Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Martin Mustermann

Entwicklung und Aufbau eines mikrorechnergesteuerten Bestï¿½ckungsautomaten

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 Bestï¿½ckungsautomaten

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 Bestï¿½ckungsautomaten

Stichworte

Steuerung, und viele weitere interessante Stichwort

Kurzzusammenfassung

Diese Arbeit umfasst alles was man mit einem Mikrorechner machen kann und natï¿½rlich 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

Abbildungsverzeichnis

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.

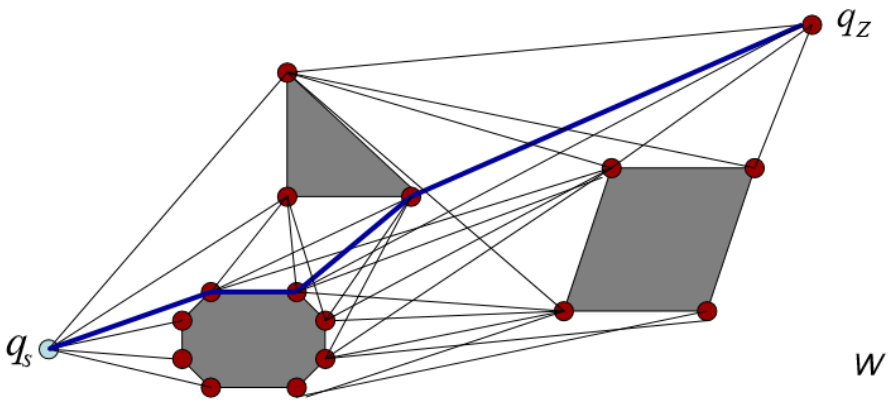

Abbildung 3.1: Beispiel Sichtbarkeitsgraph, Graue Felder sind Hindernisse, blaue Linie ist optimaler Pfad

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.

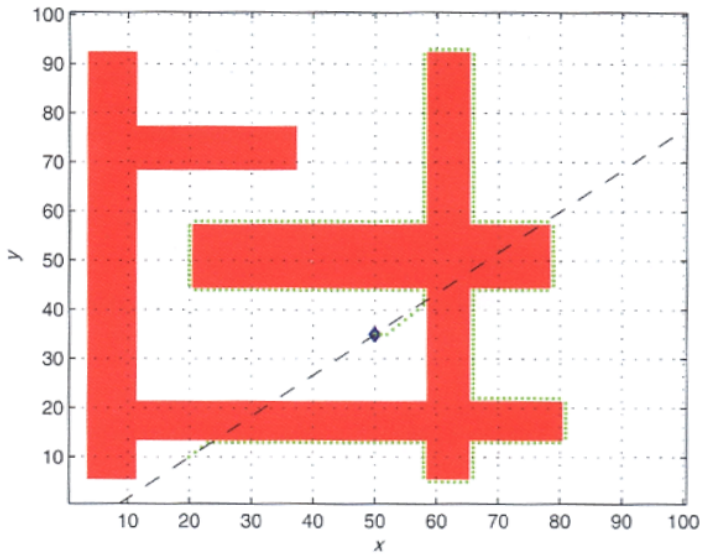

Abbildung 3.2: Beispiel Occupancy Grid, Rot-Hindernisse, Weiß-befahrbarer Raum

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.

