantenteil der Programme verändert werden. Abbildung 5.1 zeigt den Zusammenhang zwischen den Parametereinstellungen und den gelieferten Ergebnissen beim Hindernisscan. Je kleiner der Scanabstand, desto mehr Ergebnispunkte werden geliefert.

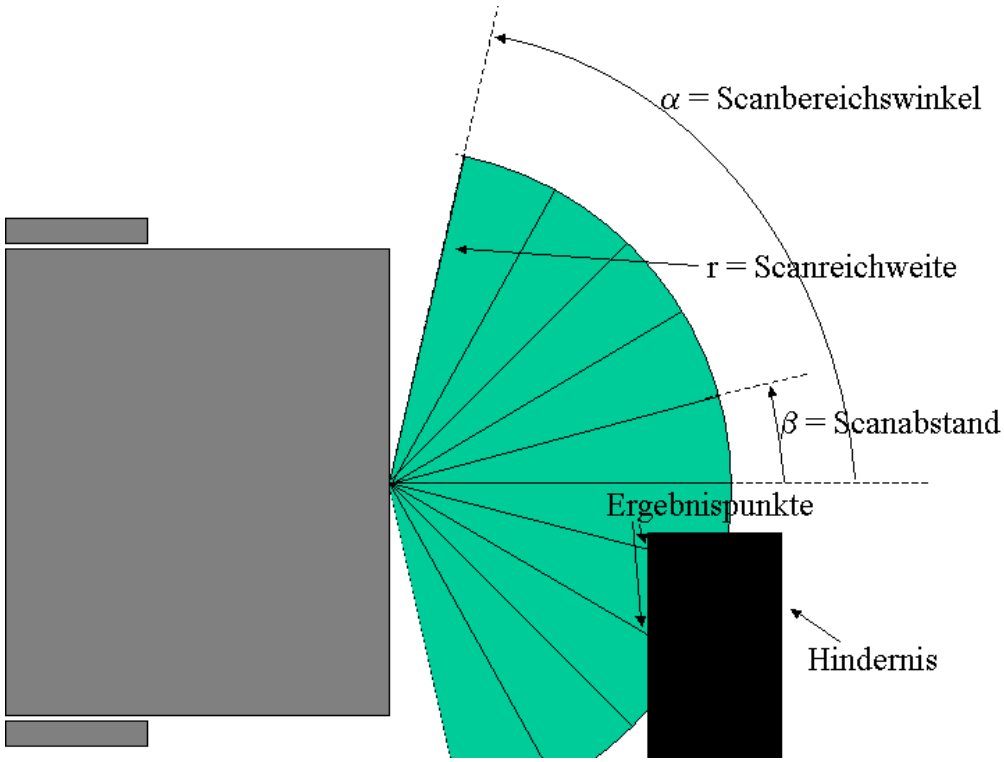


Abbildung 5.1: Relevante Parameter bei der Hinderniserkennung

5.1.1. Funkübertragung der Hindernisinformation

Die Rollstuhlsoftware erhält pro Scandurchlauf somit die Information, ob ein Hindernis im Bereich entdeckt wurde und wenn ja, an welchen Punkten. Diese Information muss nun dem Leitrechner übermittelt werden. Dem Leitrechner wird hierzu der Winkel und der Abstand zum erkannten Kollisionspunkt per Funk übertragen. Zwei Faktoren sind hierbei zu beachten. Zum Einen soll der Leitrechner möglichst schnell auf das Auftauchen von Hindernissen reagieren können, d.h. es soll in möglichst kurzen Intervallen nach Hindernissen gescannt werden. Zum

Anderen ist die Übertragungsgeschwindigkeit bei der Funkübertragung beschränkt. Der Leitrechner berechnet die Absolutposition des Hindernisses durch Aufrechnung von Winkel und Abstand zum Hindernis auf die momentane Position des Rollstuhls. Da sich der Rollstuhl seit der letzten Übertragung der Bewegungsinformation bereits weiterbewegt haben kann, muss auch beim Senden der Hindernisinformation zum Leitrechner die gemessene Odometriebewegung mitgesendet werden, damit die Absolutposition des Hindernisses nicht fehlerbehaftet berechnet wird.

Würde man nun bei jedem Scandurchlauf alle Kollisionspunkte per Funk an der Leitrechner übertragen, so würde dies zu zeitlichen Verzögerungen führen. Des Weiteren würde diese Übertragung auch die anderen erforderlichen Übertragungen von Positionsinformationen, Bewegungskommandos, etc. blockieren. Die Konzeption sieht daher vor, nur die unbedingt erforderlichen Informationen zu übertragen.

Aus diesem Grund führt die Rollstuhlsoftware selbst eine Vorverarbeitung der Hindernisinformationen durch. Es wird jeweils der nächstgelegene Kollisionspunkt bestimmt und nur dieser an den Leitrechner übertragen. Dabei ist zu beachten, dass jener Hindernispunkt, welcher den geringsten Abstand zum Sensor aufweist, nicht zwingend auch derjenige ist, welcher dem Rollstuhl am Nähesten ist.

Um diesen nächstgelegenen Punkt ausfindig zu machen, muss aus dem Sensorergebnis (Richtungen und Distanzen der Hindernispunkte) die Entfernung zum Rollstuhl berechnet werden. Da diese Berechnung bei jedem Scandurchlauf für alle erkannten Hindernispunkte wiederholt werden muss, stellen diese Berechnungen einen relevanten Zeitfaktor dar. Nachdem die Hindernisscans jedoch jeweils in spezifische Richtungen durchgeführt werden, kann man hier Berechnungszeit einsparen, indem für jede dieser Scanrichtungen jeweils im Abstand von 1cm die Entfernungen zum Rollstuhl einmalig beim Programmstart ausgerechnet und im Speicher - in einem zweidimensionalen Array - abgelegt werden.

D.h. zum Vergleich der Hindernispunkte muss nun nicht jeweils zuerst die Position des Hindernisses aus Scanrichtung und Entfernung gebildet werden, um dann den

Abstand zum Rollstuhl errechnen zu können, sondern aus der vorausberechneten Speicherstruktur kann durch Scanrichtung und abgerundeter Distanz zum Sensor direkt die Distanz zum Rollstuhl gewonnen werden.

Beispiel:
Laut Konstanteneinstellungen im Programm wird der Hindernisscan zum Beispiel in einem Scanbereichswinkel von 70 Grad nach links und nach rechts durchgeführt.
Ausgewertet wird dabei in einem Scanabstand von jeweils 10 Grad.

Dies ergibt genau 15 Scanrichtungen:-70,-60,-50,-40,-30,-20,-10,0,10,20,30,40,50,60,70 Grad
Die Scanreichweite ist auf 1,5 Meter gestellt; dies ergibt 150 Unterteilungen je einem cm.

Zum Programmstart werden nun die Entfernungen zum Rollstuhl aller möglichen Hindernispunkte vorausberechnet und in einem zweidimensionalen Array abgespeichert.

Pseudocode:

```
Schleife von –70 Grad bis +70 Grad (jeweils in 10 Grad Schritten)
    Schleife von 1 cm bis 150 cm (Abstand zum Sensor)
        Berechne Position des Hindernispunktes:
        Pos_x:=sin (AktGrad)*Aktcm
        Pos_y:=cos (AktGrad)*Aktcm
        Wenn Abs(Pos_x) < (Rollstuhlbreite/ 2) dann
            DistanzzumRollstuhl[AktGrad, Abrunden(Aktcm)]:=Pos_y
        Sonst
            DistanzzumRollstuhl[AktGrad, Abrunden(Aktcm)]:= Abstand zum
                Eckpunkt des Rollstuhls
    Schleifen-Ende
Schleifen-Ende
```

5.1.2. Kollisionsvermeidung in der Rollstuhlssoftware

Die Navigation und das Ausweichen von Hindernissen wird durch den Leitrechner durchgeführt (genaue Erläuterung folgt im Kaptitel 6. Navigation). Um in dieses System jedoch einen zusätzlichen Schutz vor Kollisionen einzubringen, wurde auch in der Software direkt am Rollstuhl, also in der untersten Ebene, ein Sicherheitsalgorithmus eingebaut [RÖF 99]. Dies ist zum Beispiel beim plötzlichen Auftauchen eines Hindernisses knapp vor dem Rollstuhl relevant. Theoretisch sollte dies nur bei bewegten Hindernissen vorkommen können. Durch die Funkübertragung der Hindernisinformation und dem Zurücksenden von entsprechenden Kommandos des Leitrechners, kann etwas Zeit verstreichen, welche ausreichen könnte, um zu einer Kollision mit diesem spontan aufgetretenen Hindernis zu führen.

Daher wird direkt am Rollstuhl nach jedem Hinderniscan eine Sicherheitsüberprüfung der durchzuführenden Bewegung durchgeführt:

Vorliegende Informationen sind dabei das aktuelle Bewegungskommando des Leitrechners, d.h. im Speziellen die durchzuführende Motoransteuerung für das linke und rechte Antriebsrad, sowie das Ergebnis des aktuellen Hindernisscans.

Die Rollstuhlsoftware betrachtet nun das nächstgelegene Hindernis. Wenn die Entfernung zu diesem Hindernis unterhalb eines Mindestabstands liegt, so wird die Zulässigkeit des Bewegungskommandos des Leitrechners überprüft. Zu diesem Zweck berechnet die Rollstuhlsoftware die Position des Rollstuhls voraus, welche dieser einnehmen würde, nachdem für einen kurzen Zeitraum die angeordnete Bewegung ausgeführt werden würde. Wenn diese neue Position dem Hindernis näher kommt, wird die Bewegung nicht durchgeführt, d.h. der Rollstuhl bleibt stehen.

Warum muss die Bewegung des Rollstuhls vorausberechnet werden, anstatt einem einfachen Stoppen, sobald ein Hindernis in eine gewisse Nähe gekommen ist?

Diese Frage lässt sich einfach am Beispiel einer manuellen Navigation am Leitrechner beantworten, d.h. der Mensch gibt die Bewegungen vor.

Angenommen ein Hindernis wurde links vorne in einem geringen Abstand entdeckt und der Rollstuhl ist automatisch stehen geblieben. Der Benutzer oder die Benutzerin gibt nun das Kommando „nach vorne fahren“. Ohne dass die Bewegung vorausberechnet wird, kann nun die Bewegung durchgeführt werden, um danach zu erkennen, dass das Hindernis näher gekommen ist. Dies würde jedoch den Rollstuhl immer näher zum Hindernis führen, bis es zu einer Kollision kommt. Alternativ könnte keinerlei Bewegung erlaubt werden, da ein Hindernis zu nahe gekommen ist. Letzteres würde dazu führen, dass sich der Rollstuhl in keine Richtung mehr steuern lässt.

Es muss also unterschieden werden, welche Bewegung durchgeführt werden darf und welche nicht. Durch das Bestimmen der erwarteten Position nach der Bewegung kann im genannten Beispiel erkannt werden, dass ein Fahren nach vorne das Hindernis näher kommen lässt. Somit wird die Bewegung nicht durchgeführt, der Rollstuhl bleibt stehen. Gibt der Mensch das Kommando „nach rechts drehen“ oder „rückwärts fahren“ so ergibt die Berechnung, dass sich das Hindernis dadurch entfernt. Die Bewegung wird somit zugelassen. Sobald die Distanz zum Hindernis wieder über der Mindestdistanz liegt, ist wieder jede Bewegung zugelassen.

5.2. Hindernisrepräsentation im Leitrechner

Es wurde bereits angesprochen, dass beim umgesetzten Konzept keinerlei Eingaben in das Programm über die Beschaffenheit der Umgebung getätigt werden. D.h. es müssen keine Informationen über den Verlauf von Wänden, die Größe von Räumen oder die Position von Hindernissen in das System eingegeben werden [SET 02]. Stattdessen nimmt das System diese Daten während der Fahrt durch die Umgebung aktiv auf und speichert diese Informationen ab.

Wie Landmarken und Leitlinien eingelernt und intern repräsentiert werden, wurde bereits ausführlich behandelt. Hier soll nun erklärt werden, wie Hindernisse repräsentiert und aktualisiert werden.

Für die rechnerinterne Darstellung der Hindernisse wurde eine rasterbasierte Darstellung gewählt. (Engl.: Grid-based Representation [MEN 95])
Hierzu wird der gesamte Umgebungsraum in Zellen unterteilt, wobei jede Zelle einen kleinen Bereich der Umgebung repräsentiert. Für jede Zelle kann festgesetzt werden, ob es sich um eine freie Zelle handelt oder ob in ihr ein Hindernis vorhanden ist.

Ein relevantes Maß hierbei stellt die gewählte Zellengröße dar. Je kleiner die Größe einer Zelle ist, desto mehr Zellen sind für die Repräsentation der Umgebung erforderlich. Dies benötigt zum Einen mehr Speicherkapazität, welches jedoch durch die räumliche Begrenzung (z.B. Wohnung) der hier vorliegenden Umsetzung kaum ein Problem darstellt. Zum Anderen erhöht sich jedoch mit der Anzahl der Zellen auch die benötigte Rechenzeit für Navigationsalgorithmen, die auf dem Hindernisraster basieren (siehe Punkt 6.1.4).

Wird die Größe einer Zelle jedoch zu groß gewählt, führt dies zu einem ungenauen Abbild der Umgebung.

Für die Umsetzung mit einem lebensgroßen Rollstuhl in einer Wohnung wurde als Größe einer Zelle jeweils 10x10cm gewählt. Bei einem Wohnbereich von 10x10m entspricht dies also einer Anzahl von 10.000 Zellen.

Für die Umsetzung auf einem Modell, welches ungefähr 1/5 der Größe eines realen Rollstuhls entspricht, wurde eine Zellengröße von 2x2cm gewählt.

In der Darstellung der Rollstuhlposition, Landmarken und Leitlinien am User-Interface des Leitrechners werden jene Zellen, die ein Hindernis enthalten, dargestellt. Bei erstmaliger Benutzung des Rollstuhles in einer neuen Umgebung sind noch keinerlei Hindernisse bekannt. Je länger sich der Rollstuhl in dieser Umgebung

bewegt, desto mehr Hindernisse werden erkannt und ergeben somit ein immer detaillierteres Abbild der Wohnung am User-Interface.

5.2.1. Hindernisse eintragen

Sobald der Leitrechner eine Hindernisinformation von der Rollstuhlsoftware erhält, muss er zunächst Berechnungen durchführen, um die entsprechende Zelle zu ermitteln, welche nun als mit einem Hindernis belegt, zu markieren ist.

Eine solche Hindernisinformation besteht aus der relativen Richtung und der Entfernung des am nächsten gelegenen Hindernispunktes vom Sensor aus gesehen. Wie bereits erklärt wurde, wird zusammen mit der Information des am nächsten gelegenen Hindernispunktes, auch die jeweilige Odometriebewegung seit der letzten Übertragung des Rollstuhls mitübertragen, damit die korrekte Absolutposition des Rollstuhls und somit auch des Hindernisses bestimmt werden kann.

Zunächst wird also die absolute Rollstuhlposition und Orientierung im Leitrechner mit Hilfe der übertragenen Odometriebewegung, wie in Punkt 3.1.4. erläutert wurde, aktualisiert.

Im Anschluss muss die Absolutposition des Entfernungssensors bestimmt werden, da dieser Punkt den Ursprung der relativen Hindernismessung darstellt.

$$x_{Sensor} = x + \cos(\theta) * d$$
$$y_{Sensor} = y + \sin(\theta) * d$$

x/y entspricht dabei der aktuellen Absolutposition des Rollstuhls.

θ entspricht der Absolutausrichtung des Rollstuhls.

d entspricht der Distanz vom Mittelpunkt der Antriebsachsen bis zur Vorderseite des Rollstuhls, an welcher der Sensor sitzt.

Ausgehend von der Absolutposition des Entfernungssensors wird nun der Absolutpunkt des gefundenen Hindernisses ebenso errechnet. Hierzu wird in Richtung des Hindernisstrahls die entsprechende Entfernung zum Hindernis zurückgelegt.

$$x_{Obstacle} = x_{Sensor} + \cos(\theta + \varpi) * a$$
$$y_{Obstacle} = y_{Sensor} + \sin(\theta + \varpi) * a$$

$x_{Obstacle}/y_{Obstacle}$
: entspricht dabei der gesuchten Absolutposition des gefundenen Hindernispunktes.

ω
: entspricht der relativen Richtung des Hindernisscans (0 Grad entspricht gerade nach vorne).

a
: entspricht der Distanz vom Sensor bis zum Hindernis.

Aus dieser Absolutposition des Hindernisses wird nun die entsprechende Zelle des Hindernisrasters mittels Division durch die entsprechende Zellengröße bestimmt. Die so gefundene Zelle wird dann als Hindernis markiert und am User-Interface entsprechend dargestellt.

Wie aus navigationstechnischer Sicht mit den erkannten Hindernissen umgegangen wird, wird im Kapitel 6. Navigation genauer erläutert.

5.2.2. Hindernisse austragen

Im vorherigen Punkt wurde darauf eingegangen, wie erkannte Hindernisse im Hindernisraster gesetzt werden. Hier wird nun erklärt, wie Zellen, welche als Hindernis beinhaltend markiert sind, wieder zu freien Zellen gesetzt werden.

Dies ist entweder der Fall, wenn sich ein Hindernis nicht mehr an derselben Position wie zuvor befindet, zum Beispiel bei einem bewegten Hindernis, oder wenn durch Messungenauigkeiten ein konstantes Hindernis bei zwei verschiedenen Messungen einmal einer bestimmten Zelle zugeordnet wurde, bei der nächsten Messung dasselbe Hindernis jedoch einer Zelle daneben zugeordnet wird. Letzteres ergibt sich daraus, dass die Absolutposition des Rollstuhls nach dem Verlassen einer Landmarke durch die Odometriemessungen bestimmt wird und diese wiederum leicht fehlerbehaftet sein können. Daher kann die errechnete Absolutposition des Rollstuhls auch beim zweimaligen Abfahren desselben Weges an ein und demselben Punkt differieren. Da Richtung und Entfernung zum Hindernis auf die Absolutposition des Rollstuhls aufgerechnet werden, schwankt somit auch die Absolutposition des Hindernisses. Je kleiner die Zellengröße gewählt wird, desto häufiger wird es zu Hindernisbelegungen kommen, die bei wiederholten Messungen von einer Zelle zur anderen springen.

