Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Martin Mustermann

Entwicklung und Aufbau eines mikrorechnergesteuerten Bestückungsautomaten

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Martin Mustermann

Entwicklung und Aufbau eines mikrorechnergesteuerten Bestückungsautomaten

Masterthesis eingereicht im Rahmen der Masterprüfung
im Masterstudiengang Automatisierung
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Martin Zapf
Zweitgutachter : Prof. Dr.Ing. Armin Kluge

Abgegeben am 3. Juni 2018

Martin Mustermann

Thema der Masterthesis

Entwicklung und Aufbau eines mikrorechnergesteuerten Bestückungsautomaten

Stichworte

Steuerung, und viele weitere interessante Stichwort

Kurzzusammenfassung

Diese Arbeit umfasst alles was man mit einem Mikrorechner machen kann und natürlich noch vieles mehr. etc.

Martin Mustermann

Title of the paper

Development and Construction of a Microprocessor controlled allocation processor

Keywords

Controller, Microprocessor, and other interesting words describing the whole process

Abstract

Inside this report the construction of a very important Controller for microproc-essors is described. etc.

Danksagung

An dieser Stelle kann man vielen Leutchen danken... Ä

Inhaltsverzeichnis

Tabellenverzeichnis **7**

Abbildungsverzeichnis **8**

1 Einführung **9**

2 Aufgabenstellung **10**

3 Bahnplanung **11**
3.1 Lokalisierung . . . 11
3.2 Kartenerstellung . . . 12
3.3 Trajektoriengenergierung . . . 12
3.3.1 Globale Bahnplanung . . . 13
3.3.2 Lokale Bahnplanung . . . 16
3.4 Ausgewähltes Konzept . . . 19
3.4.1 Gesamtkonzept . . . 19

4 Konzept **23**
4.1 Potentialfeld zur Bahngenerierung und Kollisionsvermeidung . . . 24
4.1.1 Roboter Umwelt . . . 24
4.1.2 Robotino,Sationen und IR-Hindernisse . . . 27
4.1.3 Ziel . . . 27

5 Simulation und Testplanung **29**
5.1 Simulation des AusweichmanÃ¶vers . . . 29
5.2 Simulation Wartebereiche . . . 32
5.3 Geschwindigkeit bestimmen . . . 32
5.3.1 IR Daten erfassen . . . 33
5.3.2 Fahrende Robotinos erkennen . . . 33
5.3.3 Geschwindigkeitsvektor bestimmen . . . 33
5.3.4 Ausweichrichtung bestimmen . . . 33
5.3.5 AusweichmanÃ¶ver addieren . . . 33

6 Robotino 2.0 **34**

7 Validierung **35**

Literaturverzeichnis **36**

Tabellenverzeichnis

3.1 Bahnplanungsmethoden nach ihrer Dynamik, Sicherheit, Rechenzeit und Zuverlässigkeit bewertet 19

Abbildungsverzeichnis

3.1 Beispiel Sichtbarkeitsgraph, Graue Felder sind Hindernisse, blaue Linie ist optimaler Pfad . . . 13
3.2 Beispiel Occupancy Grid, Rot-Hindernisse, Weiß-befahrbarer Raum . . . 14
3.3 Beispiel Occupancy Grid mit D*, Rot-Hindernisse, Intensität des Graus im Hintergrunds repräsentiert die Entfernung zum Ziel; blauer Punkt entspricht Ziel . 15
3.4 Bu2-Algorithmus, Grauen Flächen: Hindernisse, Roter Punkt: Startpunkt, Grüner Punkt: Ziel, Verbindungslinie: m-Linie . . . 16
3.5 Bu2-Algorithmus, Fahrweg des Roboters ist in orange dargestellt . . . 17
3.6 Bu2-Algorithmus, möglicher Deadlock, Fahrweg des Roboters ist in orange dargestellt . . . 18
3.7 Beispiel Potentialfeld . . . 18
3.8 Konfigurationsraum . . . 20
3.9 Potentialfeld des Konfigurationsraumes eingezäunt in hohe Potentiale zur eindeutigen Begrenzung des Raumes, Stationen als ein hohes Potential dargestellt, Rampen und Fahrrinnen zum leiten der Robotinos, . . . 21

4.1 Bereichseinteilung . . . 23
4.2 Potentialfeldzonen . . . 25
4.3 Potentialfeld der einzel Zonen . . . 26
4.4 Potentialfeld der Roboter Umwelt ohne RausfÃ¼hrung . . . 26
4.5 Potentialfeld der Roboter Umwelt mit RausfÃ¼hrung . . . 27
4.6 Potentialfeld der gauÃschen radialen Basisfunktion . . . 27
4.7 Potentialfeld Ziel . . . 28

5.1 Simulation mit Ausweichfunktion . . . 30
5.2 Simulation ohne Ausweichfunktion . . . 31
5.3 Subsystem Vektorberechnung . . . 32
5.4 Ablaufplan des Subsystems Vektorberechnung . . . 33

1 Einführung

Bei dem Stichwort „Autonome Systeme“ fällt der Gedanke schnell auf Industrie 4.0. Die vierte industrielle Revolution, wie sie von dem Bundesministerium für Wirtschaft und Energie genannt wird, beschreibt die selbstorganisierte Produktion durch intelligente und digitale Systeme. Ein solches autonomes System soll als Produktionsstraße im Verbundprojekt der Hochschule für Angewandte Wissenschaft Hamburg mit Hilfe von Transportrobotern, sogenannten Robotions, realisiert werden.

Zur Realisierung dieser Produktionsstraße werden bereits zu Beginn der Projektarbeit alle notwendigen Hardwarekomponenten zur Verfügung gestellt. Zu diesen Hardwarekomponenten gehören die Robotinos. Ein Robotino ist ein mobiles Robotersystem mit omnidirektionalem Antrieb, der es dem Roboter ermöglicht zu jeder Zeit in jede beliebige Richtung fahren zu können. Sie werden mittels ArUco-Marker und Deckenkameras lokalisiert. Bei den zu transportierenden Werkstücken handelt es sich um runde Bausteine. Diese Bausteine sind mit einem RFID-Transponder ausgestattet und können von den Lesegeräten an den Stationen gelesen werden. Insgesamt gibt es vier Stationen, die beidseitig angefahren werden können. Diese Stationen repräsentieren die Lager bzw. Maschinen, zu denen die Werkstück, je nach Auftrag, transportiert werden müssen. Zwei zusätzliche Stationen sind als Ladestationen für die Robotinos ausgelegt.

Auf Grund der Komplexität dieses Projektes, wird das Gesamtprojekt in einzelne Aufgabenpakete unterteilt, welche in Gruppenarbeit von drei bis vier Personen zu bearbeiten sind. Insgesamt gibt es fünf Gewerke, darunter die Auftragskoordination, Bahnplanung und Regelung, deren Ziel es ist ein funktionsfähiges und zuverlässiges Gesamtsystem zu entwickeln. Zusätzlich wurde sich zum Ziel gesetzt, eine neue Version des Robotinos, den Robotino 2.0, am Ablauf der Produktionsstraße zu beteiligen.

Um dieses Gesamtziel zu erreichen sind gemeinsame Schnittstellen, stetige Kommunikation so wie abgestimmtes Zeitmanagement von großer Bedeutung.

In der vorliegenden Dokumentation wird die Umsetzung der Bahnplanung eingehend erläutert. Zu den Aufgabenbereichen der Bahnplanung gehört die Vorgabe eines Weges für den Robotino von einem Start- zu einem Zielpunkt sowie die Kollisionsvermeidung mit statischen und dynamischen Hindernissen.

2 Aufgabenstellung

Aufgabe der Bahnplanung ist es, den Robotino von einer beliebigen Startposition im bekannten Raum zu einem vorgegebenen Ziel fahren zu lassen. Dabei ist zu beachten, dass es zu keiner Zeit zu einer Kollision mit dynamischen oder statischen Hindernissen kommt.

Für die Aufnahme und Abgabe der Werkstücke, also das Werkstückhandling allgemein, muss sowohl zu der Auftragskoordination, wie auch zu der Regelung eine geeignete und zuverlässige Schnittstelle generiert werden. Dazu gehört auch das Anfahren an die Stationen und das Fehler-Handling.

Die Algorithmen zur Realisierung der Bahnplanung und Kollisionsvermeidung sind in Matlab/Simulink zu implementieren. Der interne Programmablauf ist als Simulink/Stateflow mittels Blockschaltbilder zu realisieren. Die zu erstellende Software wird auf jeden einzelnen Robotino geladen und läuft dort dezentral als xPC-Target.