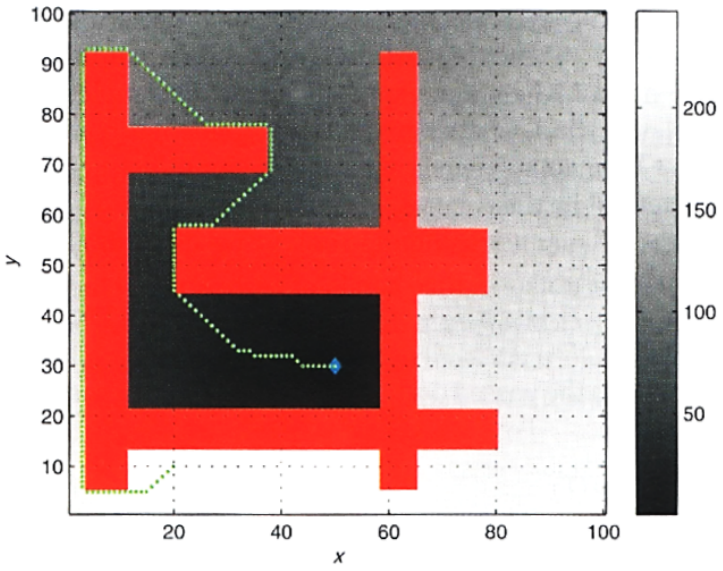

Abbildung 3.3: Beispiel Occupancy Grid mit D*, Rot-Hindernisse, Intensität des Graus im Hintergrunds repräsentiert die Entfernung zum Ziel; blauer Punkt entspricht Ziel

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.

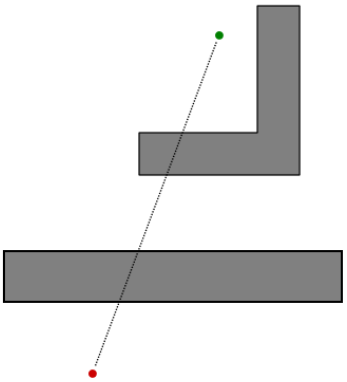

Abbildung 3.4: Bu2-Algorithmus, Grauen Flächen: Hindernisse, Roter Punkt: Startpunkt, Grüner Punkt: Ziel, Verbindungslinie: m-Linie

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

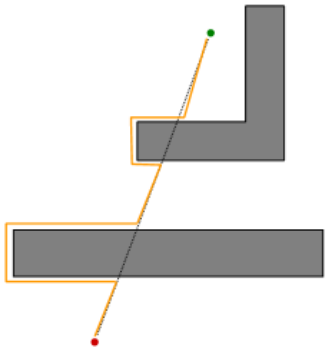

Abbildung 3.5: Bu2-Algorithmus, Fahrweg des Roboters ist in orange dargestellt

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

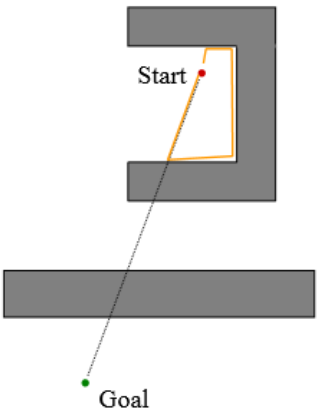

Abbildung 3.6: Bu2-Algorithmus, möglicher Deadlock, Fahrweg des Roboters ist in orange dargestellt

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.

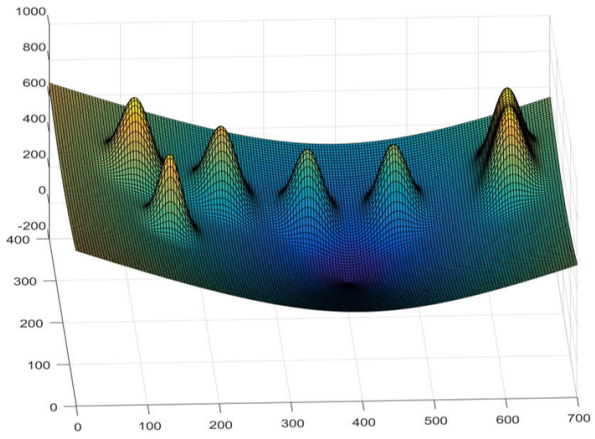

Abbildung 3.7: Beispiel Potentialfeld

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.