Der Leitrechner bekommt vom Rollstuhl jeweils den Hindernispunkt übertragen, welcher dem Rollstuhl am nähesten ist. Die korrespondierende Zelle wird im Hindernisraster, wie unter Punkt 5.2.1 erklärt wurde, als belegt markiert. Aus der Hindernisinformation vom Rollstuhl können nun aber auch freie Zellen bestimmt werden. Dies sind folglich jene Zellen die dem Rollstuhl näher sind als die Zelle mit dem übermittelten Hindernis.

Um die freien Zellen zu finden, werden die unter Punkt 5.1.1. erläuterten vorberechneten Distanzen zum Rollstuhl verwendet. Jede Scanrichtung des Hindernissensors wird solange entlanggegangen, bis die Distanz zum Rollstuhl größer ist als die Distanz der Zelle mit dem detektierten Hindernis abzüglich des Zellenradius. Alle so gefundenen Zellen werden als frei markiert.

Um die jeweiligen Absolutpositionen zu erhalten, werden für die einzelnen Scanrichtungen die jeweiligen Einheitsvektoren berechnet. Die Absolutposition in eine Scanrichtung kann dann einfach durch Multiplikation der Distanz mit dem Einheitsvektor und anschließender Aufaddierung auf die Sensorabsolutposition gebildet werden.

Pseudocode:

Schleife AktGrad: von –70 Grad bis +70 Grad (jeweils in 10 Grad Schritten)

Schleife Aktcm: von 1 cm bis 150 cm (Abstand zum Sensor)

Wenn DistanzzumRollstuhl[AktGrad, Abrunden(Aktcm)] <

(DistanzzumRollstuhl[HindernisGrad; Abrunden (Hindernisdistanz)] – Zellenradius)

dann Zelle [Aktgrad, Aktcm] als frei markieren

Sonst

cm-Schleife beenden (denn alle weiteren Punkte sind weiter entfernt !)

Schleifen-Ende

Schleifen-Ende

Zu beachten gilt es, dass auch, wenn der Rollstuhl kein Hindernis im Scanbereich gefunden hat, diese Information an den Leitrechner übertragen wird, damit der Leitrechner alle Zellen im Scanbereich als frei markieren kann.

6. Navigation

Sobald der Mensch im normalen Betriebsmodus des semiautonomen Rollstuhls ein Ziel in Form einer dem System bekannten Landmarke eingibt, soll der Rollstuhl dieses automatisch erreichen, ohne dass der Mensch weitere Steuerbefehle erteilen muss. Der Leitrechner kennt mittlerweile die Landmarken, die Leitlinien und eventuell das eine oder andere Hindernis. Um nun automatisch in der eingelernten Umgebung zum Ziel zu finden, ist die Navigation erforderlich.

Speziell im Bereich der Navigation gibt es eine Vielzahl von Umsetzungsmöglichkeiten (vgl. [KAM 95, FOX 96,KOL 97, RÖF 97, MÜL 99]). In diesem Kapitel wird zunächst die in dieser Arbeit gewählte und umgesetzte Navigationsart erläutert. Am Ende des Kapitels wird dann noch auf eine alternative Navigationsart eingegangen, welche sich bei der Simulation als nicht passend herausgestellt hat.

6.1. Erläuterung der umgesetzten Navigationsmethode

Hauptansatz der in dieser Arbeit umgesetzten Navigationsart ist die Navigation mittels einer topologischen Karte. Hierbei werden die Informationen der Landmarken und Leitlinien verwendet, um eine Route vom Start zum Ziel zu bilden.

Eine aus der topologischen Karte generierte Route sieht inhaltlich folgendermaßen aus:

Bsp.: Folge Linie 3 bis Landmarke Küche, dann Linie 5 bis Landmarke Gang/West, dann Linie 7 bis Ziel Wohnzimmer.

Mit Hilfe eines rekursiven Algorithmus zur Bestimmung des kürzesten Weges wird diese Route erstellt. Dabei wird versucht, mittels aller bekannten Linien die kürzeste

Distanz vom Start zum Ziel zu finden. Wie unter Punkt 6.1.4. erläutert wird, können Linien durch Hindernisse unpassierbar werden; diese Linien werden, sobald sie als blockiert erkannt wurden und auch ein Umfahren des Hindernisses nicht möglich ist, für die Routenbestimmung nicht als mögliches Teilstück in Betracht gezogen. Es wird in diesem Fall eine Route ohne Verwendung der blockierten Linien gesucht, oder es existiert keine Route vom Start zum Ziel.

Diese topologische Route dient als Grundlage für die Navigation. Sie gibt die jeweiligen Zwischenziele in Form der Landmarken vor, welche durch Verfolgung der Leitlinien zu erreichen sind.

Während der Rollstuhl jeweils versucht, die nächste Landmarke laut der vorbestimmten Route zu erreichen, kommt der zweite Teil der Navigation zum Tragen. Je nach äußeren Gegebenheiten kennt die Leitrechnersoftware und somit der Rollstuhl verschiedene Verhaltensmuster in Form von Bewegungsarten, welche in unterschiedlichen Kombinationen ausgeführt werden [RÖF 97]. Das wichtigste Grundverhalten stellt dabei die Linienverfolgung dar. Wenn hierbei zum Beispiel aus irgendwelchen Gründen die Leitlinie verloren geht, d.h. der Sensor kann diese nicht mehr detektieren, so wird auf ein anderes Grundverhalten umgeschaltet, um wieder zur Leitlinie zu finden.

Das Zusammenspiel und das Umschalten zwischen den verschiedenen Grundverhaltensweisen je nach Situation ergibt das autonome Verhalten des Rollstuhls. Im Folgenden werden die wichtigsten Grundverhaltensweisen und Situationen erläutert.

Im Leitrechnerprogramm wurde eine Möglichkeit geschaffen, die Navigation und im Speziellen die jeweiligen Schritte, die durchgeführt werden, mitzuverfolgen. Wenn dieser sogenannte „Debug-Modus“ gestartet wird, werden jeweils die intern durchgeführten Navigationsschritte und Verhaltensweisen textuell angezeigt.

6.1.1. Neue Zieleingabe

Sobald der Mensch ein Ziel eingegeben hat, wird die kürzeste Route zu diesem wie unter Punkt 6.1. erläutert wurde, bestimmt. Hierbei wird auch die nächstgelegene Leitlinie und diejenige Landmarke, die durch diese Linie in Richtung Ziel erreicht wird, bestimmt. Der erste Schritt der Navigation zum Ziel besteht nun darin, diese Linie bis zur bestimmten Landmarke zu verfolgen.

Sofern sich der Rollstuhl bereits auf der Linie befindet, schaltet der Leitrechner direkt in die Verhaltensweise „Linienverfolgung“. Dies wird unter 6.1.3. erläutert. Es kann natürlich auch sein, dass der Rollstuhl zu diesem Zeitpunkt entfernt von einer Leitlinie positioniert ist. In diesem Fall wird zunächst in den Modus „Zur Leitlinie finden“ geschaltet, welcher im nächsten Punkt erläutert wird.

6.1.2. Zur Leitlinie finden

In diesen Modus wird immer dann geschaltet, wenn in die Linienverfolgung geschaltet werden soll, der Rollstuhl jedoch nicht auf einer Linie platziert ist. Dies kann z.B. der Fall sein, wenn ein neues Ziel angefahren werden soll, oder wenn nach dem Ausweichen von einem Hindernis wieder zur Linie zurückgefunden werden soll, um diese weiter zu verfolgen oder aber auch direkt im Linienverfolgungsmodus, wenn dabei die Linie verloren gehen sollte.

Wie zur Leitlinie gefahren wird, ist wiederum von den jeweiligen momentanen äußeren Gegebenheiten abhängig. Wenn der Abstand von Rollstuhl zur Leitlinie unterhalb eines als Konstante einstellbaren Grenzwertes liegt, wird die Linie schräg angefahren, um beim Erreichen der Linie sofort in den Linienverfolgungs-Modus schalten zu können, ohne eine weitere gravierende Richtungskorrektur durchführen zu müssen. Diese Situation ist in Abbildung 6.1a dargestellt. Ist der Rollstuhl jedoch weiter von der Leitlinie entfernt, wird zunächst direkt in Richtung der Linie gefahren und erst wenn sich der Rollstuhl in der Nähe der Linie befindet, wird diese wieder schräg angefahren. Vergleiche dazu Abbildung 6.1b.

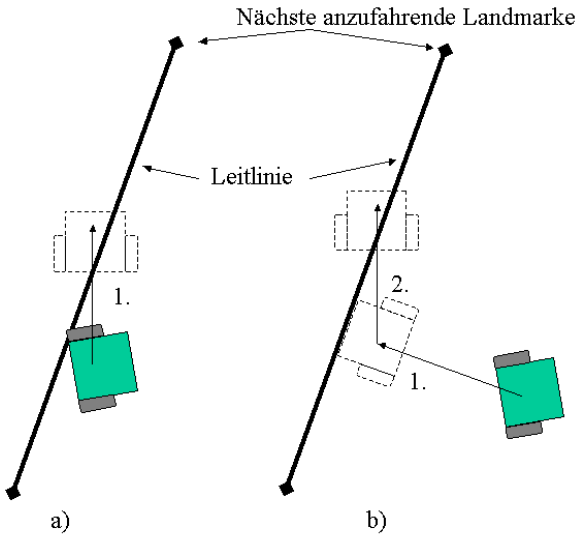


Abbildung 6.1: Zur Leitlinie finden

6.1.3. Linienverfolgung

Das Navigationskonzept sieht vor, dass der Rollstuhl immer versucht, Leitlinien zu erreichen, um diese in Richtung Ziel zu verfolgen. Selbst wenn er die Linie verlassen musste, um einem Hindernis auszuweichen, wird er, sobald dies abgeschlossen wurde, wieder versuchen, zur Linie zu finden, um wieder in das Grundverhalten der Verfolgung zurückzukehren. Wie bereits erläutert wurde, sind für die Linienverfolgung zwei analoge Sensoren in der Mitte der Antriebsachse vorgesehen.

Die Linienverfolgung ist mit Hilfe eines Regelkreises implementiert, welcher mit Hilfe der Sensorwerte der beiden Liniendetektoren und der Veränderung dieser Werte über die Zeit, den Antrieb des linken und rechten Rades regelt. Es werden also in kurzen Zeitabständen die Sensorwerte gelesen und jeweils die Veränderung zur letzten Messung bestimmt.

Diese Veränderung der Sensorwerte über ein Zeitintervall ist ein Maß dafür, in welchem Winkel sich der Rollstuhl zur Leitlinie bewegt. Haben beide Liniensensoren eine Linie detektiert und die Veränderung der Sensorwerte ist nahe

Null, so bedeutet dies zum Beispiel, dass der Rollstuhl die Linie genau im richtigen Winkel befährt. Diese Situation wird in Abbildung 6.2a dargestellt.

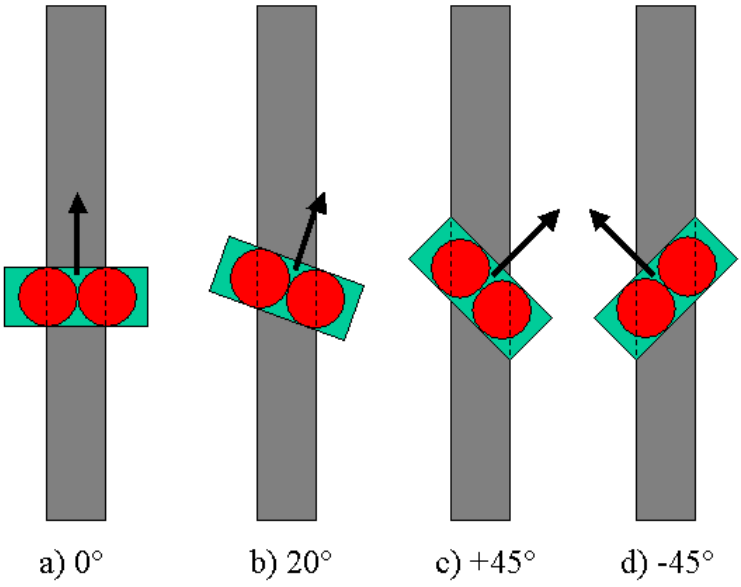


Abbildung 6.2: Abweichungen der Richtungen von Rollstuhl und Leitlinie

Abbildung 6.2b verdeutlicht die Situation, wenn der Rollstuhl eine 20° Abweichung zur Leitlinie aufweist. In diesem Fall ist eine Veränderung der Messwerte der Liniensensoren pro Messung zu verzeichnen. Es wird also eine Korrekturbewegung in der Form stattfinden, dass das rechte Antriebsrad schneller als das linke betrieben wird und somit eine Korrekturbewegung nach links erfolgt. In Abbildung 6.2c wird eine 45° Abweichung dargestellt. Hier ist die Veränderung der Sensormesswerte bereits stärker als in 6.2b, da der Rollstuhl schneller die Linie unter den Sensoren verliert. Dementsprechend wird auch die Korrekturbewegung stärker ausfallen.

Abbildungen 6.2c und 6.2d zeigt auf, warum zwei analoge Sensoren erforderlich sind und nicht einer ausreicht. Die Veränderung der Sensormessungen könnte selbstverständlich auch mit nur einem Sensor bestimmt werden. Es könnte jedoch

mit einem Sensor alleine nicht unterschieden werden, ob der Rollstuhl die Linie nach links (6.2c) oder nach rechts (6.2d) verliert. Dies ist aber relevant, um in die richtige Richtung korrigieren zu können. Mit zwei Sensoren können diese Situationen unterschieden werden, indem die beiden Sensorwerte und deren Abweichungen zur letzten Messung verglichen werden. In 6.2c würde der rechte Sensor bei der nächsten Messung weniger Linie erkennen der linke hingegen würde mehr Linie detektieren. In 6.2d ist genau das Umgekehrte der Fall.

Um Pendelbewegungen bei der Linienverfolgung möglichst gering zu halten, sind bei der Regelung feine Abstimmungen erforderlich, wie stark die einzelnen Korrekturbewegungen durchgeführt werden.

Um ein gutes Ergebnis bei der Linienverfolgung erzielen zu können, ist es erforderlich, dass der Regelkreis in kurzen Zeitabständen durchlaufen wird. In der Simulation wurde der Durchlauf alle 20ms durchgeführt, d.h. 50 mal pro Sekunde; dies führt zu guten Resultaten.

Würde man den Linienverfolgungsalgorithmus vom Leitrechner durchführen lassen wollen, so müsste man bei jedem Durchlauf beide Sensorwerte zum Leitrechner per Funk übertragen und das berechnete Ergebnis der Regelung wieder retour senden. Dies ist verständlicherweise nicht durchführbar, da dadurch keinesfalls alle 20ms eine Regelung durchgeführt werden könnte. Des Weiteren wäre dann die Zeitspanne zwischen der Messung und dem Setzten der durch die Regelung bestimmten Motorwerte für das linke und rechte Antriebsrad durch die Funkübertragung zu groß und es würde zu einem ständigen Pendeln des Rollstuhls auf der Linie kommen, bzw. würde er die Linie vermutlich oft gänzlich verlieren.

Aus diesem Grund wird die Linienverfolgung direkt in der Software am Rollstuhl durchgeführt. Der Leitrechner sendet nur das entsprechende Kommando und der Rollstuhl übernimmt dann die Regelung der Linienverfolgung. Dies stellt gleichzeitig die komplexeste Aufgabe dar, welche von der Rollstuhlsoftware selbst durchführt wird. Jegliche sonstige Navigation wird im Leitrechner durchgeführt und

dem Rollstuhl werden nur die entsprechenden Motoransteuerungen als Kommandos übermittelt. Eine Übersicht aller Kommandos ist unter Punkt 6.1.6. zu finden.

Die Wiederholungsrate der Linienverfolgungsregelung kann im Simulationsprogramm verändert werden, um so Vergleiche durchführen zu können, da speziell bei der Umsetzung auf einem Modell die Wiederholungsrate eventuell nicht so hoch angesetzt werden kann. (Abhängig von der verwendeten Hardware).

Es kann während der Linienverfolgung vorkommen, dass der Rollstuhl die Leitlinie verliert, d.h. beide Sensoren befinden sich neben dieser. Wenn dieser Zustand eintritt, wird dies dem Leitrechner mitgeteilt, welcher dann in den Modus „Zur Leitlinie finden“ (Punkt 6.1.2.) umschaltet, um wieder die Linie zu finden und deren Verfolgung dann wieder durch den Rollstuhl fortsetzen lassen zu können.

6.1.4. Hindernisbehandlung bei der Navigation

Der Rollstuhl wird solange wie möglich die Leitlinien in Richtung Ziel verfolgen. Es kann nun jedoch vorkommen, dass eine Linie durch ein Hindernis blockiert ist, bzw. der Rollstuhl die Linie verlassen muss, um an einem Hindernis vorbeinavigieren zu können.

Wird ein solches Hindernis während der Navigation erkannt, so versucht der Leitrechner zunächst in Richtung Ziel, an diesem Hindernis durch ein entsprechendes Ausweichmanöver vorbeizufahren. Dazu wird der momentane Modus unterbrochen, z.B. die Linienverfolgung und in den Ausweichmodus geschaltet. Der Rollstuhl verlässt dazu dann die Leitlinie. Das Vorbeifahren wird dabei solange versucht, bis der Leitrechner erkennt, dass ein Vorbeikommen an dem Hindernis nicht möglich ist. Sobald dies erkannt wurde, wird diese Leitlinie als blockiert markiert und es wird eine neue Route zum Ziel gesucht, welche nicht über diese momentan blockierte Linie führt.

Wenn der Leitrechner ein Hindernis passiert hat, so wird danach wieder zur Leitlinie zurückgefahren, um wieder über die Linienverfolgung weiter navigieren zu können. Dabei ist Folgendes zu beachten:

Da es sein kann, dass nach dem Verlassen der Linie und dem Ausweichen eines Hindernisses, die dann dem Rollstuhl am nächsten gelegene Linie nicht mehr dieselbe ist, welche zuvor entlanggefahren wurde, wird die nächstgelegene Linie neu bestimmt und gegebenenfalls auch eine neue Route berechnet.