3 Bahnplanung

Die Navigation, also das effiziente und kollisionsfreie Bewegen im Raum, gehört zu den Hauptaufgaben von autonomen mobilen Robotern. Um, zum Beispiel Produktionsstraßen zukunftsfähig und effizienter zu gestalten, müssen die Robotinos konkreten Aufgaben wie „Fahre zum Zielpunkt XY und nehme das Werkstück auf“ übernehmen und abarbeiten können. Dazu muss der Robotino zu jeder Zeit folgende Fragen beantworten können: „Wo befinde ich mich? Wo muss ich hin? Wie gelange ich dort hin?“(Mohamed Oubbati Einführung in die Robotik)

Anhand dieser Fragen lässt sich die Bahnplanung in drei Bereiche unterteilen:

- **Lokalisierung:** Genau Positionsbestimmung des Robotinos im bekannten Raum
- **Kartenerstellung:** Vom Roboter über Sensordaten erstellte oder durch Softwareimplementierung bekannte Karte des Raumes
- **Trajektoriengenerierung:** Berechnung von Trajektorien vom Startpunkt zum Ziel

3.1 Lokalisierung

Eine Bahnplanung kann nur dann erfolgen, wenn der Roboter seine eigene Position zu jeder Zeit lokalisieren kann. In dem Fall des hier ausgearbeiteten Projektes erfolgt die Lokalisierung über Deckenkameras, die mittels UDP-Kommunikation den Robotinos ihre eigene Position, aber auch die der anderen Robotinos im bekannten Raum senden. Intern wurde über den Raum ein Koordinatensystem gelegt,so dass die genaue Position der Roboter in x- und y-Richtung ausgegeben werden kann. Zu diesem Zweck sind die Robotinos mit ArUco-Markern ausgestattet. Um ein zuverlässiges Gesamtsystem zu erschaffen und die Ausfallwahrscheinlichkeit zu minimieren, erfolgt die Lokalisierung zusätzlich über Positionsdaten des Gewerks 3 „Regelung“. Somit wird sichergestellt, dass bei ungenauen und nicht vorhandenen Kameradaten die Robotinoposition zu jeder Zeit bekannt ist und ein unterbrechungsfreier Ablauf des Prozesses gegeben ist.

3.2 Kartenerstellung

Mit bekannten Karten und vom Roboter durch die Sensorik erstellte Karten helfen beim Wegfinden. Durch eine vorab erstellte Karte können Hindernisse und Sackgassen implementiert werden, die von dem Transportroboter gemieden werden müssen. Kennt der Robotino seine Umgebung durch eine Karte kann über Selbstlokalisierung ein optimaler Pfad generiert werden. Im Fall des vorliegenden Projektes muss der Roboter nicht durch willkürliches Fahren im Raum vorab eine Karte von unbefahrbaren Punkten sammeln. Den Robotinos sind die festen Hindernisse, das heißt Lager, Lade- und Werkstationen, auf Grund der implementierten Programmierung von vornherein bekannt.

3.3 Trajektoriengenergierung

Durch die Bahnplanung können kollisionsfreie Trajektorien von einem Start- zu einem Zielpunkt generiert werden. Als Trajektorie wird in der Physik der Bewergungsverlauf des Roboters als Kurve im Raum bezeichnet, also die Zustandsänderung relativ zum Koordinatensystem über die Zeit. Es werden Solltrajektorien berechnet, die dem Gewerk 3 „Regelung“ über eine definierte Schnittstelle übergeben werden. Über diese Trajektorie wird der Robotino zum Ziel geführt.

Die Literatur bietet viele verschiedene Ansätze zur Berechnung des bestmöglichen Pfades und hängt von der projektspezifischen Aufgabenstellung und Definition von „bester Pfad“ ab. „Bester Weg“ wird im Allgemeinen mit kürzester Distanz assoziiert. Es kann aber auch andere Faktoren beinhalten, die angeben wie „teuer“ ein Weg ist. Ist der Untergrund zum Beispiel schlecht befahrbar, so könnte es schneller sein einen Umweg zu fahren. Auch könnten kinematische und dynamische Eigenschaften des Roboters in die Definition mit eingehen, so dass Pfade vermieden werden, die Kurven beinhalten, die der Roboter nicht fahren kann. Auch Kriterien wie Energieeffizienz, Schnelligkeit und größtmöglicher Abstand zu Hindernissen haben je nach Aufgabenstellung eine andere Gewichtung bezüglich des Bahnplanungsansatzes.

Beispielsweise könnte bei der Aufgabenstellung „Finde den schnellsten Weg zum Ziel“ein anderer Berechnungsansatz der Bahnplanung verfolgt werden, als bei „Finde den energieeffizientesten Weg zum Ziel“. Weitere Kriterien zum Auswählen eines Bahnplanungsansatzes könnten sein, die kürzeste Strecke zu finden oder Strecken nach Vorhersagbarkeit des Weges zu bewerten. Allgemein wird in der Bahnplanung zwischen globaler und lokaler Bahnplanung unterschieden.

Im Nachfolgenden werden unterschiedliche Bahnplanungsansätze betrachtet und bewertet.

3.3.1 Globale Bahnplanung

Bei der globalen Bahnplanung, in der englischsprachigen Literatur auch bekannt als „Map-Based Planing“ müssen dem Robotino der Raum und die sich darin befindlichen statischen Hindernisse und Sackgassen vollständig bekannt sein, um diese in jedem Fall zu vermeiden. Im Folgenden werden zwei Möglichkeiten der globalen Bahnplanung beschrieben:

- **Sichtbarkeits-Methode mit A*-Algorithmus**
- **Occupancy Grid mit D*-Algorithmus**

Sichtbarbeitsgraph Methode mit A*

Der Sichtbarkeitsgraph (engl. Visible graphs) ist eine Methode um den kürzesten Pfad zwischen zwei Punkten zu finden. Diese Methode setzt voraus, dass vorab Start- und Zielpunkt, so wie Eckpunkte der statischen Hindernisse klar definiert und bekannt sind. Die Eckpunkte der Hindernisse werden dann durch eine Kante verbunden, wenn eine gerade Verbindungslinie gezogen werden kann ohne andere Hindernisse zu schneiden. Dadurch werden Eckpunkte zu Knotenpunkten. Die Hindernisse mit den Verbindungslinien ist in Abbildung 3.1 dargestellt.

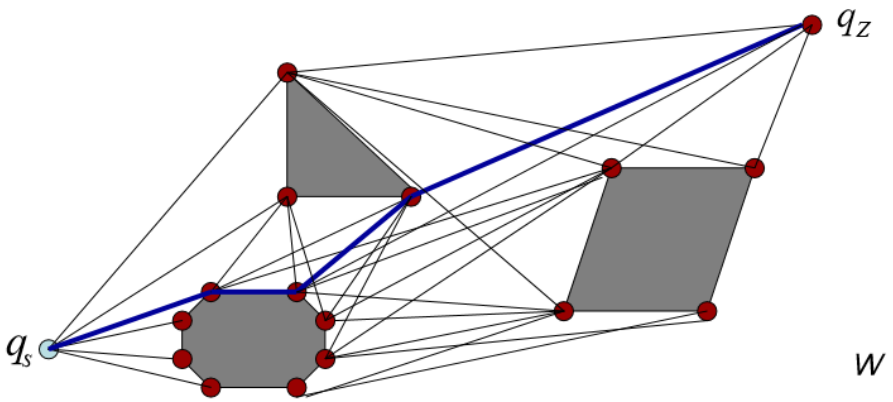

Abbildung 3.1: Beispiel Sichtbarkeitsgraph, Graue Felder sind Hindernisse, blaue Linie ist optimaler Pfad

Durch Einsatz der A*-Methode kann so der optimale Pfad bestimmt werden. Die Kanten zwischen zwei Knotenpunkten werden von dem Algorithmus je nach Anforderung, beispielsweise kürzester Weg, gewichtet. Je größer diese Gewichtung ist, desto weiter Weg befindet sich der Knotenpunkt am anderen Ende der Kante. Die direkte Entfernung eines Knotenpunktes zum Zielpunkt wird geschätzt, so dass die Knoten ebenfalls eine Gewichtung bekommen.

Vom Zielpunkt aus werden nun alle Knoten betrachtet, die über eine Kante erreicht werden können. Der aus der Betrachtung hervorgehende Knotenpunkt, der die geringste direkte Entfernung zum Ziel aufweist, wird als nächstes untersucht. Die anderen werden gespeichert und im weiteren Verlauf mit Konten hinsichtlich ihrer Gewichtung verglichen. Punkte, die bereits betrachtet wurden, werden als „untersucht “ markiert. Dieses Verfahren wiederholt sich so lange bis der optimale Pfad gefunden wurde. Dieser ist in der obenstehenden Abbildung als dicke blaue Linie dargestellt.

Ein Vorteil dieser Methode ist, dass immer der optimale Pfad vom Start zum Ziel gefunden wird. Nachteile sind jedoch, dass der Pfad immer direkt an den Ecken und Kanten der Hindernisse entlang führt. Außerdem kann bei vielen Hindernissen, bzw. Hindernisse mit vielen Ecken und Kanten, ein sehr großer Graph entstehen, der viel Rechenzeit in Anspruch nimmt.