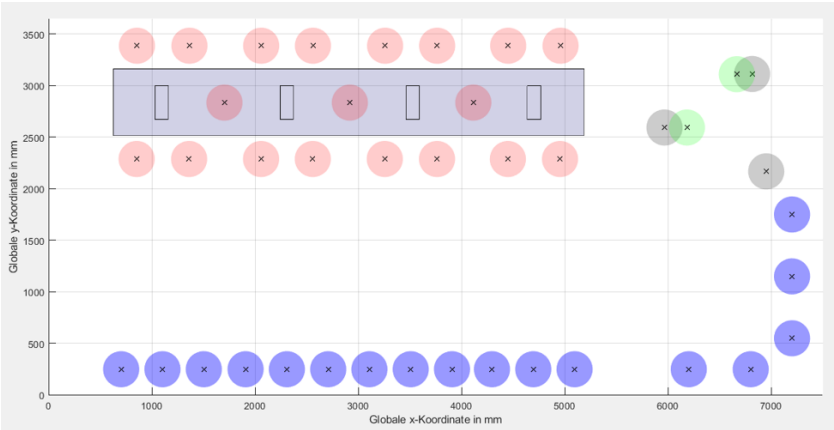

Abbildung 3.8: Konfigurationsraum

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.

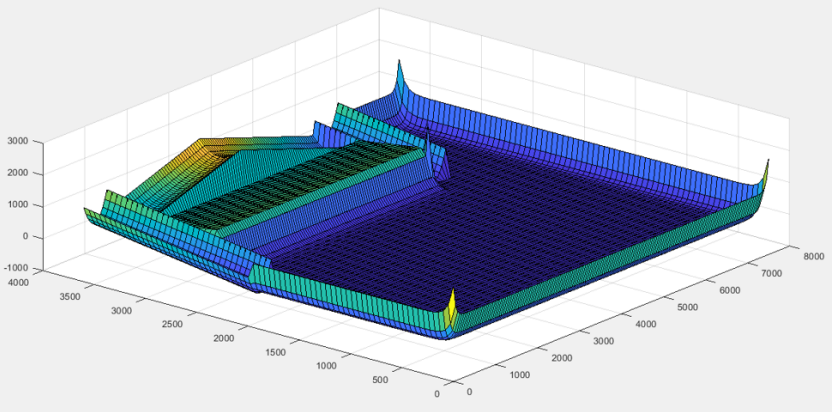

Abbildung 3.9: Potentialfeld des Konfigurationsraumes eingezäunt in hohe Potentiale zur eindeutigen Begrenzung des Raumes, Stationen als ein hohes Potential dargestellt, Rampen und Fahrrinnen zum leiten der Robotinos,

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 eingefärbt, wird die Regelung des Robotinos über die Potentialfeldmethode realisiert. Das dabei genutzte Potentialfeld wird unter Kapitel 4.1 näher erläutert. Im Fertigungsbereich wird die Regelung von der Bahnregelungsgruppe übernommen. Um zwischen den Bereichen zu wechseln, werden Übergabepunkte definiert, an denen der Bereichswechsel sicher ausgeführt werden kann. Diese Übergabepunkte sind in der Abbildung als rote Kreise dargestellt. Dazu wird eine Kommunikation zwischen den Einzelgruppen über eine Schnittstelle definiert. Diese Schnittstelle wird in Kapitel **??** definiert. Desweiteren wird das Konzept zum Vermeiden von Kollisionen in Kapitel **??** näher beschrieben. Dabei wird auf verschiedene Hindernistypen eingegangen. Um das Konzept abzuschließen wird in Kapitel **??** der Ablauf des Programmes dargestellt.