Beispiel eines möglichen Ausweichverhaltens, dargestellt in Abbildung 6.3:

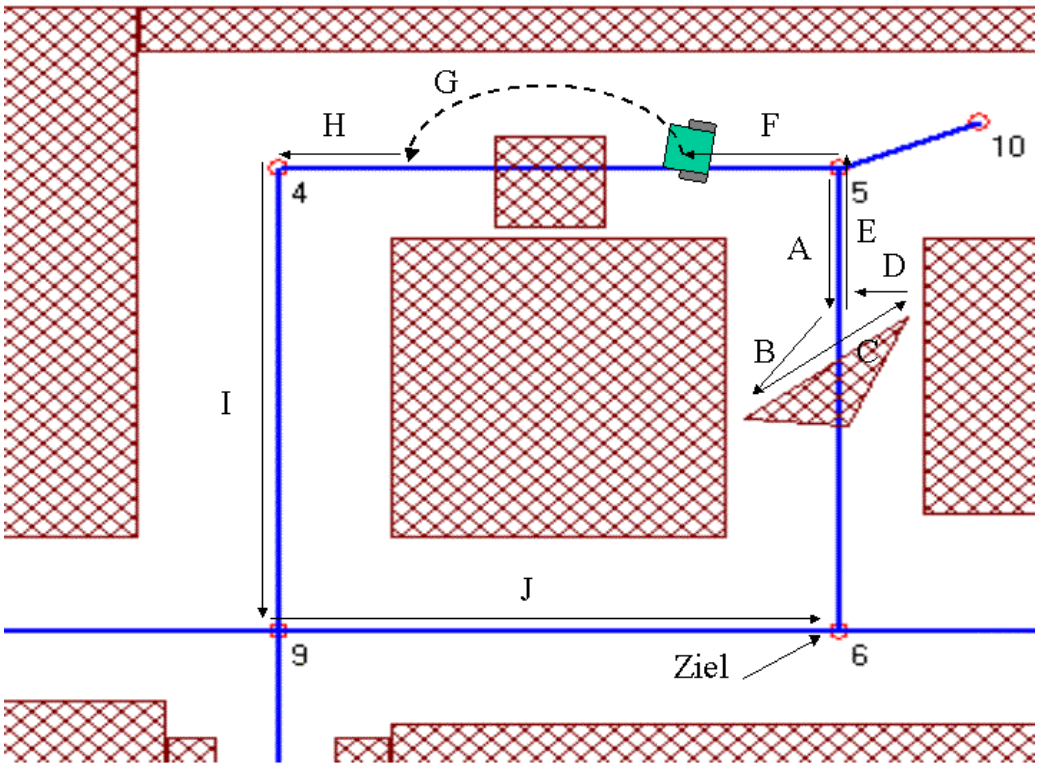


Abbildung 6.3: Beispiel: Ausweichen von Hindernissen

Der Rollstuhl befindet sich zu Beginn auf Landmarke Nr. 5. Der Benutzer oder die Benutzerin gibt nun das Ziel in der Form von Landmarke Nr. 6 ein.

Es wird zunächst vom Leitrechner die kürzeste Route von LM 5 nach LM 6 bestimmt, welche als Ergebnis die direkte Linie von 5 nach 6 liefert.

Der Rollstuhl fährt nun die Strecke A von LM 5 in Richtung LM 6 ab, bis das Hindernis erkannt wird. Nun versucht der Rollstuhl auf Strecke B an dem Hindernis

vorbeizufahren, erkennt jedoch, dass dies nicht möglich ist und versucht daher im Anschluss, an der anderen Seite am Hindernis vorbeizukommen (Strecke C). Nachdem auch dies gescheitert ist, wird im Leitrechner die Linie von LM 5 nach LM 6 als blockiert markiert und es wird eine alternative Route bestimmt.

Hier wird als Ergebnis folgende alternative Route berechnet.
LM 5 – LM 4 – LM 9 – LM 6

Es wird nun die Strecke D abgefahren, um wieder zur Leitlinie zurückzukehren. Dann wird diese Linie verfolgt bis Landmarke 5 (Strecke E). Als nächstes wird von Landmarke 5 Richtung 4 die Leitlinie entlang der Strecke F verfolgt, bis hier wiederum ein Hindernis erkannt wird. Wiederum versucht der Rollstuhl an dem Hindernis vorbeizufahren, was diesmal über die Strecke G auch gelingt.
Nach dem Ausweichmanöver wird wieder auf Linienverfolgung umgeschaltet und es werden die Strecken H, I und J bis zum Ziel abgefahren.

Bis das dargestellte Ausweichverhalten des Rollstuhls erreicht wurde, wurden verschiedene Ansätze versucht. Der erste trivialste Ansatz war es, den Rollstuhl, abhängig von den jeweils aktuellen Hindernissensorwerten, ein entsprechendes Manöver durchführen zu lassen. Bei neuen Hindernissen hat dies auch zu respektablen Ergebnissen geführt. Wenn zu einem späteren Zeitpunkt jedoch dasselbe Hindernis erneut aufgetreten ist, hat der Rollstuhl wiederum versucht, je nach aktuellen Sensorwerten um das Hindernis herum zu kommen, obwohl mittlerweile bereits mehr über den Verlauf des Hindernisses bekannt war und somit im Vorfeld eine ideale Ausweichroute bestimmbar gewesen wäre.

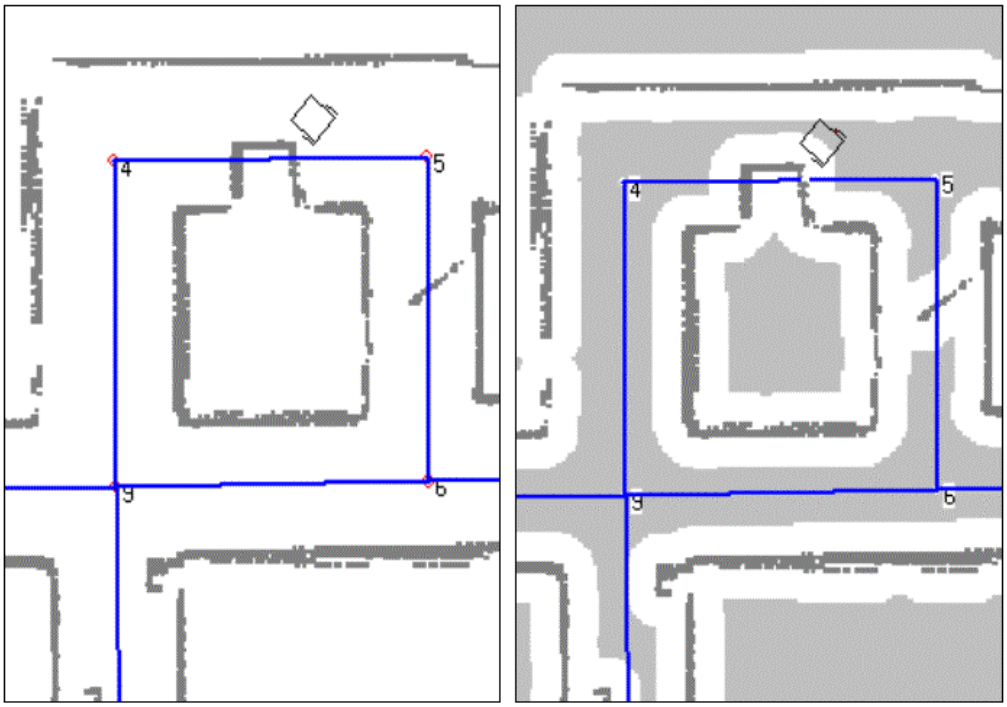
Es wurde daher nach einer Navigationsmethode gesucht, die bei einer Blockierung der Route durch ein Hindernis, die bereits bekannten Informationen der Hindernispositionen, d.h. die Werte des Hindernisrasters, auswertet, um ein besseres Ausweichverhalten zu erzielen.

Hierfür wurde zunächst der Voronoi-Algorithmus in Betracht gezogen, welcher sich jedoch für die vorliegenden Anforderungen als nicht passend herausgestellt hat. Der Grund dafür und die Funktionsweise dieses Algorithmus wird unter Punkt 6.3. eingehend erläutert.

Schließlich wurde eine Methode gefunden, welche zu guten Ergebnis geführt hat. Ziel der Methode ist es, Wegpunkte zu finden, die den Rollstuhl um das Hindernis herum zur nächsten Ziellandmarke führen. Hierfür können natürlich immer nur die momentan bekannten Informationen des Hindernisrasters ausgewertet werden. Wenn ein Weg gefunden wird, so wird in Richtung der ersten Wegmarke gesteuert und versucht, das Hindernis zu umfahren. Sollte dabei auf neue Hindernisse gestoßen werden, wird mit Hilfe des aktualisierten Hindernisrasters versucht, eine neue Ausweichroute zu finden. Sobald erkannt wird, dass keine Route zum Ziel führt, d.h. der Weg ist gänzlich blockiert, wird ein Weg über andere Landmarken gesucht.

Diese Methode basiert auf dem Hindernisraster im Leitrechner. Für jede Zelle wird hierbei bestimmt, ob diese einen gewissen Mindestabstand zum nächstgelegenen Hindernis aufweist. Diese Zellen werden in weiterer Folge als die befahrbaren Zellen bezeichnet und speziell gekennzeichnet.

Abbildung 6.4a zeigt ein Beispiel für die Repräsentation einer Umgebung im Leitrechner mit Leitlinien, Hindernissen und Landmarken. In Abbildung 6.4b ist der gleiche Bildschirmausschnitt zu sehen. Hier sind alle befahrbaren Bereiche hellgrau dargestellt. Der Mindestabstand zum Hindernis, den ein Punkt aufweisen muss, um zu einem befahrbaren Punkt erklärt zu werden, kann als Konstante im Programm gesetzt werden.



a) Umgebung im Leitrechner b) Befahrbare Bereiche (hellgrau)

Abbildung 6.4: Hindernisse und befahrbare Bereiche

Im nächsten Schritt wird versucht, einen Weg am äußeren Rand entlang der befahrbaren Bereiche zur Ziellandmarke zu finden (vgl. [KAM 95]). Hierfür wird von der aktuellen Rollstuhlposition in beide Richtungen ein Weg gesucht. Auf diesem Weg werden dann Navigationspunkte bestimmt, welche als Subziele dienen. Immer dann, wenn der Weg eine relevante Richtungsänderung vollzieht, muss ein Navigationspunkt definiert werden [MUS 00]. Abbildung 6.5 stellt die berechneten befahrbaren Wege von der momentanen Rollstuhlposition ausgehend dar. Da die Ziellandmarke die Landmarke Nr. 4 ist, wird der nach rechts führende Weg ausgeschieden, da dieser nicht zu dieser Landmarke führt. Würden beide Richtungen zur Landmarke führen, so wird derjenige Weg ausgewählt, der die kürzere Weglänge aufweist.

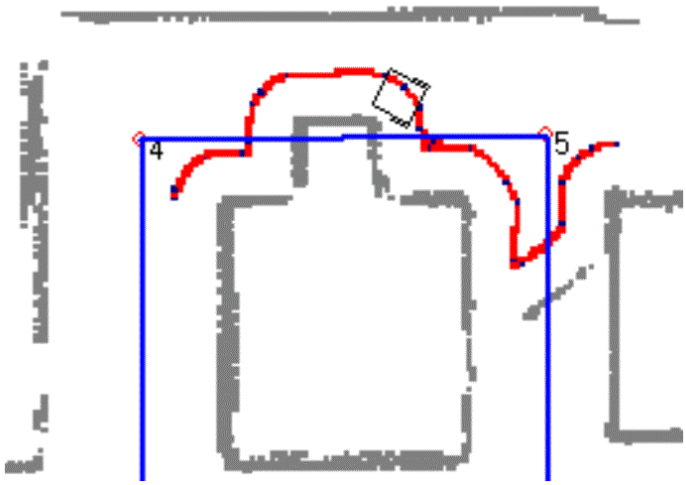


Abbildung 6.5: Berechnete Ausweichroute

Zuletzt wird untersucht, ob zwischen zwei Navigationspunkten eventuell eine direkte Verbindung über einen befahrbaren Bereich existiert, um zu verhindern, dass der Rollstuhl dem Rand entlang der befahrbaren Bereiche einen Umweg fährt. Des Weiteren wird jene Landmarke gesucht, von der direkt zum Ziel gefahren werden kann. Zu erwähnen bleibt an dieser Stelle, dass während des Ausweichmanövers ständig die aktuellen Hinderniswerte ausgewertet werden, um bei einer Veränderung der tatsächlichen Hindernisposition, zu der im Leitrechner gespeicherten bisherigen Positionen entsprechend, korrekt zu navigieren.

6.1.5. Weitere Navigationsverhalten

Die bisher genannten Navigationselemente stellen die Wesentlichsten dar. Es gibt jedoch noch verschiedene zusätzliche Verhaltesweisen, welche zu einem besseren Verhalten des semiautonomen Rollstuhls führen. Sollte zum Beispiel der Rollstuhl bei der Navigation in eine Ecke geraten sein, so muss in diesem Fall zuvor ein Stück zurückgefahren werden, um anschließend eine Drehung durchführen zu können, da es sonst bei der Drehung zu einer Kollision kommen würde.

Des Weiteren gibt es Situationen, bei denen keine Linie zum Ziel verfolgt werden kann, sondern die Landmarke direkt angefahren wird. Dies wäre zum Beispiel der

Fall, wenn sich in Abbildung 6.3 der Rollstuhl rechts neben Landmarke 10 befinden würde und der Mensch die Landmarke 10 als Ziel eingibt.

6.1.6. Übersicht über die Navigationskommandos an den Rollstuhl

Die Befehle, welcher der Leitrechner im Zuge der Navigation per Funk an das Rollstuhlprogramm übermittelt, werden hauptsächlich durch Kommandos zur direkten Motoransteuerung dargestellt. D.h. wenn der Leitrechner eine Drehung um 45° anordnen möchte, gibt er die entsprechenden Antriebsgeschwindigkeiten der Räder vor. Diese könnten zum Beispiel für das linke Rad 0 und das rechte Rad 50 sein, um die Drehung nach links auszuführen. Generell wurde in der Umsetzung davon ausgegangen, dass die beiden Antriebsräder im Bereich von –100 bis +100 angesteuert werden können. Wobei –100 einem Rückwärtsfahren mit maximaler Geschwindigkeit, +100 einem Fahren nach vorne mit maximaler Geschwindigkeit und 0 einem Stillstand entspricht. Die maximale Geschwindigkeit kann wiederum als Konstante im Programm gesetzt werden.

Da es durch die Funkübertragungen zu Zeitverzögerungen kommt und speziell bei Richtungsänderungen größtmögliche Genauigkeit erwünscht ist, wird bei den Bewegungskommandos zusätzlich zu den Motoransteuerungen auch noch ein Zielwert angegeben. D.h. bei einer Drehung um 45° wird dem Rollstuhl auch der erwünschte relative Winkel der Drehung mitübertragen. Sobald der Rollstuhl diesen erreicht hat, stoppt er und teilt dies dem Leitrechner mit. Würde man dem Rollstuhl nur die Motorbefehle übermitteln, so müsste der Leitrechner die Bewegungsinformationen des Rollstuhls auswerten und nach 45° ein Stopp senden. Bis dieses vom Rollstuhl ausgeführt werden würde, hätte er sich jedoch durch die Zeitspanne, welche durch die Funkübertragung verstrichen ist, bereits weiter gedreht, was das Navigieren komplizieren würde.

Zusätzlich überwacht auch der Leitrechner jede Bewegung und kann jederzeit neue Befehle an den Rollstuhl übertragen, bzw. muss er, wie im nächsten Punkt erläutert

wird, zu gewissen Zeitintervallen das aktuelle Kommando bestätigen, damit der Rollstuhl dieses weiter ausführt.

Die Linienverfolgung wird wie bereits unter Punkt 6.1.3. erklärt wurde, direkt von der Rollstuhlsoftware durchgeführt. Möchte der Leitrechner, dass der Rollstuhl eine Linie verfolgt, so sendet er das Navigationskommando „Linienverfolgung“ an diesen.

Ein eigenes Kommando wurde für einen Stopp-Befehl eingeführt, wobei dieses eigentlich gleichbedeutend ist mit einem Bewegungsbefehl, welcher den linken und rechten Motor auf 0 setzt. Nachdem Stopp jedoch nur einen einzigen Signalwert erfordert, kann so dieses wichtige Kommando schneller über Funk übertragen werden.

Ein spezielles Kommando stellt noch die Kombination eines Bewegungskommandos mit der Anweisung in den Linienverfolgungsmodus zu schalten, sobald eine Linie detektiert wird, dar. Dieser Befehl hat sich als notwendig erwiesen. Wenn sich der Rollstuhl entfernt von einer Linie befindet, so muss vor der Linienverfolgung zunächst per Bewegungskommando zu dieser gefahren werden. Würde der Rollstuhl beim Erreichen der Linie dies dem Leitrechner zwar mitteilen, jedoch dann weiterfahren, bis er vom Leitrechner den Befehl zur Linienverfolgung erhält, so wäre die Linie dann oftmals wegen der Zeitverzögerung bereits wieder hinter den Sensoren verschwunden und somit wiederum keine Linienverfolgung möglich.
Um dieses Problem auszuschalten, gibt es die erwähnte Kombination von Kommandos. Der Rollstuhl führt das entsprechende Bewegungskommando aus. Sobald er eine Linie detektiert, schaltet er selbstständig in den Linienverfolgungsmodus und teilt dem Leitrechner dies mit. Somit wird die Zeitspanne der Funkübertragung ausgeschaltet.

Zusammengefasst werden also folgende Kommandotypen vom Leitrechner übertragen:

- Bewegungsbefehl: Motor-Links; Motor-Rechts; Max. Winkel oder Max. Distanz
- Linienverfolgung
- Stopp
- Bewegungsbefehl mit automatischem Linienverfolgskommando beim Erreichen einer Linie

6.2. Sicherheitsaspekte bei der Navigation

Jede automatisierte Steuerung sollte aus Sicherheitsgründen ein sogenanntes Fail-Safe Verhalten aufweisen. Wie eingangs bereits kurz erläutert wurde, wird darunter verstanden, dass bei Ausfall der Steuerungsebene oder wenn diese fehlerhafte Kommandos liefert, der gesteuerte Bereich in einen sicheren Zustand geführt wird.