Occupancy Grid mit D*-Algorithmus

Wie der Name „Occupancy Grid“ vermuten lässt, wird der Raum in ein Gitternetz unterteilt. Jede entstehende Zelle kann mit einer „1“ für „belegt“ und „0“ für „frei“ versehen werden. Dadurch entsteht eine Umgebungskarte in dem Hindernisse und freier Raum zum Fahren klar definiert sind. In der Abbildung 3.2 ist ein Beispiel des Occupancy Grids dargestellt. Dabei sind anstatt Zahlen die freien Bereiche weiß und die besetzten Bereiche rot dargestellt. Der Übersichtlichkeit halber ist ein größeres Gitternetz dargestellt, als eigentlich unterteilt.

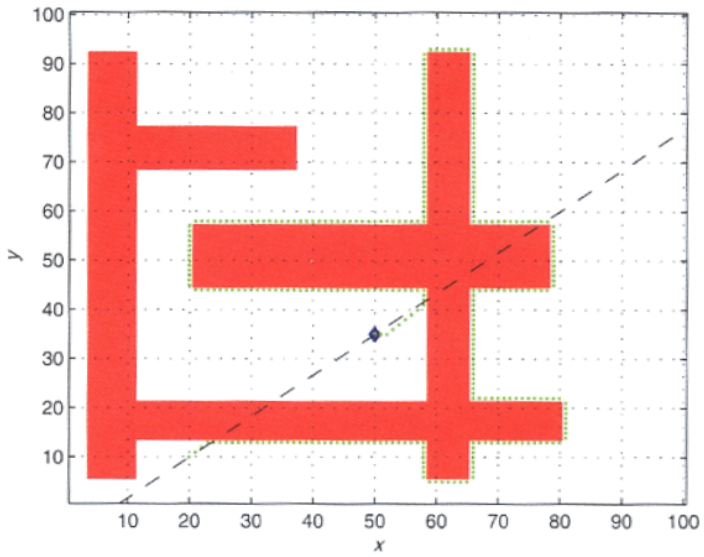

Abbildung 3.2: Beispiel Occupancy Grid, Rot-Hindernisse, Weiß-befahrbarer Raum

Der D*-Algorithmus verändert die Bewertung der einzelnen Zellen mit einem neuen Wert, der die „Kosten“der Zelle repräsentiert. Zellen, die zu einem Hindernis gehören werden mit „∞“gewichtet. Je höher die Gewichtung ist, desto weiter weg befindet sich das Ziel. Mit diesem Algorithmus wird immer der optimale Pfad gefunden. Je nach Aufgabenstellung kann diese Gewichtung zum Beispiel Distanz oder Zeit bedeuten. In der nachstehenden Abbildung 3.3 ist der geplante Weg von dem Startpunkt zum Ziel mittels grün gepunkteter Linie dargestellt. Die Hindernisse sind rot markiert. Die Intensität des Grautons im Hintergrund gibt in diesem Fall die Entfernung zum Ziel an.

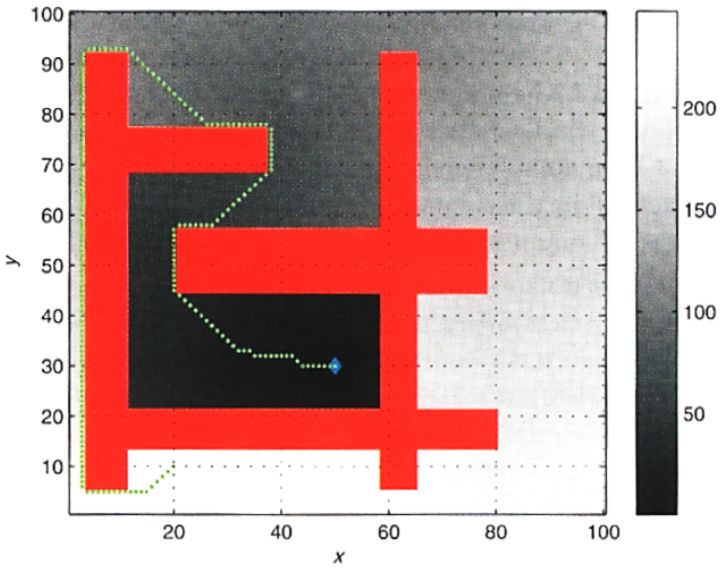

Abbildung 3.3: Beispiel Occupancy Grid mit D*, Rot-Hindernisse, Intensität des Graus im Hintergrunds repräsentiert die Entfernung zum Ziel; blauer Punkt entspricht Ziel

Ein Vorteil dieser Methode ist, dass der Weg schrittweise neu geplant werden kann. Sollte der Raum anders sein, als zuvor in dem Gitternetz eingetragen, eine Zelle beispielsweise teurer als geplant, so kann stufenweise ein besserer Pfad gefunden werden. Die Berechnung des neuen Weges erfordert nicht so viel Rechenleistung wie eine komplett neue Wegplanung, da nur die Zellen in direkter Umgebung betrachtet werden. Die Anfangsberechnung ist hingegen sehr rechenintensiv. Zudem benötigt das Occupancy Grid bei hoher Auflösung viel Speicherplatz.

3.3.2 Lokale Bahnplanung

Die lokale Bahnplanung betrachtet nur das direkte Umfeld des Roboters. Lokale Bahnplanungsalgotirthmen planen den Weg zum Ziel weniger vorausschauend. Daher benötigen sie in der Regel weniger Rechenzeit und sind somit schneller als globale Bahnplanungsmethoden. Mit den aktuellen Umgebungsdaten und Sensorinformationen wird ein lokales Navigationsziel angestrebt. Ziel ist hierbei die reaktive Kollisionsvermeidung. Im Folgenden werden zwei Möglichkeiten den lokalen Bahnplanung beschrieben:

- **Bug-Algorithmus**
- **Potentialfeld-Methode**

Bug-Algorithmus

Der „Bug-Algorithmus “ ist im Allgemeinen eine reaktive Navigationsmethode und basiert auf dem Prinzip, dass sich der Roboter sich so lange entlang eines Hindernisses bewegt, bis er wieder freie Fahrt in Richtung Ziel hat. Es gibt verschiedene Algorithmen, die diese Methode verfolgen. Dazu gehört der „Bug1 -“, „Bug2-“, „Bug3-“und „DistBug-Algorithmus “. Im Folgenden wird nur der Bug2-Algorithmus näher gehend erläutert.

Unter Verwendung des betrachteten Algorithmus kennt der Roboter zu jeder Zeit den direkten Weg zwischen seiner Startposition und dem Ziel. Hierbei werden mögliche Hindernisse zunächst nicht berücksichtigt. Abgesehen von einem Ziel im globalen Raum, kennt der Roboter nur seine direkte Umgebung. Der direkte Weg wird hier m-Linie genannt. Ein Beispiel ist in der untenstehenden Abbildung 3.4 dargestellt.

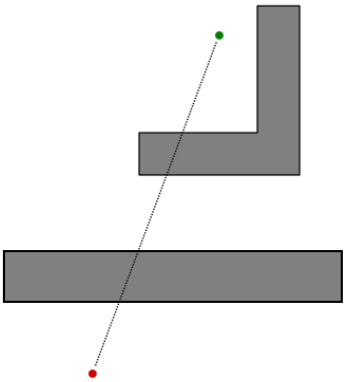

Abbildung 3.4: Bu2-Algorithmus, Grauen Flächen: Hindernisse, Roter Punkt: Startpunkt, Grüner Punkt: Ziel, Verbindungslinie: m-Linie

Der rote Punkt in der Abbildung ist die Startposition und der grüne das Ziel. Die Anweisung des Algorithmus an den Roboter können in drei einfache Befehle unterteilt werden.

- *Fahre auf der m-Linie Richtung Ziel*
- *Ist ein Hindernis im Weg, dann fahre an dessen Kante entlang, bis die m-Linie wieder erreicht wird*
- *Ist die m-Linie wieder erreicht, verlasse das Hindernis und fahre weiter Richtung Ziel*

Wie ein möglicher Weg des Roboters unter dem Bug2-Algorithmus aussehen könnte, ist in Abbildung 3.5 zusehen. Die orangene Linie zeigt dabei den Fahrweg des Roboters

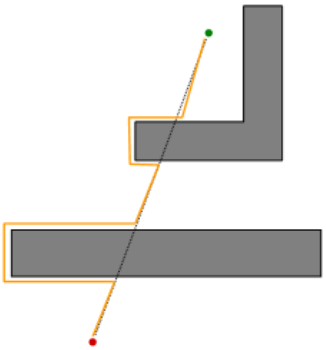

Abbildung 3.5: Bu2-Algorithmus, Fahrweg des Roboters ist in orange dargestellt

Der Bug2-Algorithmus ist ein vergleichsweise einfacher Algorithmus. Das Entlangfahren an den Hindernissen kann nur in eine Richtung, links oder rechts, vorgegeben werden. Somit ist nicht garantiert, dass immer der optimale Pfad gefunden wird. Außerdem kann es zu einer Art „Deadlock“ kommen, wenn der Roboter an einer Kante des Hindernissen entlang fahren muss, die m-Linie jedoch nicht wieder findet. Ein solches Szenario ist in Abbildung 3.6 dargestellt.