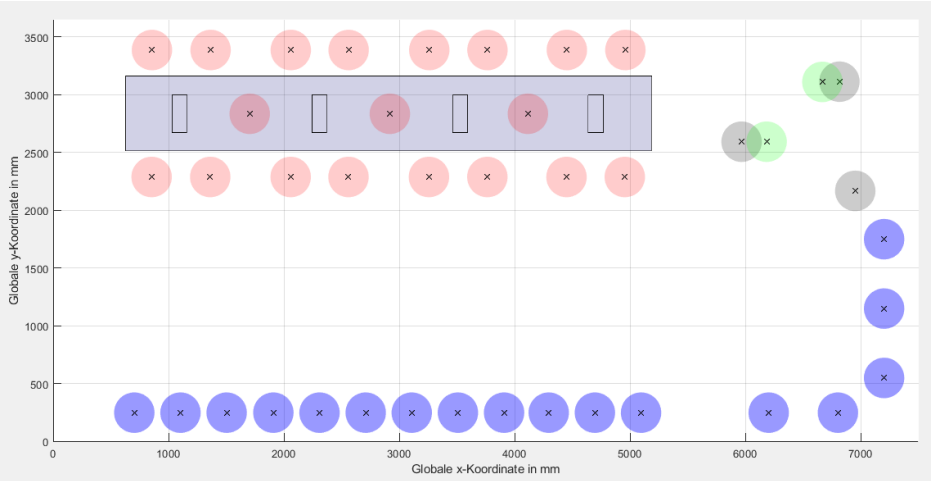

Abbildung 4.1: Bereichseinteilung

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 gewählten 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 müssen dabei drei Funktionen definiert werden, die je nach Position des Robotinos ausgeführt werden. Das Verlassen der Zone 0 erfolgt durch die Übergabe an den Fertigungsbereich, welches in Kapitel **??** näher erläutert wird.
Zone 1 bis 3 dienen zur Rückführung der Robotinos zur Zone 0, dazu wird ein Potentialfeld genutzt, dass uns ermöglicht den Robotino in eine geziehlte Richtung zu lenken. Die dazu genutzten Funktionen sind unter 4.1 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 über eine Geradengleichung noch hinten geführt. Die dbei genutzte Gleichung ist als **??** gekennzeichnet. Zusätzlich wird in Zone 4 für jede Station eine gaußsche radiale Basisfunktion, welche unter Kapitel **??** näher erläutert, 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 führen. 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 Anfahrmöglichkeit aus der Zone 0 zur Ladestation gewährleistet.
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 Übersichtlichkeit 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	Grün	Führung des Robotinos nach Zone 5
2	Blau	Führung des Robotinos nach Zone 3
3	Violet	Führung des Robotinos nach Zone 0
4	Rot	Führung des Robotinos nach Zone 1 und 2
5	Cyan	Führung des Robotinos nach Zone 0