Zur Erläuterung von Fail-Safe Verhalten wird ein Beispiel aus [SCH 98] angeführt, welches das Fail-Safe Verhalten von früheren Signalsteuerungen auf Eisenbahnstrecken erklärt:
Das Signal „freie Fahrt“ wurde dadurch gesetzt, indem über einen Seilzug der Signalarm nach oben gezogen und dadurch in einen Zustand höherer potentieller Energie gebracht wurde. D.h. der Energieaufwand war für den gefährlichen Zustand „freie Fahrt“ erforderlich. Würde diese Energie im Fehlerfall aussetzen oder das Seil reißen, so würde das Signal aufgrund seines Gewichtes in den energieärmeren Lagezustand zurückfallen und somit automatisch der sichere Zustand „Halt“ erreicht werden.

Im Teil der Rollstuhlsoftware wurde dieses Fail-Safe Verhalten so umgesetzt, dass der Rollstuhl eine Bewegung in Form eines Kommandos vom Leitrechner nur dann weiter ausführt, wenn er in gewissen Zeitintervallen eine erneute Bestätigung für den Befehl erhält. Dies gewährleistet, dass der Rollstuhl bei Ausfall des Leitrechners

oder der Funkverbindung den letzten Befehl nicht weiterhin ausführt, obwohl der Leitrechner keine Möglichkeit zur Überwachung bzw. zum Intervenieren mehr hat.

Etwas, das bei der Simulation mangels realer Hardwareteile nicht umgesetzt wurde, bei der Anwendung auf einem Modell oder primär auf einem realen Rollstuhl jedoch Berücksichtigung finden sollte, ist das Fail-Safe Verhalten des Antriebs am Rollstuhl selbst. Auch wenn das Programm, das am Rollstuhl läuft, Fail-Safe ist, so könnte immer noch dieses Programm selbst ausfallen und fehlerhaft sein. Da dieses Programm jedoch die Ansteuerung der beiden Antriebsräder vornimmt, sollte auch direkt in der Hardware des Rollstuhls gewährleistet werden, dass bei einem Ausfall des Rollstuhlprogramms der Rollstuhl stehen bleibt.

Einen anderen Sicherheitsaspekt stellt das automatische Stoppen des Rollstuhls durch die Rollstuhlssoftware ab dem Unterschreiten einer Mindestdistanz zu einem Hindernis dar. Dieser Aspekt wurde bereits unter dem Punkt 5.1.2. „Kollisionsvermeidung in der Rollstuhlssoftware“ ausführlich erläutert.

Wenn der Rollstuhl bei dem Kommando „Linienverfolgung“ die Leitlinie verliert, so sendet er dies, wie ebenfalls bereits erwähnt wurde, an den Leitrechner, damit dieser entsprechende Kommandos sendet, um wieder zur Leitlinie zurückzukehren. Sollte der Leitrechner in einem gewissen Zeitintervall nicht antworten, wird ebenfalls automatisch gestoppt. Es wird hierbei nicht sofort nach Verlust der Linie gestoppt, um nach Möglichkeit das Bremsen und wieder Beschleunigen zu vermeiden, da zumeist nur die Richtung etwas korrigiert werden muss.

Zusätzlich muss selbstverständlich auch der Mensch jederzeit eine automatische Navigation stoppen können. In der hier vorliegenden Simulation ist dies durch einen eigenen Stopp-Button am User-Interface möglich bzw. jeder manuelle Eingriff in die Steuerung durch eine manuelle Lenkbewegung unterbricht die automatische Navigation.

6.3. Alternative Navigationsmöglichkeit

Ein spezielles Merkmal der in dieser Arbeit letztendlich umgesetzten Navigationsmethode ist, dass beim Vorausberechnen der Route hauptsächlich die topologische Karte der Leitlinien und Landmarken verwendet wird und zusätzlich als blockiert erkannte Linien berücksichtigt werden. Die rasterbasierte Karte der Hindernisse wird hierbei bei der Navigation erst dann verwendet, wenn ein Hindernis die Linienverfolgung blockieren sollte.

Ein Grund für die Wahl der oben beschriebenen Navigationsmethode anstatt einer Methode, welche die bisher erkannten Hindernisse von Anfang an in die Kursberechnung einschließt, ist es, dass das Konzept darin besteht, Leitlinien als Navigationsgrundlage heranzuziehen und nicht völlig frei auf beliebigen hindernisfreien Gebieten zum Ziel zu fahren. Des Weiteren wird davon ausgegangen, dass die Leitlinien grundsätzlich befahrbar sind. Selbst wenn wie in Abbildung 6.3 ein Hindernis eine Linie völlig blockiert, so soll zu einem späteren Zeitpunkt trotzdem wieder versucht werden, diese Linie entlang zu fahren, da ansonsten niemals erkannt werden könnte, dass das Hindernis eventuell nicht mehr vorhanden ist und somit immer der Umweg gefahren werden würde.

Im Folgenden wird die Navigation nach dem Voronoi-Algorithmus beschrieben, welche sich als nicht zielführend für das hier vorliegende Konzept herausgestellt hat. Diese Methode bestimmt den Kurs aufgrund von Hindernisinformationen und benötigt hierfür möglichst umfangreiche und detaillierte Messungen von eben genau diesen Hindernissen. Bei dem vorliegendem Konzept erhält der Leitrechner aufgrund der eingeschränkten Übertragungsgeschwindigkeit der Funkübertragung jedoch immer lediglich die Information des nächstgelegenen Hindernispunktes.

6.3.1. Navigation nach dem Voronoi-Algorithmus

Bei diesem Navigationsverfahren, wird zur Bestimmung der Route vom Start zum Ziel lediglich die Information über die Hindernisse herangezogen. Ziel des Voronoi-Algorithmus ist es, aus der rasterbasierten Hinderniskarte einen sicheren Weg zum Ziel zu errechnen. Die genaue Funktionsweise des Voronoi-Algorithmus wird in Abbildung 6.6 dargestellt und im Folgenden erläutert. [THR 96, BUC 99, HIE 01, SET 02]

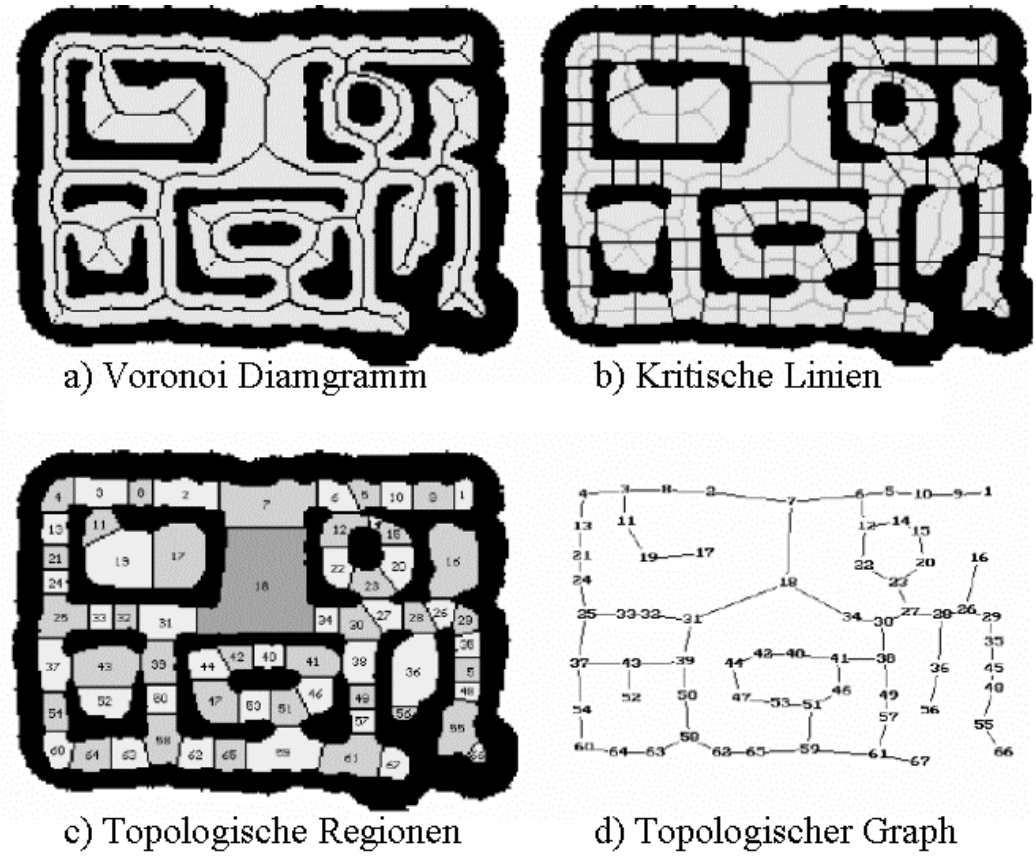


Abbildung 6.6: Voronoi Algorithmus

Voronoi Algorithmus:

1. Vorbereitung

Die Hindernisse werden in einer rasterbasierten Karte repräsentiert. Diese Karte besteht aus Zellen bestimmter Größe, wobei für jede Zelle angegeben ist, ob diese ein Hindernis beinhaltet oder nicht. Die Menge aller freien Zellen wird als Freiraum bzw. free-space (C) bezeichnet und die Menge aller belegten Zellen als occupied-space (C').

2. Voronoi-Diagramm

Für jeden Punkt im free-space <x,y> ∈ C, gibt es einen oder mehrere näheste Punkte im occupied-space C'. Diese nähesten Punkte werden als Basispunkte von <x,y> bezeichnet und die Distanz zwischen <x,y> und seinen Basispunkten als Freidistanz (engl. clearance) von <x,y>.

Das Voronoi-Diagramm ist die Menge aller Punkte im free-space, welche mindestens zwei unterschiedliche (gleich weit entfernte) Basispunkte besitzen. In Abbildung 6.6a ist ein solches Voronoi-Diagramm dargestellt.

3. Kritische Punkte

Um den Freiraum zu unterteilen, werden als nächstes sogenannte kritische Punkte ermittelt. Kritische Punkte <x,y> sind Punkte im Voronoi-Diagramm, welche die Freidistanz lokal minimieren. In anderen Worten: Jeder kritische Punkt <x,y> weist folgende zwei Eigenschaften auf:

a.) Er ist Teil des Voronoi-Diagramms.

b.) Die Freidistanzen aller Punkte in einer ε-Umgebung (Nachbarschaft) von <x,y> sind nicht kleiner als die Freidistanz von <x,y> selbst.

4. Kritische Linien

Kritische Linien werden erstellt, indem jeder kritische Punkt mit seinen Basispunkten verbunden wird (vergleiche Abbildung 6.6b). Kritische Punkte haben genau zwei Basispunkte, andernfalls würden diese kein lokales Minimum

der Freidistanz-Funktion darstellen. Die Kritischen Linien unterteilen den free-space in Regionen, dies wird in Abbildung 6.6c dargestellt.

5. Topologischer Graph

Die unterteilten Bereiche werden in einem topologischen Graphen abgebildet. Jede Region wird dabei als Knoten dargestellt. Die kritischen Linien liefern die Information, auf welche Weise navigiert werden muss, um von einem Knoten zum anderen zu kommen. Sie repräsentieren daher jeweils Bögen welche abgefahren werden sollen.

Mit dem Voronoi-Algorithmus werden demnach jeweils die sichersten Wege bestimmt, um zwischen den bekannten Hindernissen navigieren zu können. Diese sichersten Wege bestehen immer aus Punkten, die genau zwischen mindestens zwei nächstgelegenen Hindernispunkten liegen. Gleichzeitig werden durch die kritischen Linien und Punkte jene Stellen angegeben, bei denen besonders wenig freier Platz (im Vergleich zur lokalen Umgebung) zur Verfügung steht.

Zu beachten gilt es hierbei, dass immer dann, wenn neue Hindernispunkte erkannt werden bzw. bestehende Hindernisse nicht mehr vorhanden sind, sich ein neues Voronoi-Diagramm ergibt, d.h. eine Neuberechnung der betroffenen Areale im Voronoi-Diagramm erforderlich ist. Hauptproblem beim Voronoi-Algorithmus ist, dass er nur dann zu brauchbaren Ergebnissen führt, wenn die Hindernisse und räumlichen Begrenzungen im System bekannt sind. Solange nur eine Wand eines Raum im System bekannt ist und die gegenüberliegende Wand noch nicht detektiert wurde, liefert der Voronoi-Algorithmus keine gültige Route durch diesen Raum. Dies ist auch der Grund, dass bei der hier vorliegenden Umsetzung dieser Algorithmus nicht zur Navigation bzw. im Speziellen nicht zum Ausweichen von Hindernissen verwendet wurde, sondern die unter Punkt 6.1.4. erläuterte Methode. Da zu Beginn keinerlei Informationen über die Wände und Hindernisse bekannt sind, könnte in diesen Fall nicht mit dem Voronoi-Algorithmus navigiert werden. Erst zu späteren Zeitpunkten, wenn die Umgebung ausreichend wahrgenommen wurde, könnte mit Voronoi gearbeitet werden.

Da für die Umsetzung des semiautonomen Rollstuhls in dieser Arbeit auch der Voronoi-Algorithmus zum Zwecke der Navigation in Betracht gezogen wurde, ist zu Versuchszwecken und zur Verdeutlichung der Funktionsweise des Algorithmus im Leitstellenprogramm die Berechnung und eine Möglichkeit zur visuellen Darstellung des Voronoi-Diagramms enthalten.

7. Simulation

Unter Punkt 2.1. wurde bereits eine Einleitung in die Simulation mit Erläuterung ihrer Vor- und Nachteile gegeben. In diesem Kapitel soll nun detaillierter auf die Funktionsweise der Simulation eingegangen werden.

Die Simulation errechnet aufgrund der vom Rollstuhlprogramm gesetzten Aktuatorenwerte mit Hilfe der Kinematik die tatsächliche Position und Ausrichtung des Rollstuhls. Dieses wird im Simulationsprogramm grafisch sowie numerisch dargestellt. In der Simulationsumgebung wird außerdem der Aufbau der zu simulierenden Umgebung mit Wänden, Hindernissen, Leitlinien und Landmarken eingegeben. Eine genaue Programmanleitung, wie diese Eingaben getätigt werden, und Beispiele für die Darstellung derselben am User-Interface, sind unter Punkt 8.1. zu finden. Aufgrund der jeweiligen Position des Rollstuhls in der Umgebung stellt die Simulation die entsprechenden Sensorwerte dem Rollstuhlsteuerungsprogramm zur Verfügung:

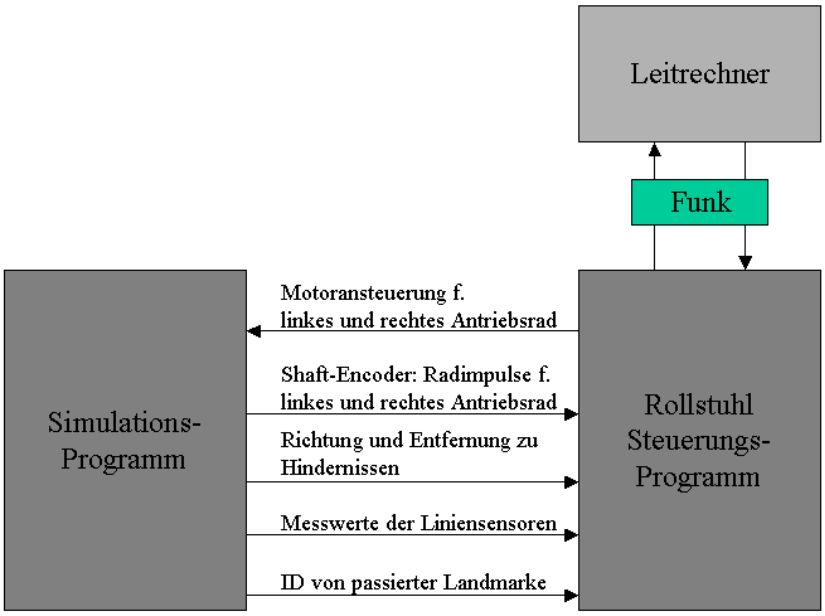


Abbildung 7.1: Information vom und zum Simulationsprogramm

Im Folgenden wird erläutert, wie die einzelnen von der Simulation bereitgestellten Sensorwerte berechnet werden. Die Berechnung der Rollstuhlposition und Ausrichtung aufgrund der Motoransteuerung wird im Simulator wiederum mit Hilfe der Kinematik eines differentialen Steuerungssystems durchgeführt.

7.1. Wiederholungsrate der Sensorberechnung

Einen wesentlichen Faktor bei der Simulation der Sensorwerte stellt die Zeit dar, die zwischen zwei Berechnungen liegt. Leitliniensensor und Shaft-Encoder können in der realen Umgebung jederzeit abgegriffen werden und es werden dann die momentanen Werte geliefert. Bei der Simulation müssen diese gelieferten Werte jedoch berechnet werden. Aus diesem Grund sollte dies in möglichst kurzen Zeitabständen wiederholt werden, um jeweils möglichst aktuelle Werte bereitstellen zu können.

Diese Zeitabstände sind von verschieden Faktoren, wie der Rechnerleistung des verwendeten Computers, von der Anzahl und Intensität von anderen Programmen und Ähnlichem abhängig. Da jedoch auch bei der Verwendung von schwächeren Computern eine Wiederholung der Berechnung ca. alle 30 ms erzielt wird, stellt dies keinerlei Problem dar. Könnten, als extremes Beispiel, die Sensorwerte der Liniensensoren nur zweimal pro Sekunde berechnet wären, wäre es undenkbar, eine Echtzeitsimulation der Linienverfolgung durchzuführen.

Alternativ könnte man die Berechnung der Sensorwerte immer dann durchführen, wenn das Rollstuhlprogramm die jeweiligen Werte abgreifen möchte. Da jedoch das Konzept darauf ausgelegt ist, die Simulation als eigenständigen Prozess zu führen und dadurch modular von dem Rollstuhlprogramm zu trennen, wurde diese Möglichkeit nicht weiter in Betracht gezogen.