Potentialfeld-Methode

Bei der Potentailfeld-Methode ist der Roboter künstlichen Kräften von Zielen und Hindernissen ausgesetzt. Die hohen Potentiale stoßen den Roboter ab, niedrigere Potentiale ziehen ihn an. Jeder Punkt in dem Konfigurationsraum erhält ein Potential. So wird dem Start ein hohes und dem Ziel ein sehr niedriges Potential zugeordnet. Hindernisse bekommen sehr hohe

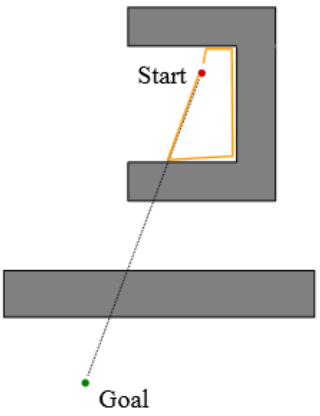

Abbildung 3.6: Bu2-Algorithmus, möglicher Deadlock, Fahrweg des Roboters ist in orange dargestellt

Potentiale. Die Bewegungsrichtung des Roboters kann über die Gradientenberechnung des Potentialfeldes erfolgen, da der Roboter auf seinem Weg vom Start zum Ziel dem negativen Gradienten des globalen Potentialfeldes folgt. In die Berechnung des Gradienten gehen nur die Hindernisse mit ein, die sich in relativer Nähe zum Roboter befinden. In der Abbildung 3.7 ist ein Beispiel eines Potentialfeldes dargestellt. Der rote Teil, auf dem sich die Startposition hat, genau so wie die Hindernisse ein hohes Potential. Das niedrige Potential des Ziels ist in blau dargestellt.

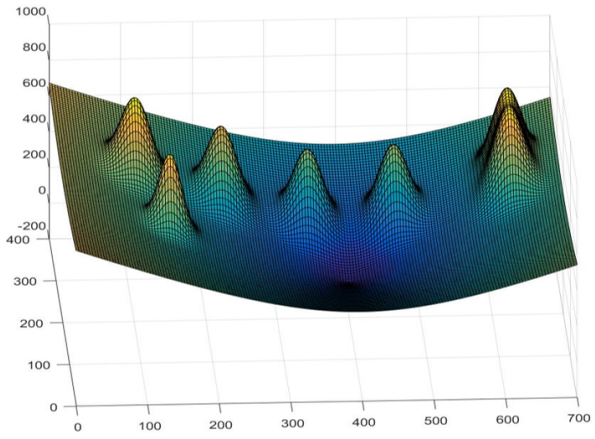

Abbildung 3.7: Beispiel Potentialfeld

Die Vorteile der Potentialfeld-Methode sind sowohl die einfache Implementierung als auch die geringe Rechenzeit. Durch die Echtzeit-Kollisionsvermeidung ist das System sehr dyna-

misch. Es besteht jedoch die Gefahr, dass lokale Minima entstehen, aus denen der Robotino nicht wieder raus kommt.

3.4 Ausgewähltes Konzept

An der zu simulierenden Produktionsstraße sind bis zu fünf Robotinos gleichzeitig beteiligt. Um einen zuverlässigen Produktionsablauf sicherstellen zu können, werden hohe Anforderungen an die Bahnplanungsmethode gestellt. Die Methode muss dynamisch und zuverlässig sein und sollte geringen Rechenaufwand benötigen. Die soeben erläuterten Methoden sind in der Tabelle 3.1, reduziert auf die wichtigsten Eigenschaften, übersichtliche dargestellt.

Tabelle 3.1: Bahnplanungsmethoden nach ihrer Dynamik, Sicherheit, Rechenzeit und Zuverlässigkeit bewertet

Methode	Dynamik	Sicherheit	Rechenzeit	Zuverlässigkeit
Sichtbarkeitsgraph mit A*	gering	mittel	mittel/hoch	sehr hoch
Occupancy Grid mit D*	mittel	hoch	mittel/hoch	hoch
Bug-Methode	mittel	gering	gering	gering
Potentialfeld-Methode	sehr hoch	mittel	gering	hoch

Bei sehr hoher Dynamik und hoher Zuverlässigkeit benötigt die Potentialfeld-Methode nur wenig Rechenzeit. Damit ist sie von den betrachteten Methoden die beste. Das Potentialfeld hat zudem den Vorteil, dass ein komplett autonomes Gesamtsystem geschaffen werden kann, da es für die Trajektorienberechnung nicht notwendig ist das Ziel und die Fahrtroute der anderen Robotinos zu kennen.

3.4.1 Gesamtkonzept

Im Folgenden wird das Gesamtkonzept mit der Bahnplanungsmethode Potentialfeld kurz erläutert. Genauere Erklärungen mit Auszügen aus dem Originalprogramm wird anschließend erläutert. Der Konfigurationsraum ist in Abbildung 3.8 mit globalem Koordinatensystem, eingezeichneten Stationen, Wartepositionen und Übergabepunkten dargestellt.

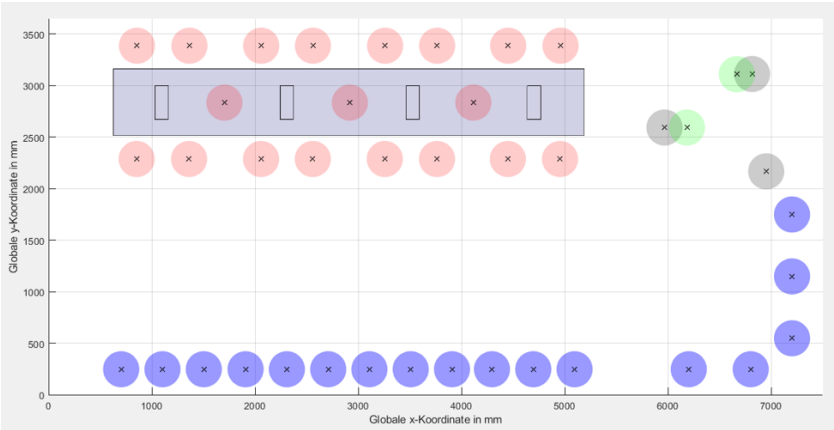

Abbildung 3.8: Konfigurationsraum

Warteporitionen

Jedem Robotino wird eine eigene Warteposition am Rand des Konfigurationsraumes zugeordnet. Diese Warteposition ist zu Beginn des Gesamtsystem die Startposition des jeweiligen Robotinos und immer dann Zielposition, wenn der Robotino keinen Auftrag empfängt. Sollten alle „Fifo-Plätz“ einer Station belegt sein, so dass sich ein Robotino nicht mehr anstellen kann, muss er aus seiner Warteposition auf einen freien Platz warten.

Stationen und Übergabepunkte

Wie in Abbildung 3.8 zu sehen ist, befindet sich vor und hinter jeder Station Übergabepunkte. Hat ein Robotino die Aufgabe zu einer Station zu fahren, fährt er stattdessen nur vor die Station auf den Übergabepunkt. Ab da wird Steuerung des Roboters an das Gewerk3 „Regelung“ übergeben. Sollten mehrere Robotinos die gleiche Station anfahren wollen, so darf der Robotino die Station zuerst befahren, der den Übergabepunkt zuerst erreicht hat. Über die Übergabepunkte wird die Station auch wieder verlassen. Der vordere Übergabepunkt wird immer dann genutzt, wenn sich kein weitere Robotino im Fifo befindet. Über den hinteren Übergabepunkt wird die Station verlassen, wenn ein andere Robotino bereits im Fifo wartet. Das Potentialfeld ist in den Stationen, solange Gewerk3 „Regelung“ den Robotino steuert, nicht aktiv. Aktiviert bzw. ausgeschaltet wird das Potentialfeld zum Zeitpunkt der Übergabe. Sollte ein Robotino in eine Station gedrückt werden, so verlässt er sie unverzüglich nach hinten raus. In diesem Fall bleibt das Potentialfeld aktiv.

Warteschlange „Fifo“

Ist eine Station, in die ein Robotino fahren soll, bereits belegt, so fährt er zu der zugehörigen Warteschlange, dem sogannten „Fifos“. Die Fifos sind Warteposition für die Stationen, die sich am Rand des Konfigurationsraumes befinden, um während des Wartens den befahrbaren Raum nicht zu blockieren. Der Robotino, der die Warteschlange zuerst betreten hat, darf diese, sobald die Station wieder frei ist, auch als erster wieder verlassen. Erst, wenn dieser die Station erreicht hat, rücken eventuell wartende Robotinos im gleichen Fifo auf.