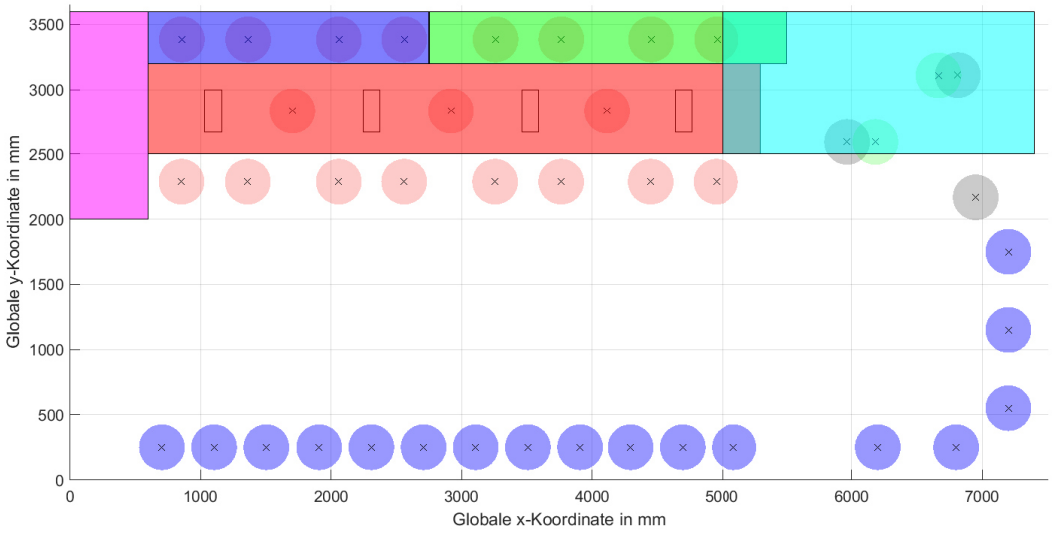

Abbildung 4.2: Potentialfeldzonen

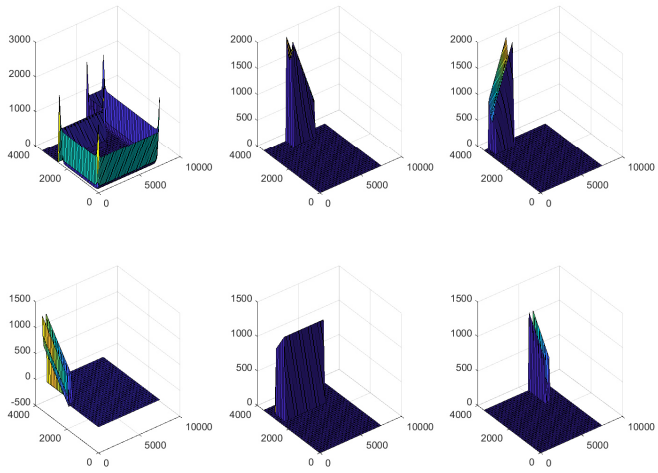

Abbildung 4.3: Potentialfeld der einzel Zonen

$$H_{0(x=0..5500,y=0..2500)} = K_{exp} \cdot (e^{(-\frac{x}{\sigma})} + e^{(\frac{x-7400}{\sigma})} + e^{(-\frac{y}{\sigma})} + e^{(\frac{y-2500}{\sigma})}) \quad (4.1)$$

$$H_{0(x=5500..7400,y=0..2500)} = K_{exp} \cdot (e^{(-\frac{(x-5500)-(y-2500)}{\sigma})} + e^{(\frac{x-7400}{\sigma})} + e^{(-\frac{y}{\sigma})}) \quad (4.2)$$

$$H_{0(x=5500..7400,y=2500..3600)} = K_{exp} \cdot (e^{(-\frac{(x-5500)}{\sigma})} + e^{(\frac{(y-3600)}{\sigma})} + e^{(\frac{x-7400}{\sigma})} + e^{(-\frac{y}{\sigma})}) \quad (4.3)$$

$$H_1 = -K_{gerade} \cdot x + K_{parabel} \cdot \frac{(y-3500)^2}{B} \quad (4.4)$$

$$H_2 = K_{gerade} \cdot x + K_{parabel} \cdot \frac{(y-3500)^2}{B} \quad (4.5)$$

$$H_3 = K_{parabel} \cdot \frac{(x-300)^2}{B} + K_{gerade} \cdot y \quad (4.6)$$

$$H_4 = -K_{gerade} \cdot y \quad (4.7)$$

$$H_5 = K_{parabel} \cdot \frac{(x-5500)^2}{B} - K_{gerade} \cdot y \quad (4.8)$$

4.1.2 Robotino,Sationen und IR-Hindernisse

Die Robotinos, die Stationen und die IR-Hindernisse werden im Potentialfeld als gaußsche radiale Basisfunktion betrachtet. Die genutzte Funktion ist unter **??** definiert. Da sich die gaußsche radiale Basisfunktion aufgrund der genutzten e-Funktion gut ableiten lässt, wird

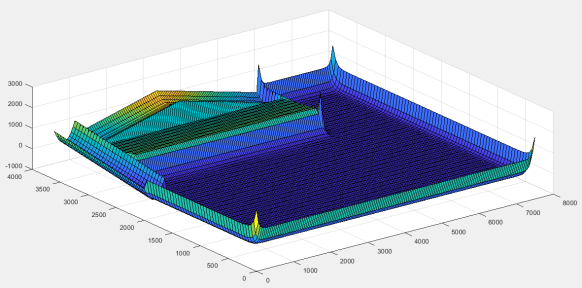

Abbildung 4.4: Potentialfeld der Roboter Umwelt ohne Rausführung

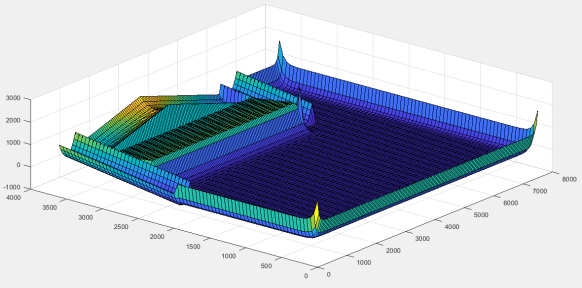

Abbildung 4.5: Potentialfeld der Roboter Umwelt mit Rausführung

diese Funktion genutzt, um ein abstoßendes Potential zu generieren. Das Potentialfeld der gaußschen radialen Basisfunktion ist in Abbildung 4.6 dargestellt. Dabei wird für die Stationen eine Konstante Position eingestellt. Die Position des Robotinos und der IR-Hindernisse werden als flexibel betrachtet, da sie während der laufzeit bestimmt werden müssen.