Für die Hinderniserkennung kann im Programm ein eigenes Zeitintervall zwischen den Berechnungen festgelegt werden, da auch bei der Umsetzung auf einem realen

Rollstuhl nicht jederzeit die Richtung und die Distanz zu den Hindernissen abgefragt werden kann, sondern dies eine Auswertung der vom Sensor gelieferten Daten erfordert, was auch hier eine maximale Wiederholungsrate bedingt.

7.1. Shaft-Encoder

Die Berechnung der Radimpulse für linkes und rechtes Rad ist relativ einfach. Wichtig hierbei ist, dass das Zeitintervall zwischen diesen Berechnungen kurz gehalten wird, da die kinematischen Berechnungen nur dann korrekt sind, wenn die Geschwindigkeiten der beiden Antriebsräder seit der letzten Berechnung konstant waren. Dies wird bei der Simulation außerdem dadurch gewährleistet, dass bei jeder Übernahme von Motoransteuerungen vom Rollstuhlprogramm auch die Radimpulse neu berechnet werden.

Bei jedem Berechnungsdurchlauf werden die seit dem letzten Durchlauf hinzugekommenen Radimpulse errechnet, und zu den jeweiligen Gesamtsummen für linkes und rechtes Rad addiert. Diese Gesamtsummen stehen dann dem Rollstuhlsteuerungsprogramm zur Verfügung.

Berechnung der neuen Radimpulse für das linke Antriebsrad mittels des zurückgelegten Weges seit der letzten Berechnung. Die Berechnung für das rechte Rad wird auf dieselbe Weise durchgeführt:

$$R_{links} = (v_l / (100 * v_{\max}) * t) / C$$

R_{links}: Neue Radimpulse seit der letzten Berechnung.

t: Zeit seit der letzten Berechnung.

v_l: Geschwindigkeit des linken Rades in Prozent der Maximalgeschwindigkeit (konstant seit der letzten Berechnung).

v_{max}: Geschwindigkeit bei maximalem Radantrieb.

C Konstante, welche angibt, nach wie viel cm jeweils ein Radimpuls ausgelöst wird. Je kleiner dieser Wert, desto korrekter ist die Berechnung der Odometriebewegungen.

Hinweis: Da nur ganze Radimpulse gezählt werden können, wird der sich ergebende Rest bei der nächsten Berechnung berücksichtigt. Die Trägheiten bei Beschleunigung und Abbremsung wurde bei der Simulation vernachlässigt.

Da zwei unterschiedliche Programmprozesse auf dieselben Daten parallel zugreifen, im Speziellen auf die Summen der Radimpulse, muss eine Synchronisation dieser Zugriffe erfolgen. Dies wird durch die Verwendung von sogenannten Events erreicht.

Der Prozess, der auf die Daten zugreifen möchte, versucht dieses Event zu setzten. Wenn es frei ist, so setzt er es auf belegt und kann den Zugriff durchführen. Am Ende gibt der Prozess das Event wieder frei. Sollte das Event nicht frei sein, so wartet der Prozess auf die Freigabe desselben durch den anderen Prozess. Diese Synchronisation gewährleistet, dass nicht beide Prozesse gleichzeitig auf die Daten zugreifen und des Weiteren, dass beim Lesen der Radimpulse immer bereits die Werte für linkes und rechtes Rad aktuell sind. Würde keine Synchronisation erfolgen, so könnte es passieren, dass die Radimpulse für das rechte Rad bereits aktualisiert sind, für das linke Rad die zusätzlichen Impulse jedoch noch nicht addiert wurden. Somit würde die im Rollstuhlprogramm berechnete Odometriebewegung stark fehlerhaft sein.

7.2. Hindernissensoren

Im Kapitel 5 wurde erläutert, dass die Suche nach Hindernissen im Scanbereich jeweils in diskrete Richtungen mit einem bestimmten Scanabstand (=Winkel) zueinander durchgeführt wird, um die Anzahl der möglichen Kollisionspunkte in einen endlichen Rahmen zu bringen. D.h. es werden vom Sensor ausgehend spezifische Geraden vom Rollstuhl wegverlaufend verfolgt und dabei ein Schnittpunkt mit einem Hindernis gesucht. Ein Schnittpunkt einer solchen Scangeraden mit einem Hindernis repräsentiert dann jeweils einen Ergebnispunkt der Hindernissuche (=Kollisionspunkt).

Jede Seite eines in der Simulationsumgebung definierten Hindernisses wird ebenfalls durch eine Gerade repräsentiert, welche durch die Eckpunkte der Hindernisse zu einer Strecke reduziert wird. Ein rechteckiger Schrank wird somit durch vier Geradengleichungen repräsentiert, wobei jede durch zwei Eckpunkte eingeschränkt wird.
Da die Hindernisse, solange sie im Simulator nicht versetzt werden, eine konstante Position aufweisen, sind die korrespondierenden Geraden ebenfalls konstant und werden daher zur Einsparung von Rechenzeit unmittelbar nach der Definition eines Hindernisses im Simulator einmalig berechnet.

Die jeweiligen Geraden der Hindernis-Scanrichtungen müssen dagegen für jede Berechnung der Kollisionspunkte neu bestimmt werden, da diese jeweils abhängig von der Rollstuhlposition und Ausrichtung unterschiedlich sind. Die Geradengleichung für eine Scanrichtung wird dabei durch Sensorposition (=Startpunkt) und Endpunkt (Scanreichweite in Scanrichtung) der Geraden bestimmt.

Bestimmung der allgemeinen Geradengleichung mittels zweier Punkten auf der Geraden:

$P1(x_{P1}, y_{P1})$
$P2(x_{P2}, y_{P2})$
$g : ax + by + d = 0$

für $x_{P2} - x_{P1} \neq 0$**:**	**für** $x_{P2} - x_{P1} = 0$**:**
$a = \frac{y_{P2} - y_{P1}}{x_{P2} - x_{P1}}$	$a = -1$
$b = -1$	$b = 0$
$d = y_{P1} - a * x_{P1}$	$d = x_{P1}$

Um die Ergebnispunkte des Hindernisscans zu bestimmen, werden die Schnittpunkte der Geraden der jeweiligen Scanrichtung mit den Geraden, welche die Hindernisse repräsentieren, ermittelt:

Schnittpunkt (x_s,y_s) zweier Geraden:

$g1 : a_1 x + b_1 y + d_1 = 0$
$g2 : a_2 x + b_2 y + d_2 = 0$

$$x_s = \frac{b_1 d_2 - b_2 d_1}{a_1 b_2 - a_2 b_1}$$

$$y_s = \frac{a_2 d_1 - a_1 d_2}{a_1 b_2 - a_2 b_1}$$

Bedingung: $a_1 b_2 - a_2 b \neq 0$

Nachdem ein Schnittpunkt bestimmt wurde, muss zuletzt noch überprüft werden, ob sich dieser auch auf dem Bereich der Geraden befindet, der das Hindernis darstellt und ob die Distanz zum Sensor innerhalb der Scanreichweite liegt. Dies ist erforderlich, da beim Schneiden der Geraden diese als unendlich angesehen werden, d.h. der sich ergebende Schnittpunkt kann auch weit außerhalb der zu betrachtenden Abschnitte liegen.

Diese Überprüfung erfolgt mittels eines einfachen Koordinatenvergleichs des Schnittpunktes mit den Start und Endpunkten der relevanten Streckenabschnitte auf den Geraden.

In Abbildung 7.2 werden beispielhaft einige mögliche Schnittpunkte dargestellt. S1 ist der Schnittpunkt von der Geraden in Scanrichtung 1 geschnitten mit der Geraden der Hindernisvorderseite. Eine Überprüfung der Koordinaten ergibt hierfür, dass zum Einen der Schnittpunkt im Bereich des Hindernisses liegt, und zum Anderen innerhalb der Scanreichweite zum Sensor. Mit diesem Schnittpunkt wurde ein Hindernis im Scanbereich erkannt, dieser Punkt wird der Rollstuhlsoftware von der Simulation als Sensorergebnis zur Verfügung gestellt.

S2 zeigt, dass auch in Scanrichtung 2 ein Schnittpunkt mit der Hindernisvorderseite gefunden wird, da keine Strecken, sondern Geraden geschnitten werden. Eine Überprüfung der Koordinaten ergibt jedoch, dass der Schnittpunkt nicht im Bereich des Hindernisses liegt, d.h. er befindet sich nicht zwischen den beiden Eckpunkten des Hindernisses, auf der Gerade. Dieser Schnittpunkt wird daher verworfen.

S3 ist der Schnittpunkt in Scanrichtung 2 mit der Rückseite des Hindernisses. Hierbei ist der Schnittpunkt weder im Bereich der Scanreichweite noch zwischen den Eckpunkten des Hindernisses zu finden. Auch dieser Punkt wird verworfen.

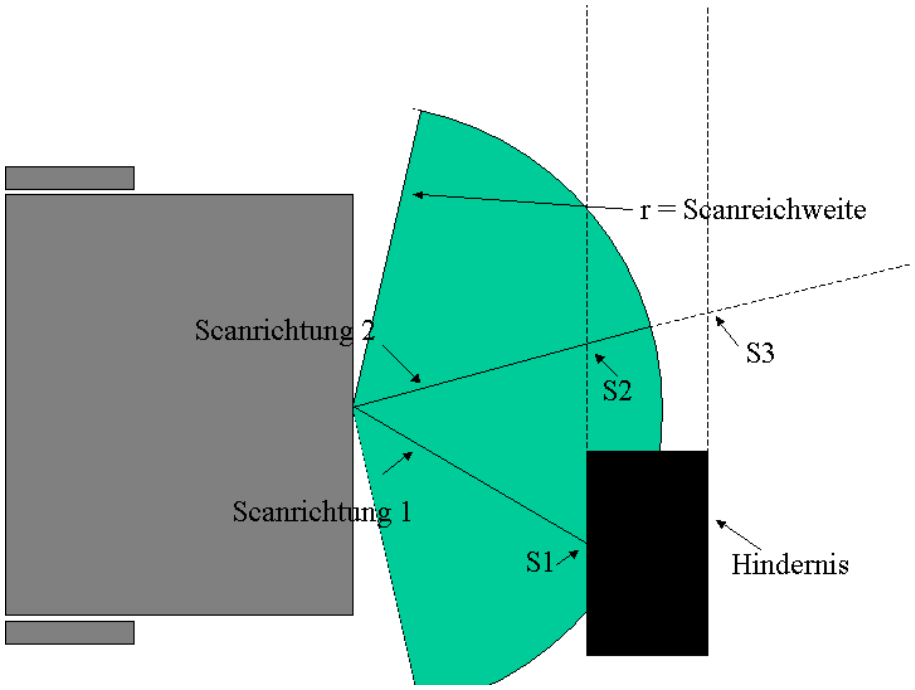


Abbildung 7.2: Simulation der Hinderniserkennung

7.3. Leitlinien-Sensoren

Für die Simulation der Leitlinien-Sensoren muss zunächst eine Annahme über die zu simulierende Methodik der Linienverfolgung und die Art der zu simulierenden Sensoren getroffen werden, da je nach Sensorart andere Ergebnisse geliefert werden.

Zusätzlich ist gerade in diesem Bereich eine Simulation nicht exakt durchzuführen, da hierbei sehr viele Faktoren miteinspielen. Bei der Verwendung eines Infrarot-Emitters und Kollektors zur Liniendetektierung treten beispielsweise je nach Sensormodell höchst unterschiedliche Einflüsse durch das Umgebungslicht auf. Genauso kann die Form und Größe der Reflexionsfläche am Boden von Modell zu Modell unterschiedlich sein bzw. ist diese nicht exakt abgrenzbar. Je nach Bodenart wird bei Infrarot-Sensoren nicht nur von der Leitlinie sondern auch von anderen Bodenbelägen ein gewisser Infrarot-Anteil reflektiert werden. Durch die Verwendung von magnetischen Linien können einige dieser Probleme reduziert werden, aber auch hier können die exakten Ergebnisse nicht vorausgesagt werden.

Es wird daher notwendig sein, weitere Feinabstimmungen der Linienverfolgung bei der Umsetzung an einem Modell durchzuführen, da erst dann sämtliche Eigenschaften ersichtlich werden können.

Annahmen:
Für die in dieser Arbeit umgesetzte Simulation wurde die Annahme getroffen, dass je Liniensensor ein Emitter Infrarotlicht aussendet, welches auf einer kreisförmigen Fläche mit einem Durchmesser, welcher der Breite der Leitlinie entspricht, am Boden auftrifft. Trifft dieses Infrarotlicht auf eine Leitlinie so wird dieses reflektiert und vom Kollektor aufgenommen. Es wurde des Weiteren vereinfachend angenommen, dass von anderen Oberflächen als der Leitlinie kein Infrarotlicht reflektiert wird.

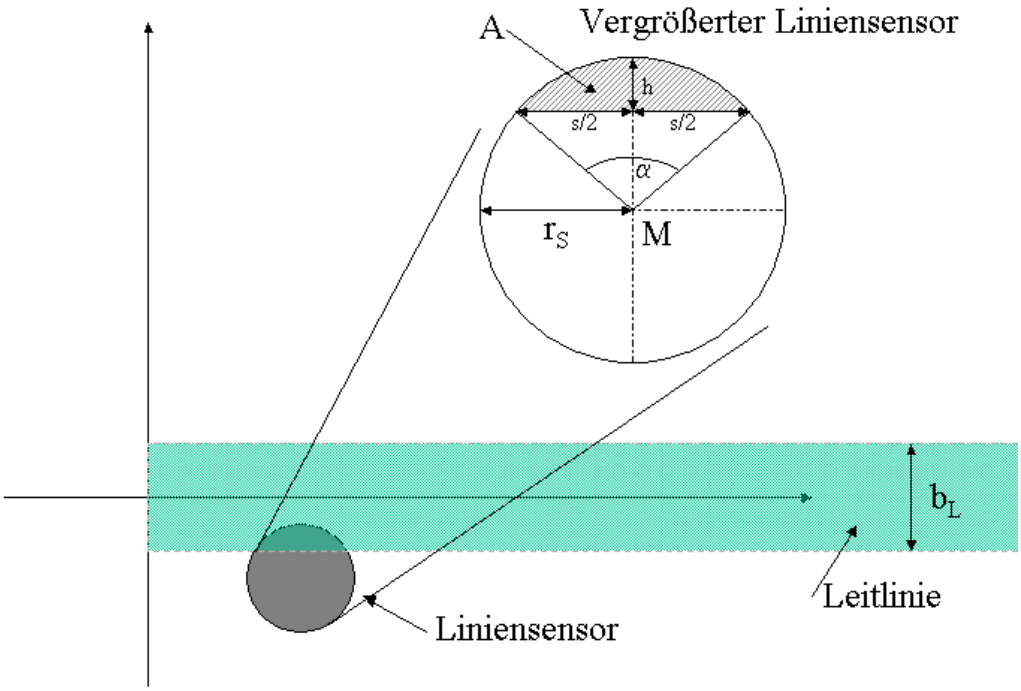


Abbildung 7.3: Simulation der Liniensensoren

Abbildung 7.3 verdeutlicht die Funktionsweise der Berechnung der Liniensensorenwerte. Das Ergebnis einer Sensormessung liegt zwischen 255 wenn keinerlei Überschneidung des Sensors mit der Leitlinie vorliegt und 0 wenn der

Sensor vollständig über einer Leitlinie liegt. Da angenommen wurde, dass der Sensor eine kreisförmige Fläche am Boden nach Leitlinien untersucht, ist für die Simulation die Flächenberechnung des Kreisabschnitts durchzuführen, welcher über der Leitlinie liegt.

Um diese Berechnung durchführen zu können, muss im ersten Schritt die Höhe des Kreisabschnittes, welcher auf der Leitlinie liegt, bestimmt werden. Dazu wird Leitlinie und Sensor so gedreht und verschoben, dass die Leitlinie genau auf Punkt 0/0 beginnt und in Richtung der positiven x-Achse verläuft. Die Höhe h des gesuchten Kreisabschnittes A kann dann einfach mit Hilfe der y-Koordinate des Sensormittelpunktes M bestimmt werden, da die Breite der Leitlinie und der Durchmessers der Sensorreflexionsfläche bekannt sind.

Höhenberechnung der Schnittfläche von Leitlinie und Liniensensor. Vergleiche dazu Abbildung 7.3.

$$h = r - \left(|y_M| - \frac{b_L}{2} \right), \quad \text{für } |y_M| > \frac{b_L}{2}$$

$$h = 2r - \left(r - \left(|y_M| - \frac{b_L}{2} \right) \right), \quad \text{für } |y_M| \leq \frac{b_L}{2}$$

Mit Hilfe der Höhe des gesuchten Kreisabschnittes und dem Radius des Kreises kann nun die Fläche des Kreisabschnittes errechnet werden. Zu beachten gilt es hierbei, dass diese Berechnung nur bei einer Abschnittshöhe h kleiner als der Radius r_S des Kreises nach folgenden Formeln zu berechnen ist. Wenn die Höhe größer als der Radius ist, so wird umgekehrt jene Kreisabschnittfläche berechnet, die nicht auf der Leitlinie liegt und diese von der Gesamtfläche des Kreises abgezogen.

Berechnung der Fläche A eines Kreisabschnittes. Vergleiche dazu Abbildung 7.3.

$$s = 2\sqrt{2hr_s - h^2}$$

$$\alpha = 2\arcsin(\frac{s}{2 * r_s})$$

$$A = \frac{r^2}{2}(\alpha[rad] - \sin(\alpha))$$

Der Anteil der somit berechneten Reflexionsfläche auf der Leitlinie von der gesamten Sensorreflexionsfläche bestimmt dann den jeweils simulierten Sensorwert zwischen 0 (100 % auf Leitlinie) und 255 (0 % auf Leitlinie).

Hinweis: Um Rechenzeit während der Laufzeit einzusparen, werden sämtliche Leitlinien bereits zu Programmstart in der auf den Ursprung verschobenen und auf die x-Achse gedrehten Form berechnet und gespeichert. Um die Berechnung der jeweiligen Höhe des überschneidenden Kreisabschnittes zu berechnen, muss zur Laufzeit daher nur mehr der Sensor entsprechend rotiert und verschoben werden.