Ladestation

Die Ladestationen befinden sich an der rechten Seite des Konfigurationsraumes. Eine Ladestation ist für die Robotinos 1.0 und eine weitere für den Robotino 2.0. Auch hier sind Übergabepunkte definiert. Aus platztechnischen Gründen muss der Robotino 2.0 von hinten an die Ladestation fahren. Um dieses möglichst effizient zu realisieren ist der Übergabepunkt für diese Ladestation vergleichsweise weit im eigentlichen Raum. Ab diesem Punkt wird an Gewerk3 übergeben, die den Robotino 2.0 vor und in die Ladestation fahren lässt.

Potentialfeld

Hindernisse im Potentialfeld werden mit hohen Potentialen versehen. Zu diesen Hindernissen gehören hier die Stationen und die Robotinos. In Abbildung 3.9 ist das gesamte Potentialfeld dargestellt, das auf jeden Robotino geladen wird.

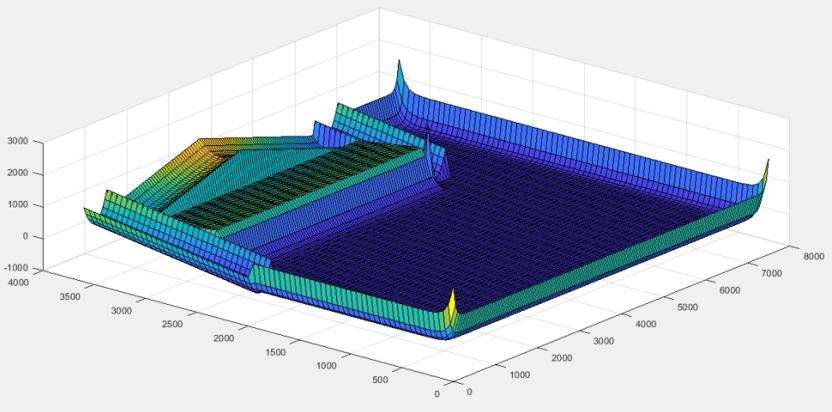

Abbildung 3.9: Potentialfeld des Konfigurationsraumes eingezäunt in hohe Potentiale zur eindeutigen Begrenzung des Raumes, Stationen als ein hohes Potential dargestellt, Rampen und Fahrrinnen zum leiten der Robotinos,

Über eine Begrenzung durch hohes Potential wird der Raum, in dem sich die Robotinos bewegen dürfen eindeutig definiert. Die Stationen sind als ein gesamter Block mit hohem Potential dargestellt, da der Robotino zu keiner Zeit zwischen oder an eine Station fahren soll, wenn nicht zuvor die Übergabe an Gewerk3 erfolgt ist. Zur Vermeidung von kritischen Situartionen oder lokalen Minima zwischen Stationen und Wand sind Rampen und Fahrrinnen implementiert worden, die je nach Station den Roboter nach links oder rechts in eine Fahrrinne und von dort in den frei befahrbaren Raum führen.

4 Konzept

In diesem Kapitel wird das in diesem Bericht genutzte Konzept beschrieben. Das Grundkonzept bestehet darin, dass der zu befahrene Bereich in einen Fertigungsbereich und einen Transportbereich unterteilt wird. Die Grenzen der Bereiche sind in Abbildung 4.1 dargestellt. Im Transportbereich, in der Abbildung **??** nicht eingefÃrbt, wird die Regelung des Robotinos Ã¼ber die Potentialfeldmethode realisiert. Das dabei genutzte Potentialfeld wird unter Kapitel 4.1 nÃher erlÃutert. Im Fertigungsbereich wird die Regelung von der Bahnregelungsgruppe Ã¼bernommen. Um zwischen den Bereichen zu wechseln, werden Ãbergabepunkte definiert, an denen der Bereichswechsel sicher ausgefÃ¼hrt werden kann. Diese Ãbergabepunkte sind in der Abbildung als rote Kreise dargestellt. Dazu wird eine Kommunikation zwischen den Einzelgruppen Ã¼ber eine Schnittstelle definiert. Diese Schnittstelle wird in Kapitel **??** definiert. Desweiteren wird das Konzept zum Vermeiden von Kollisionen in Kapitel **??** nÃher beschrieben. Dabei wird auf verschiedene Hindernistypen eingegangen. Um das Konzept abzuschlieÃen wird in Kapitel **??** der Ablauf des Programmes dargestellt.

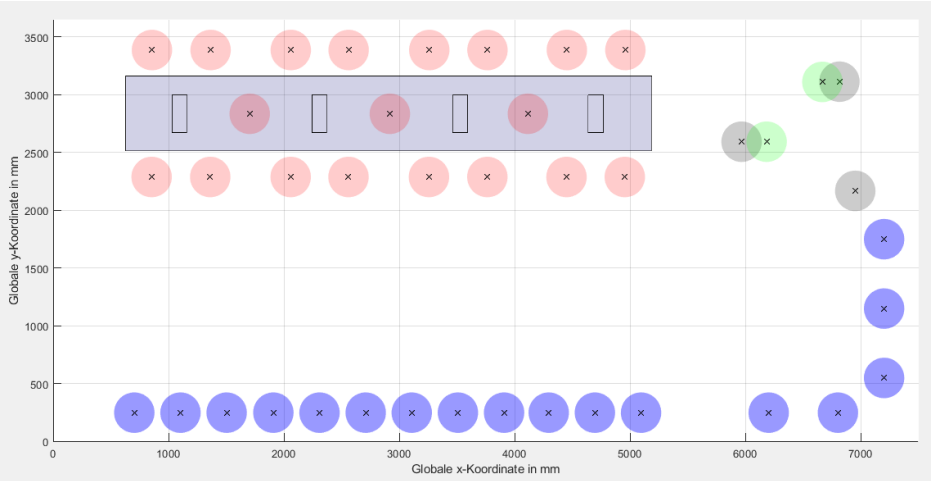

Abbildung 4.1: Bereichseinteilung

4.1 Potentialfeld zur Bahngenerierung und Kollisionsvermeidung

In diesem Abschnitt wird beschrieben, wie das Potentialfeld aufgebaut ist, dass zur Generierung des Geschwindigkeitsvektors genutzt wird.

4.1.1 Roboter Umwelt

Um die statische Umwelt des Robotinos im Potentialfeld darzustellen, wird der zu befahrende Bereich in Zonen unterteilt. Die gewÃhlten Zonen sind in Abbildung 4.2 farbig dargestellt. In Tabelle 4.1.1 ist die Zurordnung der Zonen zur Abbildung 4.2 beschrieben. Dabei wird eine Hauptfahrzone (Zone 0) definiert, in der mit einem virtuellen Zaun der zu befahrene Bereich abgegrenzt wird. Dieser Zaun ist notwendig, damit die Zone nicht unkontrolliert verlassen wird, dazu werden e-Funktionen genutzt, die wie in den Funktionen **??** bis **??** definiert sind. Aufgrund der Form der Zone mÃ¼ssen dabei drei Funktionen definiert werden, die je nach Position des Robotinos ausgefÃ¼hrt werden. Das Verlassen der Zone 0 erfolgt durch die Ãbergabe an den Fertigungsbereich, welches in Kapitel **??** nÃher erlÃutert wird.
Zone 1 bis 3 dienen zur RÃ¼ckfÃ¼hrung der Robotinos zur Zone 0, dazu wird ein Potentialfeld genutzt, dass uns ermÃ¶glicht den Robotino in eine geziehlte Richtung zu lenken. Die dazu genutzten Funktionen sind unter 4.1.1 bis **??** definiert. Diese Potentialfelder nutzen in Fahrrichtung eine Geradengleichung mit definierter Steigung und senkrecht dazu eine Parabelform, um den Robotino auf Kurs zu halten. Wenn sich der Robotino in Zone 4 befindet, wird dieser Ã¼ber eine Geradengleichung noch hinten gefÃ¼hrt. Die dbei genutzte Gleichung ist als **??** gekennzeichnet. ZusÃtzlich wird in Zone 4 fÃ¼r jede Station eine gauÃsche radiale Basisfunktion, welche unter Kapitel **??** nÃher erlÃutert, genutzt, damit keine Kollision mit den Stationen auftreten. Diese Zone wird nur im Fehlerfall oder beim Start des Robotinos betreten, da in dieser Zone der Fertigungsbereich aktiv ist. Die Zone 5 dient dazu den Robotino geziehlt in Richtung der mitte der Zone 0 zu fÃ¼hren. Dazu wird wie in Zone 1 bis 3 eine Parabel- mit einer Geradengleichung genutzt. Die dabei genutzte Funktion ist als **??** definiert. Dabei ist zu beachten, dass vorgesehen ist, dass Zone 5 nur aktiv ist, wenn sie aus Zone 1 betreten wird. Dadurch wird die AnfahrmÃ¶glichkeit aus der Zone 0 zur Ladestation gewÃhrleistet.
Je nach Zone gilt dabei ein eigenes Potentialfeld, welche kombiniert das Gesamtpotentialfeld ergebenen. Die einzelnen Zonen sind in Abbildung 4.3 dargestellt. Das gesamt Potentialfeld ist in Abbildung 4.4 dargestellt, wobei zur Ãbersichtlichkeit die Stations und Ladestationspotentiale nicht dargestellt werden. Da die Zone 5 nur durchlaufen wird, wenn sie aus Zone 1 betreten wird, wird in Abbildung 4.5 dargestellt, wie das Potentialfeld in diesem Fall aussieht.

Zone	Farbe	Beschreibung
0	ohne Farbe	Hauptfahrzone
1	GrÃ¼n	FÃ¼hrung des Robotinos nach Zone 5
2	Blau	FÃ¼hrung des Robotinos nach Zone 3
3	Violet	FÃ¼hrung des Robotinos nach Zone 0
4	Rot	FÃ¼hrung des Robotinos nach Zone 1 und 2
5	Cyan	FÃ¼hrung des Robotinos nach Zone 0

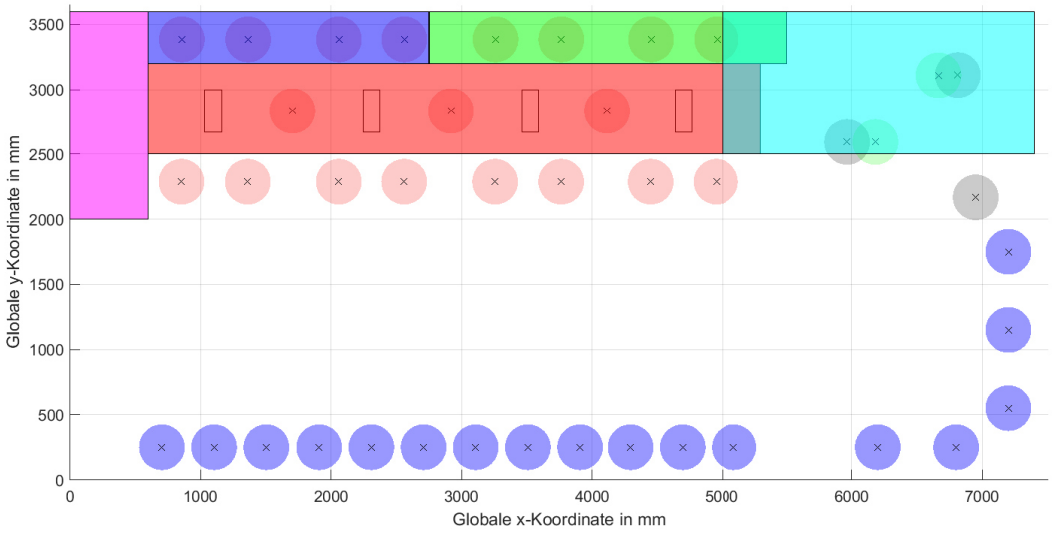

Abbildung 4.2: Potentialfeldzonen

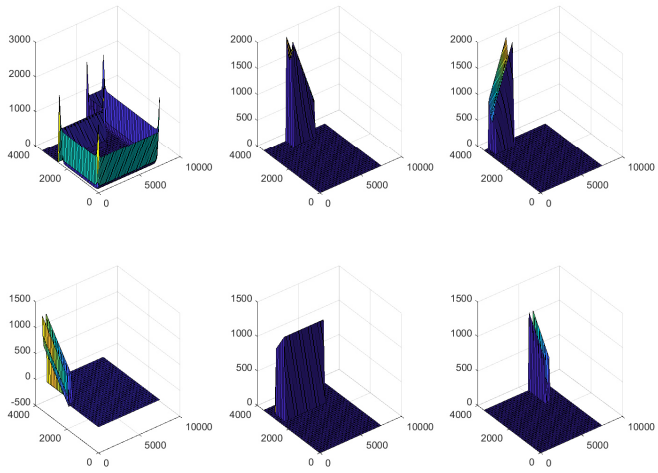

Abbildung 4.3: Potentialfeld der einzel Zonen

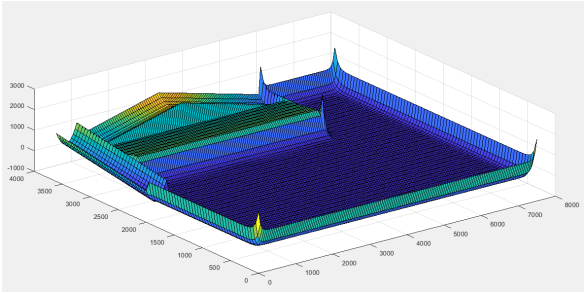

Abbildung 4.4: Potentialfeld der Roboter Umwelt ohne RausfÃ$\frac{1}{4}$hrung

$$H_{0(x=0..5500,y=0..2500)} = K_{exp} \cdot (e^{(-\frac{x}{\sigma})} + e^{(\frac{x-7400}{\sigma})} + e^{(-\frac{y}{\sigma})} + e^{(\frac{y-2500}{\sigma})})$$
$$H_{0(x=5500..7400,y=0..2500)} = K_{exp} \cdot (e^{(-\frac{(x-5500)-(y-2500)}{\sigma})} + e^{(\frac{x-7400}{\sigma})} + e^{(-\frac{y}{\sigma})})$$
$$H_{0(x=5500..7400,y=2500..3600)} = K_{exp} \cdot (e^{(-\frac{(x-5500)}{\sigma})} + e^{(\frac{(y-3600)}{\sigma})} + e^{(\frac{x-7400}{\sigma})} + e^{(-\frac{y}{\sigma})})$$
$$H_1 = -K_{gerade} \cdot x + K_{parabel} \cdot \frac{(y-3500)^2}{B}$$
$$H_2 = K_{gerade} \cdot x + K_{parabel} \cdot \frac{(y-3500)^2}{B}$$
$$H_3 = K_{parabel} \cdot \frac{(x-300)^2}{B} + K_{gerade} \cdot y$$
$$H_4 = -K_{gerade} \cdot y$$
$$H_5 = K_{parabel} \cdot \frac{(x-5500)^2}{B} - K_{gerade} \cdot y$$

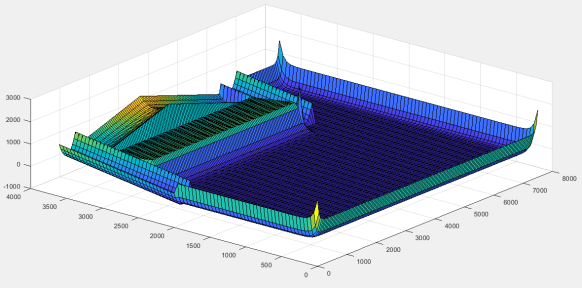

Abbildung 4.5: Potentialfeld der Roboter Umwelt mit RausfÃ¼hrung

Abbildung 4.6: Potentialfeld der gauÃschen radialen Basisfunktion

4.1.2 Robotino,Sationen und IR-Hindernisse

Die Robotinos, die Stationen und die IR-Hindernisse werden im Potentialfeld als gauÃsche radiale Basisfunktion betrachtet. Die genutzte Funktion ist unter **??** definiert. Da sich die gauÃsche radiale Basisfunktion aufgrund der genutzten e-Funktion gut ableiten lÃsst, wird diese Funktion genutzt, um ein abstoÃendes Potential zu generieren. Das Potentialfeld der gauÃschen radialen Basisfunktion ist in Abbildung 4.6 dargestellt. Dabei wird fÃ¼r die Stationen eine Konstante Position eingestellt. Die Position des Robotinos und der IR-Hindernisse werden als flexibel betrachtet, da sie wÃhrend der laufzeit bestimmt werden mÃ¼seen..

$$H_{Gau} = K \cdot e^{-\frac{(x-x_{pos})^2+(y-y_{pos})^2}{2\sigma^2}}$$

4.1.3 Ziel

Das Ziel im Potentialfeld wird als Kegel mit konstantem Faktor realisiert. Dazu wird die in Gleichung **??** dargestellte Funktion genutzt. Um durch TrÃgheit verursachte Schwingungen

Abbildung 4.7: Potentialfeld Ziel

um den Zielpunkt zu vermeiden, wird eine AbstandsabhÃngiger Faktor als Filterung nachgeschaltet. Dabei wird der Abstand Ã$\frac{1}{4}$ber den Pythagoras bestimmt. Der dabei genutzte Faktor wird, wie in den Funktion **??** und **??** beschrieben, bestimmt. Das bei der Multiplikation von $K_{Ziel} \cdot H_{Ziel}$ entstandene Potentialfeld ist in Abbildung 4.7 dargestellt.

$$H_{Ziel} = K \cdot \sqrt{(x - x_{Ziel})^2 + (y - y_{Ziel})^2}$$
$$K_{Ziel(Abstand_{ist} =< Abstand_{max}} = \frac{Abstand_{ist}}{Abstand_{max}}$$
$$K_{Ziel(Abstand_{ist} > Abstand_{max}} = 1$$