7.4. Landmarken-Detektor

Die Simulation der Landmarkendetektierung ist sehr einfach durchzuführen. Der Simulator überprüft lediglich, ob sich der Sensor über einer Landmarke befindet und liefert in diesem Fall die Nummer der Landmarke als Identifikation.

Was es jedoch hierbei zu beachten gilt, ist die richtige Größenwahl der Aufnahmefläche des Sensors. D.h. die Größe des Gebietes, innerhalb dem eine Landmarke erkannt wird, wenn sich der Sensor darüber befindet. Einerseits darf diese Aufnahmefläche nicht zu klein gewählt werden, da sonst der Sensor immer exakt über die Landmarke geführt werden müsste, damit er diese auch als solche erkennt. Bei der Linienverfolgung kann es jedoch zu Korrekturbewegungen

kommen, welche bei einer sehr kleinen Aufnahmefläche dann dazu führen würde, dass oftmals Landmarken „übersehen“ werden.

Andererseits sollte die Aufnahmefläche des Sensors auch nicht zu groß gewählt werden. Je größer die Aufnahmefläche ist, desto ungenauer kann im Zuge der Korrektur der Rollstuhlsposition die exakte Position des Rollstuhls festgelegt werden. Würde z.B. ein Barcodescanner mit einer Aufnahmefläche von 20x10cm als Sensor zur Landmarkendetektierung verwendet werden und dieser eine Landmarke erkennen, so könnte der Leitrechner die momentane Position des Rollstuhlmittelpunktes auch nur auf diese 20x10cm einschränken, aber diese nicht genauer festlegen. Dieses Problem besteht, weil nicht bekannt ist, ob die Landmarke z.B. ganz links oder ganz rechts außen erkannt wurde. Bei großen Aufnahmeflächen des Landmarkdetektors kann es also zu kurzzeitigen Ungenauigkeiten bei der Positionsdarstellung zwischen zwei Landmarken kommen. Durch die Verwendung von Mittelwerten zum Aufbau der Umgebung und die jeweilige Korrektur der Position beim Passieren einer Landmarke werden jedoch größere Probleme vermieden.

Um diese kurzzeitigen Ungenauigkeiten trotzdem reduzieren zum können, gibt es eine weitere Möglichkeit, welche auch in der hier vorliegenden Umsetzung simuliert wurde. Statt nur einen Sensor mit einem gewissen Erkennungsbereich für die Landmarkenerkennung anzubringen, wurde in der Simulation die Möglichkeit geschaffen, mehrere Landmarkdetektoren auf einer Sensorbank nebeneinander anzubringen. Damit kann die Größe des Gebietes, innerhalb dem eine auftretende Landmarke erkannt wird, erweitert werden. Beim Erkennen einer Landmarke wird in diesem Fall zusätzlich zur Identifikation der Landmarke auch die Nummer des Sensors, unter dem die Landmarke erkannt wurde, übertragen. Dadurch kann die relative Position des Rollstuhls zur Landmarke genauer bestimmt werden.

8. Programm-Anleitung

In diesem Kapitel wird eine Anleitung zur Benutzung des Simulationsprogramms und des Leitstellenprogramms gegeben. Da die beiden Programme miteinander kommunizieren, ist es erforderlich, dass beide Programme gleichzeitig laufen. Welches Programm zuerst gestartet wird, ist dabei unerheblich. Wenn jedoch eines der beiden Programme neu gestartet werden soll, so muss auch das andere Programm neu gestartet werden, da ansonsten der Kommunikationsprozess zwischen den beiden Programmen gestört wird. In diesem Fall wird eine entsprechende Fehlermeldung ausgegeben.

8.1. Simulationsprogramm Simroll

Im Simulationsprogramm wird die Umgebung sowie das Verhalten des Rollstuhls in dieser simuliert. In diesem Programm laufen vier Hauptprozesse ab. Zunächst der Prozess der Rollstuhlssteuerung. In diesem Prozess wird das Programm ausgeführt, welches direkt am Rollstuhl abläuft. Des Weiteren ist der Simulationsprozess in diesem Programm enthalten. Dieser nimmt die Stellwerte, d.h. die Aktuatoreinstellungen des Rollstuhlsteuerungsprozesses auf, simuliert das entsprechende Verhalten und stellt die daraus resultierenden Sensorwerte wiederum diesem zur Verfügung. Der dritte Hauptprozess beinhaltet die Kommunikation zwischen Rollstuhlsteuerung und dem Leitrechnerprogramm. Der vierte Prozess ist schließlich für die visuelle Darstellung der simulierten Umgebung zuständig.

8.1.1. Visuelle Darstellung im Simulator

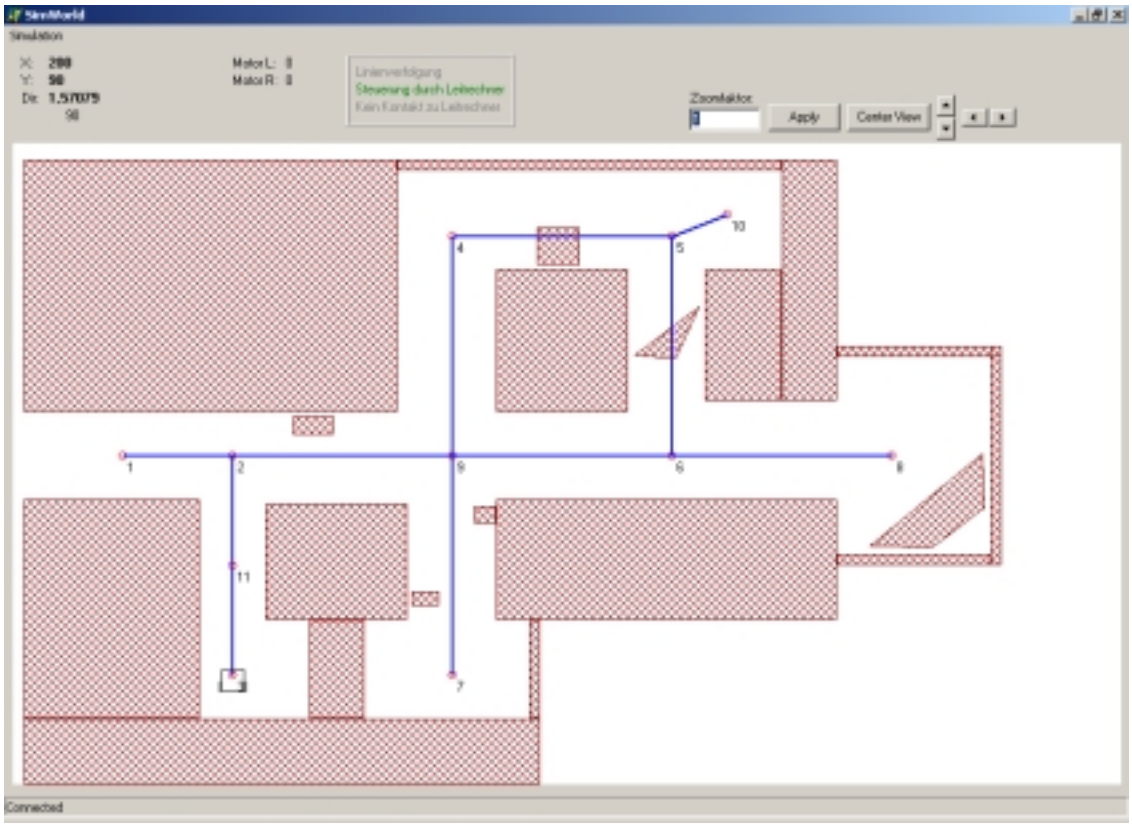


Abbildung 8.1: Hauptbildschirm des Simulationsprogramms

In der Mitte des Bildschirms wird der Grundriss der Umgebung, welche simuliert wird, dargestellt. Hindernisse werden dabei rot schraffiert dargestellt. Leitlinien werden als blaue Linien, und Landmarken als rote Kreise angezeigt. Neben jeder Landmarke ist jeweils die Identifikationsnummer derselben zu finden. Des Weiteren werden die aktuelle Position des Rollstuhls und seine Bewegungen durch die Umgebung visualisiert. Im Simulationsprogramm entspricht die Rollstuhlposition immer seiner tatsächlichen Position. Im Gegensatz dazu ist die Position, welche im Leitrechnerprogramm (siehe dazu Punkt 8.2.) dargestellt wird, die Position, die aus den Sensorwerten errechnet wird und ev. durch Odometriefehler von der tatsächlichen Position abweichen kann.

Am oberen Bildschirmbereich werden auf der linken Seite die Koordinatenwerte der aktuellen Position des Rollstuhls angezeigt. Darunter ist die momentane Ausrichtung des Rollstuhls sowohl in Radiant als auch in Grad ersichtlich. Daneben werden die jeweiligen Motoransteuerungen für das linke und rechte Antriebsrad angezeigt. Wie bereits erwähnt wurde, können diese Motoransteuerungen in einem Bereich von –100 und +100 liegen.

Im mittleren Bereich des oberen Bildschirmteils ist eine Statusanzeige zu sehen. Hier ist ersichtlich, ob der Rollstuhl momentan eine Linienverfolgung ausführt oder direkt die Motoransteuerungen durch den Leitrechner erhält. Dies ist relevant, da die Regelung der Motoren bei der Linienverfolgung direkt durch das Rollstuhlprogramm durchgeführt wird und nicht durch den Leitrechner. Sollte kein Kontakt zum Leitrechner bestehen, so wird dies ebenfalls auf der Statusanzeige angezeigt. Bei der Simulation ist dies im Wesentlichen nur dann der Fall, wenn das Leitrechnerprogramm nicht ausgeführt oder beendet wird.

Im Bereich rechts oben hat man die Möglichkeit, die Darstellung der Umgebung und des Rollstuhls anzupassen. Mit dem Zoomfaktor kann die Ansicht vergrößert oder verkleinert werden. Mit Hilfe des Buttons „Center View“ kann der Bildschirmausschnitt direkt über der momentanen Rollstuhlposition zentriert werden, oder der sichtbare Bereich wird mit den Richtungspfeilen manuell verschoben.

In der Menüleiste kann unter dem Punkt „Simulation“ der Umgebungseditor gestartet werden. Damit können die Leitlinien, Landmarken und Hindernisse bearbeitet werden.

8.1.2. Editieren der simulierten Umgebung

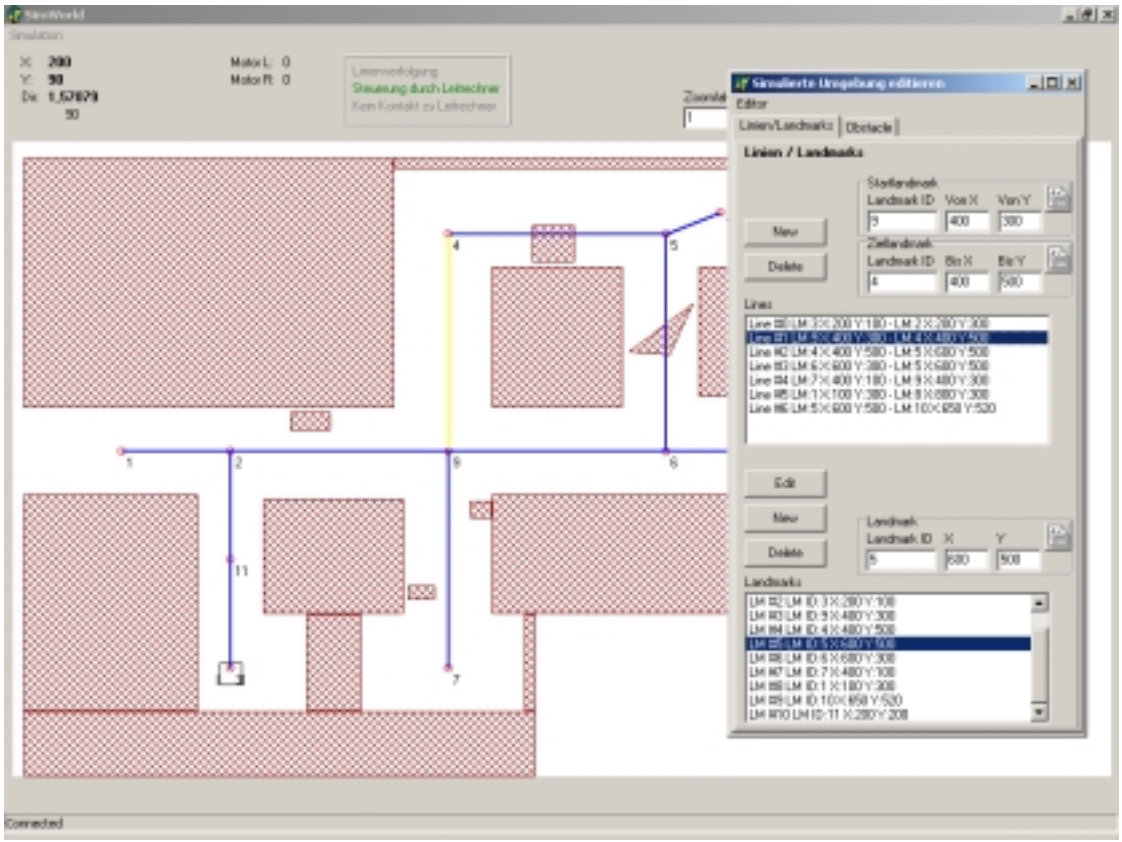


Abbildung 8.2: Editieren der simulierten Umgebung

Nachdem der Editor über die Menüleiste gestartet wurde, erscheint das Editorfenster über der Darstellung der Umgebung. Der Editor ist auf zwei Registerblätter aufgeteilt. Im ersten Registerblatt können die Leitlinien sowie die Landmarken editiert werden und im zweiten Registerblatt können die Hindernisse bearbeitet werden. Für jeden dieser drei Objekttypen wird eine Liste angezeigt, in der alle bereits angelegten Objekte dieses Typs aufgelistet werden.

Wenn in einer Liste eines der enthaltenen Objekte ausgewählt wird, also zum Beispiel eine Leitlinie, so wird dieses in der Darstellung der Umgebung farblich hervorgehoben. In Abbildung 8.2 ist im Editor die Linie von Landmarke 9 nach

Landmarke 4 ausgewählt. Diese Linie ist daher in der Umgebungsdarstellung hervorgehoben.

Durch den Button „Delete“ kann das jeweils ausgewählte Objekt gelöscht werden. Durch den Button „Edit“ können die Daten des gewählten Objektes verändert werden. Und mit „New“ kann ein neues Objekts des jeweiligen Typs angelegt werden.

Alle Objekte werden über ihre Koordinaten in der Umgebung positioniert. Wenn ein Objekt, also eine Leitlinie, eine Landmarke oder ein Hindernis geändert oder neu angelegt werden soll, so können die jeweiligen Koordinaten in die Eingabefelder über den jeweiligen Objektlisten direkt eingegeben werden. Um diese Eingabe einfacher zu gestalten, können diese Koordinaten auch mit Hilfe der Maus eingegeben werden. Hierzu muss auf den kleinen Knopf neben den Eingabefeldern gedrückt werden, um die Mausauswahl zu aktivieren. Anschließend klickt man direkt im Darstellungsbereich der Umgebung auf die gewünschte Position. Die jeweiligen Koordinaten werden dann in die Eingabefelder übernommen.

In der Menüleiste des Editors kann unter dem Punkt „Editor“ der Befehl „Save“ ausgewählt werden, um die durchgeführten Änderungen abzuspeichern.

8.1.2.1. Leitlinien eingeben

Da definiert ist, dass auf jeder Leitlinie an deren Anfang und Ende jeweils eine Landmarke angebracht werden muss, ist auch beim Anlegen einer Leitlinie im Editor jeweils die Startlandmarke und die Ziellandmarke anzugeben. Wenn die eingegebene Landmarke bereits existiert, so werden die jeweiligen Koordinaten nach der Eingabe der Landmarken-Identifikation automatisch ausgefüllt. Wird für eine neue Linie eine Landmarke angegeben, welche noch nicht im System existiert, so wird die Landmarke automatisch angelegt, d.h. sie erscheint in der darunter liegenden Liste der Landmarken.

Die Eingabefelder beim Bearbeiten oder Anlegen einer Leitlinie bestehen wie in Abbildung 8.2 ersichtlich ist, aus Identifikationsnummer und Koordinaten der Startlandmarke sowie die Identifikation der Ziellandmarke wiederum mit ihren Koordinaten.

8.1.2.2. Landmarken eingeben

In der Liste der Landmarken sind bereits all jene, die beim Anlegen der Leitlinien verwendet wurden, enthalten, da diese automatisch bei der Linieneingabe angelegt wurden. Da jedoch jederzeit auf bestehenden Leitlinien zusätzliche Landmarken angebracht werden können, gibt es hier die Möglichkeit, diese in das System einzugeben.

Jede Landmarke muss eine eindeutige Identifikationsnummer erhalten, mittels derer diese vom Sensor unterschieden werden. Diese Identifikation wird während des Programmablaufs dann jeweils an den Leitrechner gesendet. Die Eingabe einer Landmarke besteht daher aus den Feldern ID, X-Koordinate und Y-Koordinate.

8.1.2.3. Hindernisse eingeben

Prinzipiell werden die Hindernisse nur durch ihre Grundfläche repräsentiert, d.h. zwei-dimensional. Die Höhe der jeweiligen Hindernisse wurde für die Simulation in dieser Arbeit nicht berücksichtigt.

Beim Anlegen von Hindernissen im Simulator kann jeweils einer von zwei unterschiedlichen Hindernistypen gewählt werden. Die einfachere Art stellen Hindernisse mit einem rechteckigen Grundriss dar. Alternativ dazu können Hindernisse auch als Polygon definiert werden.