5 Simulation und Testplanung

Um das Potentialfeld auch ohne Robotinos zu testen, wird zu Beginn des Projektes mit einer vereinfachten Simulation das Potentialfeld auf Ihre FunktionsfÃhigkeit geprÃ¼ft. Bei dieser Simulation wird der Robotino als enfache I-Strecke betrachtet. Dabei wird der Ausgang der Strecke auf den Istpositionsinput gegeben und der berechnete Geschwindigkeitsvektor auf die Strecke gegeben. Desweiteren werden auf die restlichen Inputs Konstanten gegeben und die Ausgangsdaten in einer Logdatei abgespeichert. Da nach wenigen Versuchen erkennbar ist, dass das Potentialfeld seine Funktion erfÃ¼llt, wird im folgenden mit dem Robotinos direkt getestet. Dazu wird zu Beginn der Fertigungsbereich nicht betrachtet und nur ein Festprogrammiertes Ziel angefahren. Um flexibel Ziele anfahren zu kÃ¶nnen und die Schnittstelle mit der Fertigungsplanung zu teseten wird ein UDP-Dummy in Simulink erstellt mit dem AuftrÃge an den Robotino per UDP in Simulink gesendet werden kÃ¶nnen. Dadurch kÃ¶nnen mehrere Ziele hintereinander Angefahren werden. Im nÃchsten Schritt wird die Schnittstelle mit der Bahnregelung gestestet dazu werden beide Simulinkmodelle kombiniert und auf den Robotino geladen. Dadurch kÃ¶nnen Schnittstellenprobleme zwischen der Bahnregeleung und der Bahnplanung frÃ¼hzeitig behoben werden. Da sich im laufe des Projektes gezeigt hat, dass die Fertigungsplanung den Gesamtsystemtesttermin nicht einhalten kann, wird der UDP-Dummy in zusammenarbeit mit der Bahnregelung dahingehend erweitert, zufallsgenerierte AuftrÃge zu senden. Im letzten Schritt wird der UDP-Dummy als Fertigungsplanungsersatz erweitert indem sich die Positionen der WerkstÃ¼cke gemerkt wird und die zufallsgenerierten AuftrÃge auf ausfÃ¼hrbare AuftrÃge gefiltert wird. Zum Ende des Projektes wird zusÃtzlich eine Simulation, wie zu Beginn, mit der Erweiterung auf 5 Robotinos ausgefÃ¼hrt um geringe Ãnderungen im Programm aufzuzeichnen, da die Aufzeichnung von Daten von mehreren Robotinos unter Simulinkrealtime einige Probleme verursacht. Unter Kapitel **??** wird eine Simulation des AusweichmanÃ¶vers dargestellt.

5.1 Simulation des AusweichmanÃ¶vers

In diesem Abschnitt wird die Simulation des AusweichmanÃ¶vers dargestellt. Dazu werden zwei Simulationen miteinander Verglichen. In der ersten Simulation, welche in Abbildung 5.1 dargestellt ist, werden zwei Robotinos betrachtet. Robotino 1 in Rot dargestellt fÃhrt von der Position [500 1000] zur Station 7 welche sich ganz rechts befindet. Robotino 2 in blau

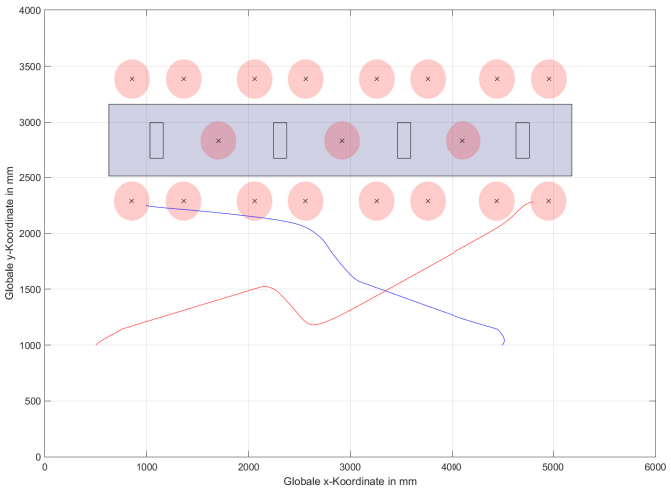

Abbildung 5.1: Simulation mit Ausweichfunktion

dargestellt fÃhrt von der Position [4500 1000] zur Station 0 welche sich ganz links befindet. Wie in der Abbildung zu sehen ist, umfahren sich die Robotinos rechts herum.

In der zweiten Simulation, welche in Abbildung 5.2 dargestellt ist, werden die gleichen Robotinos mit der gleichen Start und Zielposition betrachtet. In dieser Simulation wird jedeglich die Ausweichfunktion auf Null gesetzt. Wie in der Abbildung zu erkennen ist, kommt es zu einem AbstoÃverhalten, wodurch die Robotinos stark ausgebremst werden. Wie in den Abbildungen zu erkennen ist, hat die entwickelte Ausweichfunktion eine Verbesserung im Ausweichverhalten verursacht.

5.2 Simulation Wartebereiche

5.3 Geschwindigkeit bestimmen

Um die Sollgeschwindigkeit des Robotinos zu bestimmen, wird das Subsystem Vektorberechnung genutzt. Dieses Subsystem ist wie in Abbildung 5.3 dargestellt, aufgebaut. Das Subsystem hat dabei den in Abbildung 5.4 gezeigten Ablauf. Dabei ist zu erkennen, dass zuerst die IR Daten erfasst werden und die fahrenden Robotinos erkannt werden. Die IR Datenerfassung wird unter Kapitel **??** nÃher erlÃutert. Wie die fahrenden Robotinos erkannt werden, wird in Kapitel **??** beschrieben. Anhand der IR Daten und der Informationen Ã$\frac{1}{4}$ber die fahrenden Robotinos wird dann der Geschwindigkeitsvektor bestimmt. Die dabei genutzte

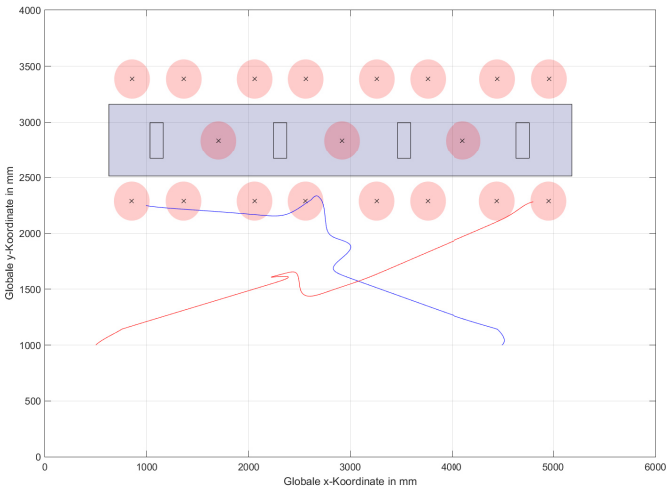

Abbildung 5.2: Simulation ohne Ausweichfunktion

Funktion wird unter Kapitel **??** beschrieben. Anhand des bestimmten Geschwindigkeitsvektors wird dann eine Ausweichrichtung bestimmt, wenn ein anderer Robotino die Route kreuzt. die dazu genutzte Funktion ist in Kapitel **??** nÃher beschrieben. Im nÃchsten Schritt wird dann das AusweichmanÃ¶ver ausgefÃ¼hrt. Die Implementierung des AusweichmanÃ¶vers wird in Kapiitel **??** beschrieben.

5.3.1 IR Daten erfassen

In der IR Daten erfassen Funktion werden zuerst die IR Daten aus den RobotinoDaten gefiltert und auf NaN gesetzt, wenn der gemessene Abstand grÃ¶Ãer als 150 mm ist. Bei einem Abstand kleiner gleich 150 mm wird die Position der Hindernisse in globalen Koordinaten bestimmt und ausgegeben.

5.3.2 Fahrende Robotinos erkennen

Um zu erkennen ob die anderen Robotinos fahren, werden die Postionen der Robotinos mit den Postionen vor einer Sekunde verglichen. Wenn die Robotinos mehr als 10 mm in der Sekunde zurÃ¼ckgelegt haben, wird der Robotino als fahrend gesetzt.

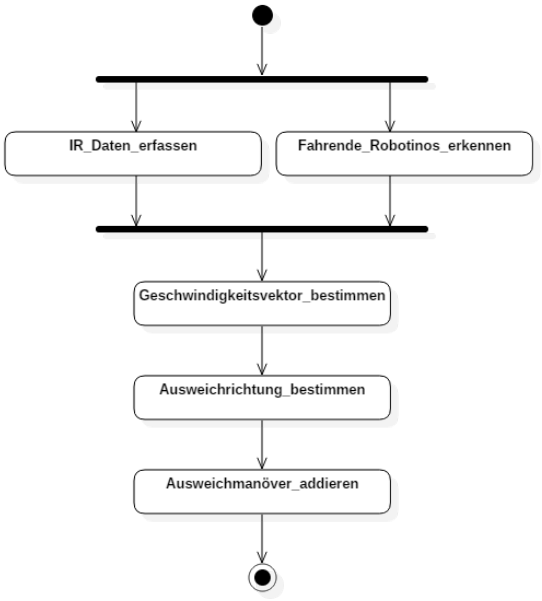

Abbildung 5.3: Subsystem Vektorberechnung

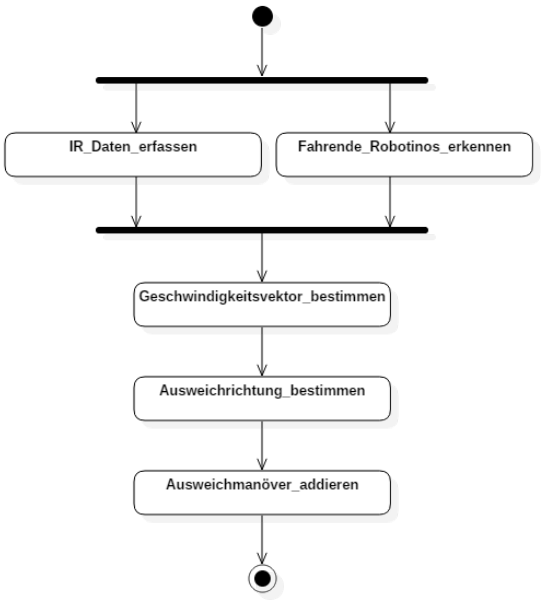

Abbildung 5.4: Ablaufplan des Subsystems Vektorberechnung

5.3.3 Geschwindigkeitsvektor bestimmen

5.3.4 Ausweichrichtung bestimmen

5.3.5 AusweichmanÃ¶ver addieren

6 Robotino 2.0

In diesem Kapitel wird beschrieben welche Anpassungen am Programm vorgenommen werden mÃ¼ssen, um das Programm von den alten Robotinosauf dem neuen Robotino lauffÃ¤hig zu bekommen. Um das Programm auf den Robotino 2.0 anzupassen wird eng mit der Bahnregelung und Gewerk4 zusammengearbeitet. Die hauptÃnderung des Programmes besteht darin den Simulinkblock zur UDP-Kommunikation, durch ein Output zu ersetzen, da die UDP-Kommunikation auf den UDP-Kontroler des Robotinos gemappt werden muss. Desweiteren werden einige Datentypen konvertiert.

7 Validierung

Literaturverzeichnis

[Babic 2003] BABIC, Alexander: *Entwicklung einer profilverarbeitenden ubiquitären Anwendung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2003. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/babic.pdf

[Bigus und Bigus 2001] BIGUS, Joseph ; BIGUS, Jennifer: *Intelligente Agenten mit Java programmieren*. Addison-Wesley, 2001. – ISBN 3-8273-1841-6

[Böhm 2002] BÖHM, Oliver: *Java Software Engineering unter Linux*. SuSE Press, 2002. – ISBN 3-935922-53-1

[Dawson 2003] DAWSON, Christian W.: *Computerprojekte im Klartext*. Pearson Studium, 2003. – ISBN 3-8273-7067-1

[DeMarco und Lister 1999] DEMARCO, Tom ; LISTER, Tomothy: *Wien wartet auf Dich - Der Faktor Mensch im DV-Management*. Carl Hanser Verlag München Wien, 1999. – ISBN 3-446-21277-9

[Drosdowski u. a. 1997] DROSDOWSKI, Prof. Dr. Dr. h.c. Günther (Hrsg.) ; SCHOLZE-STUBENRECHT, Dr. W. (Hrsg.) ; WERMKE, Dr. M. (Hrsg.): *Das Fremdörterbuch*. Duden, 1997. – ISBN 3-411-04056-4

[Ferber 2001] FERBER, Jacques: *Multiagentensysteme*. Addison-Wesley, 2001. – ISBN 3-8273-1679-0

[Goossens u. a. 2000] GOOSSENS, Michel ; MITTELBACH, Frank ; SAMARIN, Alexander: *Der LaTeX -Begleiter*. Addison-Wesley, 2000. – ISBN 3-8273-1689-8

[Günther 2002] GÜNTHER, Karsten: *LaTeX GE-PACKT*. mitp, 2002. – ISBN 3-8266-0785-6

[Heinsohn und Socher-Ambrosius 1999] HEINSOHN, Jochen ; SOCHER-AMBROSIUS, Rolf: *Wissensverarbeitung*. Spektrum Akademischer Verlag, 1999. – ISBN 3-8274-0308-1

[Hewlett-Packard 2004] HEWLETT-PACKARD: *Cooltown (Overview)*. 2004. – URL http://www.cooltown.com/cooltown/

[Hörauf 2001] HÖRAUF, Julia: *Hewlett-Packard Cooltown Project*. Universität Karlsruhe - Institut für Telematik, Seminararbeit zum Seminar Ubiquitäre Systeme. 2001. – URL http://www.informatik.uni-mannheim.de/pi4/lectures/ws0102/UbiquitousComputing/cooltown_arbeit.pdf

[Hunt und Thomas 2003] HUNT, Andrew ; THOMAS, David: *Der Pragmatische Programmierer*. Hanser, 2003. – ISBN 3-446-22309-6

[IDEAlliance 2004] IDEALLIANCE: *CP Exchange*. 2004. – URL http://www.cpexchange.org/standard/

[Kohm und Morawksi 2003] KOHM, Markus ; MORAWKSI, Jens-Uwe: *KOMA-Script*. dante, 2003. – ISBN 3-936427-45-3

[Kollakowski 2004] KOLLAKOWSKI, Malte: Mobile Aktivitäten. In: *Der Entwickler* (2004), April, Nr. 4.04, S. 15–20

[Kruse 2000] KRUSE, Otto: *Keine Angst vor dem leeren Blatt*. campus concret, 2000. – ISBN 3-593-36659-2

[Kühnel 2001] KÜHNEL, Ralf: *Agentenbasierte Softwareentwicklung*. Addison-Wesley, 2001. – ISBN 3-8273-1739-8

[Kurose und Ross 2002] KUROSE, James F. ; ROSS, Keith W.: *Computernetze*. Pearson Studium, 2002. – ISBN 3-8273-7017-5

[Lamport 1995] LAMPORT, Leslie: *Das LaTeX Handbuch*. Addison-Wesley, 1995. – ISBN 3-89319-826-1

[von Lüde u. a. 2004] LÜDE, Rolf von ; MOLDT, Daniel ; VALK, Rüdiger: *Sozionik - Modellierung soziologischer Theorie*. LIT, 2004. – ISBN 3-8258-5980-0

[Luger 2001] LUGER, George F.: *Künstliche Intelligenz*. Pearson Studium, 2001. – ISBN 3-8273-7002-7

[Lüpke 2004] LÜPKE, André: *Entwurf einer Sicherheitsarchitektur für den Einsatz mobiler Endgeräte*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2004. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/luepke.pdf

[Nilsson 1998] NILSSON, Nils J.: *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, 1998. – ISBN 1-55860-535-5

[Poenicke 1988] POENICKE, Klaus: *Wie verfaßt man wissenschaftliche Arbeiten?* Duden, 1988. – ISBN 3-411-02751-7

[Rötzer 1999] RÖTZER, Florian: *Ein neuer Standard soll Kundendaten zusammenführen*. 1999. – URL http://www.heise.de/newsticker/meldung/6970

[Schirru 2004] SCHIRRU, Rafael: *Trust, Reputation, Privacy*. Universität Karlsruhe - Lehrgebiet Datenverwaltungssysteme, Seminararbeit zum Seminar Grundlagen webbasierter Informationssysteme. 2004. – URL http://wwwdvs.informatik.uni-kl.de/courses/seminar/SS2004/rschirrub.pdf

[Schmatz 2004] SCHMATZ, Klaus-Dieter: *Java 2 Micro Edition - Entwicklung mobiler Anwendungen mit CLDC und MIDP*. dpunkt.verlag, 2004. – ISBN 3-89864-271-2

[Streitz und Nixon 2005] STREITZ, Norbert ; NIXON, Paddy: The Disappearing Computer. In: *Communications of the ACM* 48 (2005), März, Nr. 3, S. 33–35

[Vogt 2001] VOGT, Carsten: *Betriebssysteme*. Spektrum Akademischer Verlag, 2001. – ISBN 3-8274-1117-3

[W3C 2004] W3C: *Platform For Privacy Preferences (P3P) Project*. 2004. – URL http://www.w3.org/P3P/

[Wooldridge 2002] WOOLDRIDGE, Michael: *MultiAgent Systems*. Wiley, 2002. – ISBN 0-471-49691-X

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 3. Juni 2018	
Ort, Datum	Unterschrift