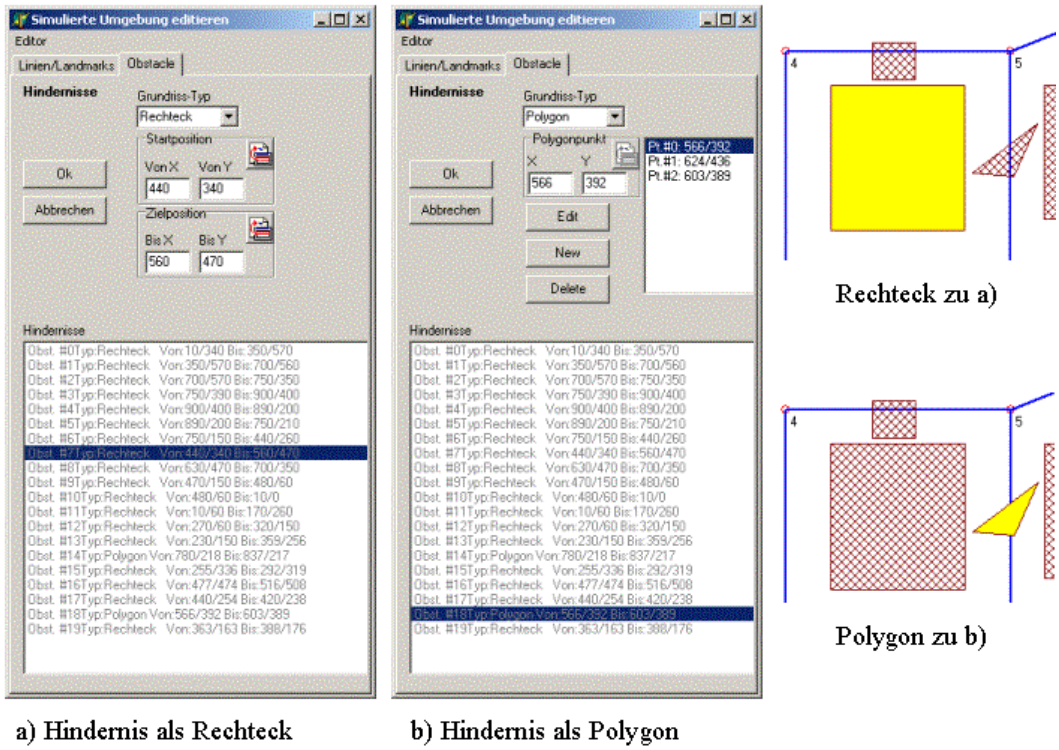


Abbildung 8.3: Eingabe von Hindernissen im Umgebungseditor

Abbildung 8.3 stellt die beiden unterschiedlichen Eingabeanforderungen je nach Hindernistyp dar. Im Fall 8.3a wird ein Hindernis mit rechteckigem Grundriss eingegeben. Bei diesem Typ muss lediglich eine Start- und eine Zielposition des Hindernisses eingegeben werden. Diese Punkte stellen dann jeweils schräg gegenüberliegende Eckpunkte des Rechtecks dar. Zu beachten ist, dass die Seiten des Hindernisses hierbei immer parallel zur x- und y-Achse verlaufend interpretiert werden. Wenn man schräg zu den Achsen verlaufende Hindernisse anlegen möchte, so muss dies in Form von Polygonen getätigt werden.

In Abbildung 8.3b ist die Eingabe eines Hindernisses mit einem Grundriss in Form eines Polygons dargestellt. Hierbei muss nacheinander jeder Eckpunkt des Hindernisses eingegeben werden, die Anzahl der Eckpunkte ist dabei nach oben unbegrenzt. Ein Polygon sollte jedoch zumindest aus drei Eckpunkten bestehen, da andernfalls das Ergebnis ein Punkt oder eine Linie wäre. Auf dem Registerblatt erscheint daher eine weitere Liste, welche die Eckpunkte des Polygons beinhaltet.

Auch hier können diese Punkte wieder durch die Buttons „New“, „Edit“ und „Delete“ bearbeitet werden. Wenn die Definition des Polygons abgeschlossen ist, so ist der Button „OK“ auf der linken Bildschirmseite zu drücken.

8.2. Leitrechnerprogramm Rollrob

Im Leitrechnerprogramm laufen drei zentrale Prozesse ab. Im ersten Prozess findet die eigentliche Steuerung des semiautonomen Rollstuhls statt. Er wertet die Informationen, welche er vom Rollstuhl erhält, aus, um die jeweils erforderlichen Navigationsschritte zu bestimmen und entsprechende Kommandos an der Rollstuhl zu übertragen. Der zweite Prozess beinhaltet die Kommunikation mit der Rollstuhlsteuerung des Simulationsprogramms. Der dritte Prozess ist für die visuelle Darstellung der über die Sensorik wahrgenommenen Umgebung zuständig sowie für allfällige Eingaben des Benutzers oder der Benutzerin.

8.2.1. User-Interface im Leitrechnerprogramm

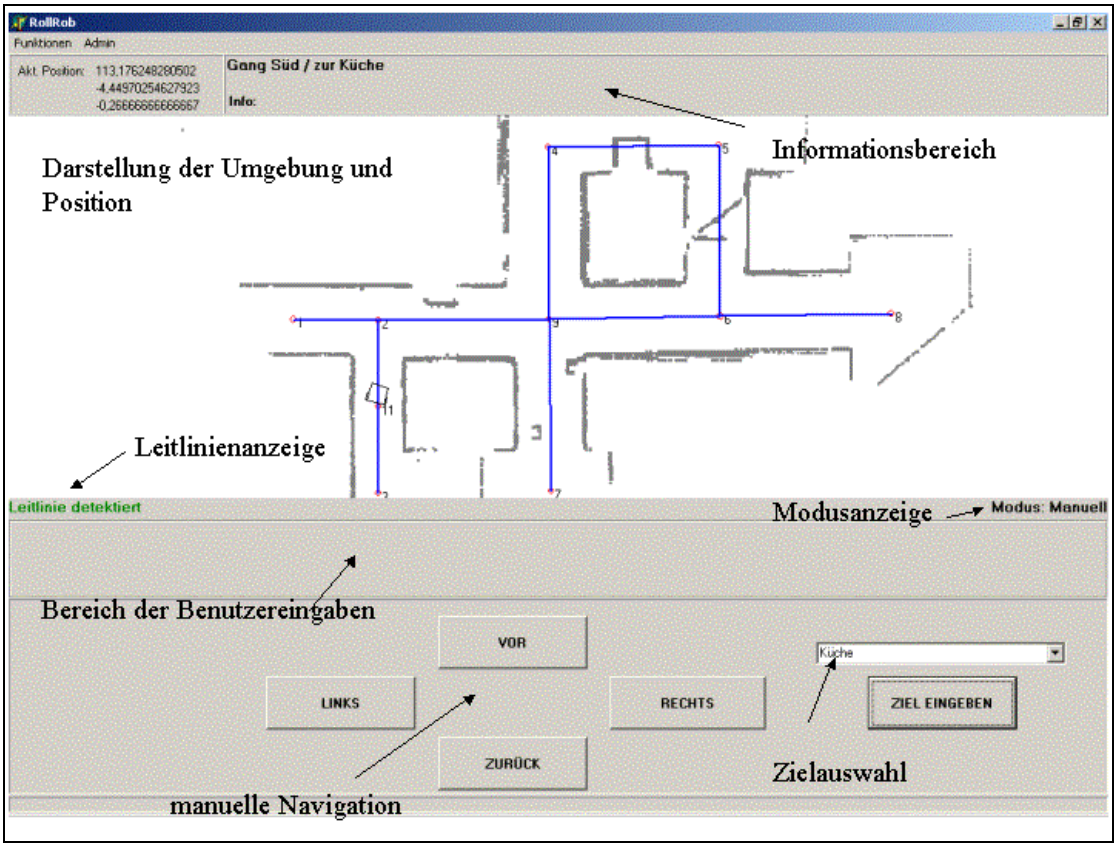


Abbildung 8.4: Hauptbildschirm des Leitrechnerprogramms

Der Hauptbildschirm ist in verschiedene Bereiche aufgeteilt. Im mittleren Bereich wird die Umgebung, so wie der Leitrechner diese mit Hilfe der Sensoren wahrgenommen hat dargestellt. Des Weiteren wird hier die jeweils angenommene momentane Position des Rollstuhls dargestellt, sofern diese bereits bestimmt wurde. Oben links wird Position und Ausrichtung nochmals numerisch dargestellt. Rechts daneben befindet sich der Informationsbereich. Hier wird zum Einen jeweils die zuletzt passierte Landmarke mit ihrer Bezeichnung angezeigt und zum Anderen sonstige Informationen, die während des Programmablaufes dem Benutzer oder der Benutzerin mitgeteilt werden sollen. Ein Beispiel für Letzteres ist eine Benachrichtigung, wenn ein Ziel nicht erreicht werden kann, da ein Hindernis den Weg versperrt und diesem nicht ausgewichen werden konnte.

Unter der visuellen Darstellung der Umgebung befindet sich der Bereich, in dem Eingaben durch den Benutzer oder der Benutzerin vorgenommen werden können, dieser Bereich simuliert also das User-Interface des Rollstuhls. Solche Eingaben sind lediglich im Teaching-Modus erforderlich. Rechts darüber wird der jeweils aktuelle Modus angezeigt. Es gibt drei verschieden Modi: Der Erste ist der Modus „manuell“, dies ist dann der Fall, wenn keine autonome Navigation durchgeführt wird. Letzteres findet im Modus „Navigation“ statt, welcher durch Auswahl eines Ziels durch den Menschen eingeleitet wird. Zuletzt gibt es noch den Modus „Teaching“. Auf der gegenüber liegenden linken Seite ist ersichtlich, ob momentan eine Leitlinie von den Sensoren detektiert wird oder nicht.

Im unteren Bereich befinden sich Knöpfe, um den Rollstuhl manuell zu bewegen. Rechts daneben kann aus einer Liste ein Ziel gewählt werden und durch Drücken des darunter liegenden Buttons „Ziel eingeben“ die automatische Navigation in Richtung des gewählten Ziels gestartet werden. Sollte während einer autonomen Navigation eine Bewegung durch die manuelle Steuerung erfolgen, so wird die Navigation umgehend abgebrochen.

In der Menüleiste können unter dem Punkt „Funktionen“ die „Optionen“ geöffnet werden. Mit Hilfe dieser Optionen kann die aktuelle Darstellung der Umgebung vergrößert, verkleinert, gedreht und verschoben werden. Die jeweiligen Einstellungen werden bei Programmende abgespeichert. Ein weiterer Punkt im Menü „Funktionen“ ist „Teaching“. Durch Auswahl dieses Punktes wird der Teaching Modus, dessen Bedienung im Folgenden genauer erläutert wird, gestartet. Die Auswahlmöglichkeiten im Menü „Admin“ werden unter dem Punkt 8.2.3. beschrieben, da diese Punkte allesamt zur Überwachung und Visualisierung der automatischen Navigation zu einem Ziel dienen.

8.2.2. Anleitung zum Teaching

Der Ablauf und die Funktion des Teachings wurde in den vorangegangenen Kapiteln bereits eingehend erläutert (vgl. Kapitel 4.1.). Hier soll nun noch ergänzend dargestellt werden, wie dieses programmtechnisch durchzuführen ist.

Nachdem im Simulationsprogramm eine Umgebung eingegeben wurde und das Leitrechnerprogramm zum ersten Mal ausgeführt wird, kennt dieses noch keinerlei Umgebungsinformationen. Deshalb kann auch im Darstellungsbereich nichts angezeigt werden. Zusätzlich ist die Rollstuhlposition unbekannt. Als Beispielumgebung dient die in Abbildung 8.1 dargestellte simulierte Wohnung. Es wird nun unter dem Menü „Funktionen“ der Punkt „Teaching“ ausgewählt.

Im Informationsbereich der Maske erscheint nun die Aufforderung, den Rollstuhl auf eine Linie zu bringen, und im Anschluss den ebenfalls neu angezeigten Knopf „Start Teaching“ zu drücken. Hierzu bewegt man den Rollstuhl mit den manuellen Steuerungsknöpfen. Im Simulatorprogramm kann die tatsächliche Position verglichen werden, um eine Linie zu finden. In der Beispielumgebung von Abbildung 8.1 befindet sich der Rollstuhl zu Beginn kurz vor Landmarke 3. Es muss also nur ein wenig nach vorne gesteuert werden. In der Darstellung am Leitrechner kann die Leitlinienanzeige zu Hilfe genommen werden, um zu erkennen, wann eine Linie detektiert wird. Die Richtung des Rollstuhls soll ungefähr in Richtung der nächsten einzulernenden Landmarke gestellt werden. Hat man schließlich den Rollstuhl auf einer Linie positioniert, drückt man „Start Teaching“.

Der Rollstuhl verfolgt nun die Leitlinie bis zur nächsten Landmarke. Beim Beispiel ist dies die Landmarke 11. Nun wird man aufgefordert, eine Bezeichnung für diese neue Landmarke einzugeben und im Anschluss die Eingabe zu bestätigen.

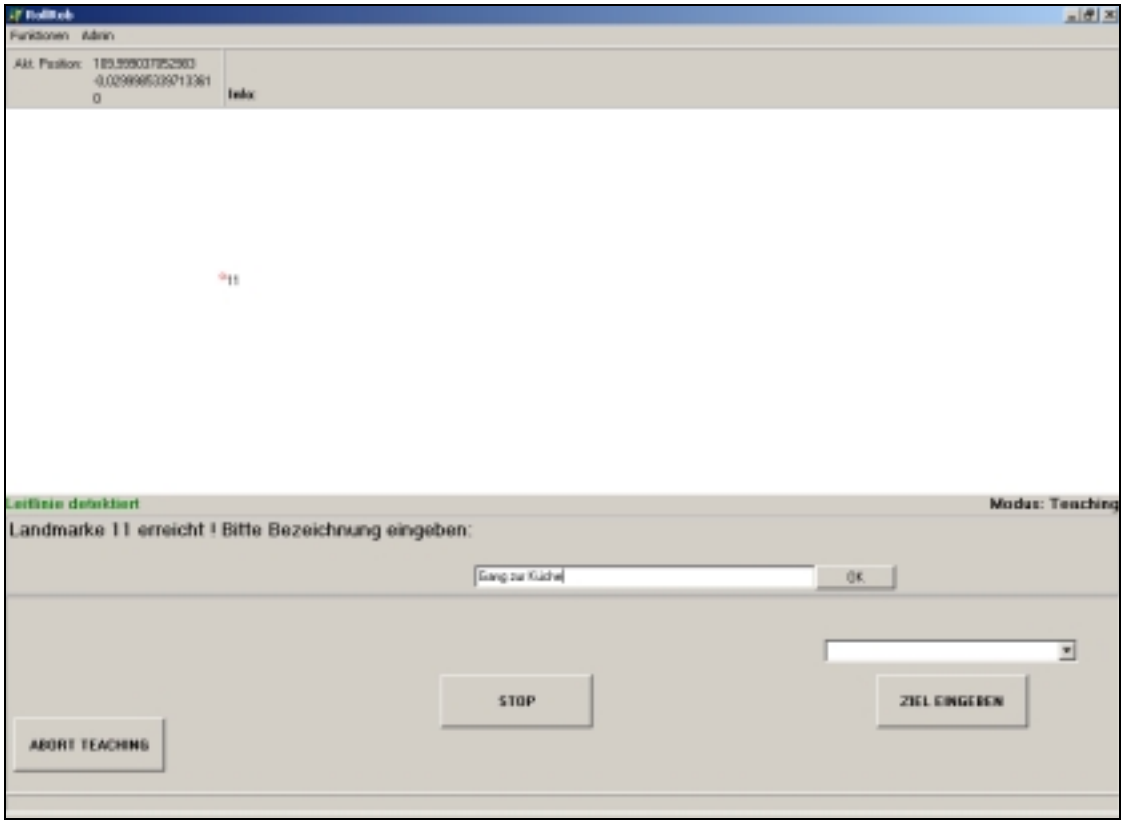


Abbildung 8.5: Erste Landmarke beim Teaching; Position des Rollstuhls noch unbekannt

Der Rollstuhl befindet sich jetzt auf dieser neuen Landmarke. Bevor die nächste Landmarke angefahren wird, muss noch die Richtung durch manuelle Steuerung eingestellt werden. Da im Beispiel die nächste Landmarke in der gleichen Richtung liegt, muss dies in diesem Fall nicht geschehen. Es wird sofort wieder der Knopf „Start Teaching“ gedrückt. Kurz darauf wird die Landmarke Nr. 2 erreicht. Wiederum gibt man eine Bezeichnung dafür ein. Nun wird man zusätzlich gefragt, ob die direkte Leitlinie zwischen Landmarke 11 und Landmarke 2 korrekt ist, was mit „ja“ zu beantworten ist. Verneinen müsste man diese Frage dann, wenn zwischen Landmarke 11 und Landmarke 2 eine weitere Landmarke liegen würde, welche beim Teaching nicht erkannt wurde.

Da nun zwei Landmarken auf einer Linie abgefahren wurden, kann, wie in Kapitel 4.1. erklärt wurde, bereits die Rollstuhlposition und Ausrichtung berechnet und angezeigt werden. Wiederum dreht man den Rollstuhl in Richtung jener Landmarke, die als nächstes angefahren und eingelernt werden soll. Als Beispiel wählen wir die

Landmarke Nr. 9 als nächstes Ziel aus. Während man sich ungefähr 90 Grad nach rechts dreht, kann ersehen werden, dass dadurch dass die Rollstuhlposition nun bekannt ist, auch bereits die nächstgelegenen Hindernispunkte eingezeichnet werden. Mit „Start Teaching“ kann die Linienverfolgung in Richtung Landmarke 9 gestartet werden.

In Abbildung 8.6 ist der Bildschirm des Leitrechners abgebildet, wie er sich nach dem Erreichen der Landmarke 9 und der Eingabe einer Bezeichnung für selbige darstellt. Alle bekannten Leitlinien und Landmarken können sofort zur semiautonomen Navigation verwendet werden. Das Teaching kann jederzeit abgebrochen und zu einem späteren Zeitpunkt fortgesetzt werden. Im Normalfall sollte man nun aber auch alle anderen Landmarken auf die gleiche Weise einlernen.

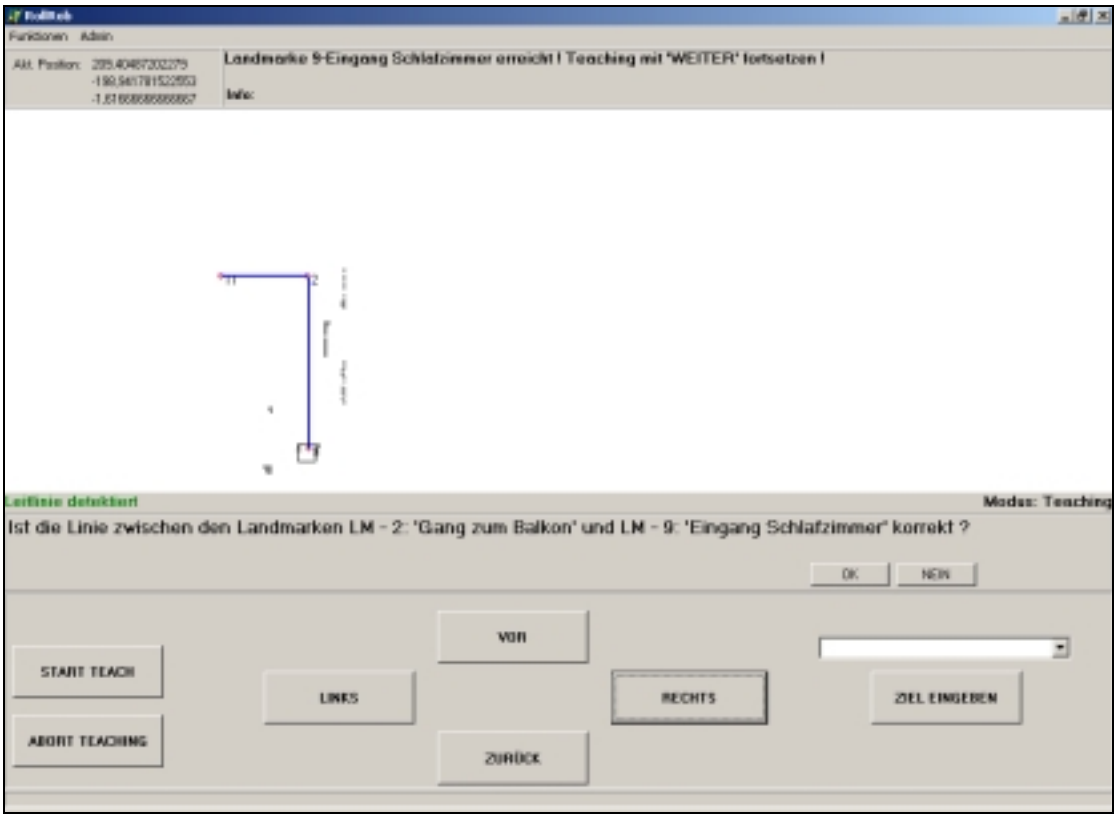


Abbildung 8.6: Dritte Landmarke beim Teaching

Sollte der Rollstuhl bei der Linienverfolgung auf ein Hindernis stoßen, welches die Leitlinie blockiert, so stoppt er automatisch. Der Benutzer oder die Benutzerin muss

dann manuell dem Hindernis ausweichen, sich danach wieder auf der Linie positionieren und in Richtung Ziellandmarke das Teaching erneut starten. Der Rollstuhl fährt dann wieder der Linie entlang bis zur nächsten Landmarke und lernt diese ein. Erst im Navigationsmodus fährt er selbstständig um das Hindernis herum.

8.2.3. Navigation und Zusatzfunktionen

Die Bedienung des Programms nach der Durchführung des Teachings ist sehr einfach. Man kann nun in der Zielauswahl eine der eingelernten Landmarken als Ziel auswählen und über den Button „Ziel eingeben“ die automatische Navigation zu diesem Ziel starten. Nun kann man die Bewegung des Rollstuhls sowohl im Leitrechnerprogramm als auch im Simulationsprogramm beobachten bzw. somit die vom Leitrechner angenommenen und die tatsächlich durchgeführten Bewegungen vergleichen. Mit den Knöpfen im Bildschirmbereich der manuellen Navigation kann der Rollstuhl beliebig bewegt werden. Eine eventuell zu diesem Zeitpunkt laufende automatische Navigation wird dann abgebrochen. Zu beachten ist, dass auch bei der manuellen Navigation durch die Rollstuhlsoftware automatisch gestoppt wird, wenn ein Hindernis in einen gewissen Nahbereich eintritt und weiter in diese Richtung gesteuert wird.

Während der Navigation können über die Menüleiste unter dem Punkt „Admin“ verschiedene Zusatzfunktionen aktiviert werden, welche allesamt nur zur Demonstration und zum besseren Verständnis der internen Programmabläufe während der Navigation dienen. Diese Möglichkeiten werden im Folgenden erläutert:

Menüpunkt 1: Fahrbare Flächen und Ausweichroute berechnen
Dies dient zur Darstellung des Verhaltens beim Ausweichen von Hindernissen, wie es in Kapitel 6.1.4. beschrieben wurde. Es wird der befahrbare Bereich sowie die jeweilige Ausweichroute im Falle eines Hindernisses zum nächsten Ziel berechnet. Diese Berechnung wird automatisch immer dann intern durchgeführt, wenn ein Hindernis in eine gefährliche Nähe gerückt ist und das Ausweichmanöver gestartet

wird. Dieser Menüpunkt dient dazu, diese Berechnungen auch zu beliebigen Zeitpunkten durchführen und anzeigen zu können. Um nach der Berechnung ein Resultat am Bildschirm sehen zu können, muss entweder der Punkt „Fahrbare Flächen / Voronoi einzeichnen“ oder der Punkt „Ausweichroute einzeichnen“ auf „Ja“ gesetzt werden.

Menüpunkt 2: Voronoi berechnen

Der unter Punkt 6.3. beschriebene Voronoi-Algorithmus findet in der umgesetzten Navigation keine Verwendung, da er sich bei der Simulation als nicht zielführend herausgestellt hat. Nachdem die Berechnung jedoch bereits umgesetzt wurde, wurde diese trotzdem im Programm belassen und kann durch diesen Menüpunkt demonstrativ durchgeführt werden. Damit das entsprechende Ergebnis am Bildschirm dargestellt wird, muss der Punkt „Fahrbare Flächen / Voronoi einzeichnen“ auf „Ja“ gesetzt werden.

Menüpunkt 3: Fahrbare Flächen / Voronoi einzeichnen

Um nach jeweiliger Auswahl von Menüpunkt 1 oder Menüpunkt 2 das entsprechende Ergebnis grafisch dargestellt zu bekommen, muss diese Option auf „Ja“ gesetzt werden. Um die Darstellung wieder zu entfernen, wird er auf „Nein“ gesetzt. Wenn dieser Punkt angewählt ist, so werden auch nach internen Berechnungen der befahrbaren Flächen die Ergebnisse angezeigt. Der befahrbare Bereich bzw. sämtliche Voronoi-Punkte werden in rot eingezeichnet. Ein Beispiel für die Darstellung der befahrbaren Flächen ist in Abbildung 6.4b ersichtlich.

Menüpunkt 4: Ausweichroute einzeichnen

Ebenso auf „Ja“ zu setzten, wenn die mittels Menüpunktes 1 oder intern beim Durchführen eines Ausweichmanövers automatisch durchgeführte Berechnung der jeweiligen Ausweichrouten am Bildschirm dargestellt werden soll. Mit „Nein“ wird diese Darstellung wieder entfernt. Wie die Ausweichroute dargestellt wird, ist in Abbildung 6.5 ersichtlich.

Menüpunkt 5: Koordinaten auf Klick

Wenn dieser Punkt auf „Ja“ gestellt wird, erscheinen nach jedem Mausklick im Darstellungsbereich die jeweiligen Koordinaten des gewählten Punktes. Hierbei erscheinen sowohl die Absolutkoordination des Punktes, als auch die Koordinaten dieser Position im Hindernisraster. Dieser Punkt ist hilfreich bei Arbeiten am Programm und wurde im Sinne von späteren Erweiterungen etc. beibehalten.

Menüpunkt 6: Debug

Mit diesem Menüpunkt wird ein Fenster geöffnet, in welchem während einer Navigation sämtliche durchgeführten Navigationsverhaltensweisen textuell in einer Liste angezeigt werden. Zusätzlich werden wichtige Ereignisse, wie zum Beispiel das Verlieren einer Leitlinie bei der Linienverfolgung in dieser Liste angezeigt. Somit kann nachvollzogen werden, warum der Rollstuhl das jeweilige Verhalten aufweist.

9. Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde ein Systemkonzept erstellt und simuliert, um einem elektrischen Rollstuhl mit einer einfach gehaltenen Sensorausstattung zu ermöglichen, automatisch ein von dem Benutzer oder der Benutzerin ausgewähltes Ziel zu erreichen. Zur Vorbereitung der semiautonomen Navigation muss kein genauer Plan der Umgebung, in der sich der Rollstuhl durch automatische Steuerung bewegen können soll, in das System eingegeben werden. Stattdessen nimmt der Rollstuhl die Beschaffenheit der Umgebung mit Raumbegrenzungen und Hindernissen aktiv während des Betriebes wahr. Jedoch muss die Umgebung mit geraden Leitlinien und Landmarken ausgestattet werden und diese dem Leitrechner durch einmaliges benutzergeführtes Abfahren, dem sogenannten Teaching, „beigebracht“ werden.

Als Ausstattung des Rollstuhls wurden zwei Sensoren an der Unterseite zur Detektierung der Leitlinie vorgesehen. Ein oder wahlweise mehrere Sensoren ebenfalls an der Unterseite zur Erkennung und Identifizierung der jeweiligen Landmarken, ein Hindernissensor an der Vorderseite des Rollstuhls und für jedes der beiden Antriebsräder jeweils ein Sensor, der die durchgeführten Bewegungen abtastet. Über Odometrieberechnungen werden aus den jeweiligen Teilbewegungen die Position und die Ausrichtung des Rollstuhls im Leitrechner berechnet. Die dabei entstehenden Odometriefehler werden beim Erreichen einer Landmarke, welche als Referenzpunkt verwendet wird, wieder ausgeglichen.

Es wurde in der Umsetzung vorgesehen, dass Leitrechner und Rollstuhlprogramm getrennt voneinander arbeiten und über Funk miteinander kommunizieren. Dies wurde deshalb vorgesehen, da als weitere Schritte geplant wurde, diese Umsetzung auf einem Modell eines Rollstuhls zu betreiben, und am Modell kein Platz für den Leitrechner mit User-Interface (und User) vorhanden ist. Bei der Umsetzung auf einem lebensgroßen elektrischen Rollstuhl kann Rollstuhlsteuerung und Leitrechner zusammengelegt werden, was zu zusätzlichen Vorteilen führt, da die Funksteuerung,

welche einen gewissen Engpass vor allem bei der übertragbaren Hindernisinformationen darstellt, eingespart werden kann.

Die Navigation läuft in erster Linie über die topologische Karte der Leitlinien und Landmarken ab, d.h. der Rollstuhl versucht so lange wie möglich den Leitlinien zu folgen, um so die gewünschten Ziele zu erreichen. Sollte dies durch das Auftreten eines Hindernisses nicht weiter möglich sein, so wird mit Hilfe der aktuellen Hindernisinformationen der Sensoren und den im Hindernisraster gespeicherten Informationen versucht, eine Ausweichroute zu bestimmen und das Hindernis damit zu umfahren.

Im Simulator kann eine Umgebung entworfen und manipuliert werden, um das Verhalten des Rollstuhls in dieser zu testen und zu visualisieren.

Im Resultat ist ersichtlich, dass der Rollstuhl mit dem umgesetzten Systemkonzept sehr sicher die einzelnen Ziellandmarken erreicht. Auch auftretenden Hindernissen kann er gut ausweichen. Als Ausblick sollte jedoch bei weiteren Verfeinerungen die Anzahl der Korrekturbewegungen dabei noch reduziert werden, um den Benutzern bzw. Benutzerinnen mehr Komfort zu bieten, da jede Richtungsänderung für diese unangenehm ist. Generell wurden bei der Umsetzung die Teilbewegungen der Navigation noch nicht allesamt optimiert, d.h. es sollte in weiteren Schritten versucht werden, die Bewegungen des Rollstuhls für den Benutzer bzw. die Benutzerin so angenehm wie möglich zu gestalten. Durch die Verwendung nur eines Sensors zur Hinderniserkennung der nach vorne gerichtet ist, entstehen gewisse blinde Bereiche bei der Navigation, welche durch das Anbringen von zusätzlichen Hindernissensoren zur Seite und nach hinten ausgeschaltet werden sollten.

Das aufgestellte und simulierte Konzept erfüllt sehr zufriedenstellend die gestellten Anforderungen der semiautonomen Navigation und erlaubt mit Hilfe des Teaching eine einfache Inbetriebnahme in einer neuen Umgebung mit relativ einfacher Sensorausstattung. Somit kann in weiteren Schritten dieses Konzept auf einem Modell oder lebensgroßen Rollstuhl weiterverfolgt werden.

Auf der dieser Arbeit beiliegenden CD sind der Programmcode sowie die ausführbaren Versionen der Programme RollRob und SimRoll enthalten. Sämtliche Funktionen wurden im Code kommentiert. Auf der CD ist außerdem ein Readme zu finden (Readme.txt), welches eine einfache Anleitung zur Installation enthält.

Literatur

[ALE 88] Alexander J.C., Maddocks J. H., On The Kinematics of Wheeled Mobile Robots, Maryland, USA, 1988

[BUC 99] Buck M., Schäfer D., Noltemeier H, Practical Strategies for Hypotheses Elimination on the Self-Localization Problem, Report No. 236, Würzburg, 1999

[FEH 00] Fehr L, Langbein W.E., Skaar S.B., Adequacy of power wheelchair control interfaces for persons with severe disabilities: A clinical survey, In Journal of Rehabilitation Research and Development Vol. 37 No. 3, May/June 2000, Pages 353-360, 2000

[FOX 96] Fox D., Burgard W., Thrun S., Controlling Synchro-drive Robots with the Dynamic Window Approach to Collision Avoidance, In Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems, Osaka, Japan, 1996

[FOX 98] Fox D., Burgard W., Thrun S., Active Markov Localization for Mobile Robots, 1998

[FOX 99] Fox D., Burgard W., Dellaert F., Thrun S., Monte Carlo Localization: Efficient Position Estimation for Mobile Robots, In: AAAI/IAAI 1999, Pages 343-349 , Florida, USA, 1999

[FOX 01] Fox D., Thrun S., Burgard W., Dellaert F., Particle Filters for Mobile Robot Localization, In Sequential Monte Carlo Methods in Practice (Autoren: A. Doucet, J.F.G. de Freitas, N.J. Gordon), Springer-Verlag, New York, 2001

[GOM 98] Gomi T., Griffith A., Developing Intelligent Wheelchairs for the Handicapped, Ontario, Canada, 1998

[HIE 01] Hieß S., Schäfer D., Taskbasierte Positionsverfolgung und Zielsuche autonomer mobiler Roboter, Würzburg, 2001

[JÖR 99] Jörg K.W:, Gattung T., Weber J., Supporting Mobile Robot Localization by Visual Bar Code Recognition, Proceedings of the 1999 IASTED Conference on Robotics and Applications, Santa Barbara, California, USA, 1999

[KAM 95] Kamon I., Rimon E., A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots, Technion, Israel, 1995

[KOL 97] Kollmann J., Lankenau A., Bühlmeier A., Krieg-Brückner B., Röfer T., Navigation of a Kinematically Restricted Wheelchair by the Parti-Game Algorithm, Bremen, 1997

[LAN 01] Lankenau A., Röfer T., Selbstlokalisation in Routengraphen, Bremen, 2001

[LEV 99] Levine S.P., Bell D.A., Jaros A.J., Simpson R.C., Koren Y., Borenstein J., The NavChair Assistive Wheelchair Navigation System, In IEEE Transactions on Rehabilitation Engineering, vol. 7, no. 4, Pages 443-451, 1999

[LUC 01] Lucas G.W., A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators, The Rossum Project, http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html, 2001

[MEN 95] Menezes P., Dias J., Araújo H., A. de Almeida, Low Cost Sensor Based Obstacle Detection and Description, Experiments with Mobile Robots using Grid Representation, Coimbra, Portugal, 1995

[MÜL 99] Müller R., Röfer T., Lankenau A., Musto A., Stein K., Eisenkolb A., Coarse Qualitative Description in Robot Navigation, 1999

[MUS 00] Musto A., Stein K., Eisenkolb A., Röfer T., Brauer W., Schill K., From Motion Observation to Qualitative Motion Representation, Deutschland, 2000

[RÖF 97] Röfer T., Routemark-based Navigation of a Wheelchair, Bremen, 1997

[RÖF 98] Röfer T., Routenbeschreibung durch Odometrie-Scans; Bremen, 1998

[RÖF 98_2] Röfer T., Strategies for Using a Simulation in the Development of the Bremen Autonomous Wheelchair, Bremen, 1998

[RÖF 99] Röfer T., Lankenau A., Ensuring Safe Obstacle Avoidance in a Shared-Control System, In J. M. Fuertes (Hrsg.): Proc. of the 7th Int. Conf. on Emergent Technologies and Factory Automation. 1405 - 1414., Bremen, 1999

[SCH 98] Schildt H., Kastner W., Prozessautomatisierung, Springer-Verlag, Wien, 1998

[SET 02] Setalaphruk V., Uneno T., Kono Y., Kidode M., Topological Map Generation from Simplified map for Mobile Robot Navigation, Nara, Japan, 2002

[SIM 99] Simpson R.C., Levine S.P., Automatic Adaptation in the NavChair Assistive Wheelchair Navigation System, In IEEE Transactions on Rehabilitation Engineering, vol. 7, no. 4, Pages 452-463, 1999

[THR 00] Thrun S., Beetz M., Bennewitz M., Burgard W., Cremers A.B., Dellaert F., Fox D., Hähnel D., Rosenberg C., Roy N., Schulte J., Schulz D., Probalistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva, 2000

[THR 96] Thrun S., Bücken A., Integrating Grid-Based and Topological Maps for Mobile Robot Navigation, Proceedings of the Thirteenth National Conference on Artificial Intelligence AAAI, Oregon, USA, 1996

[WAK 92] Wakaumi H., Nakamura K., Matsumura T., Development of an automated wheelchair guided by a magnetic ferrite marker lane, In Journal of Rehabilitation Research and Development Vol. 29 No. 1, Pages 27-34, 1992