$$H_{Gau} = K \cdot e^{-\frac{(x-x_{pos})^2+(y-y_{pos})^2}{2\cdot\sigma^2}} \tag{4.9}$$

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 Trägheit verursachte Schwingungen um den Zielpunkt zu vermeiden, wird eine Abstandsabhängiger Faktor als Filterung nachgeschaltet. Dabei wird der Abstand über 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.

Abbildung 4.6: Potentialfeld der gaußschen radialen Basisfunktion

Abbildung 4.7: Potentialfeld Ziel

$$H_{Ziel} = K \cdot \sqrt{(x - x_{Ziel})^2 + (y - y_{Ziel})^2} \tag{4.10}$$

$$K_{Ziel(Abstand_{ist} =< Abstand_{max}} = \frac{Abstand_{ist}}{Abstand_{max}} \tag{4.11}$$

$$K_{Ziel(Abstand_{ist} > Abstand_{max}} = 1 \tag{4.12}$$

4.2 Kollisionsvermeidung durch kritische Bereiche

Bei der Analyse des verwendeten Verfahrens zur Bahnplanung entstehen mehrere Bereiche bei den die Potentialfeldmethode an ihre Grenzen stößt. Es treten lokale Minimas auf die nicht gleich mit dem Zielpunkt sind, zwei Roboter können sich bei gleichem Ziel jeweils gegenseitig am erreichen hindern. Hinter den Stationen und links von der ersten Station ist nicht genügend Platz zur perfekten Bahnplanung mittels Potentialen. Würde ein anderer

Robotino in die gleich Region fahren, ist die geplante Strecke nur noch schwer zu realisieren. Die Wahrscheinlichkeit, dass der Robotino gegen die Wand oder Station "gedrückt"wird steigt stark.

Aus diesem Grund wurden hinter den Stationen EinbahnstraÃen entworfen in den der Robotino geschätzt wieder in den freien Bereich geführt wird. Hiermit wird jedoch noch nicht das Problem der Roboterbegegnung vor den Zielen verhindert.

4.2.1 Selbstorganisation durch Wartebereiche

Um die Entstehung von lokalen Minima zu verhindern wurde die Software der Robotinos um eine Selbstorganisation erweitert. Diese Erweiterung beinhaltet Wartepositionen damit ein Roboter der zur einer bereits besetzten Station fahren soll sich außerhalb des Kollisionsbereichs anstellt.

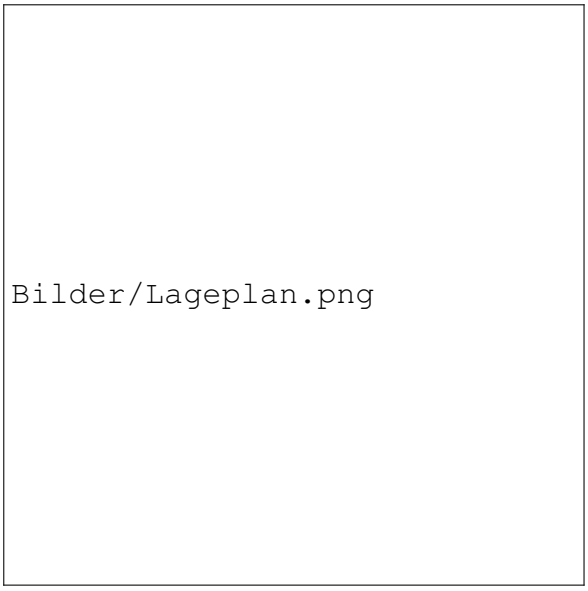

Abbildung 4.8: Lageplan der FIFO PlÃtze

In Abbildung 4.8 erkennt man am unteren Rand der Karte 12 blaue Kreise, welche die Wartebereiche darstellen. Zu jeder Station gehören 3 Wartepositionen. Durch sich verändernde Zählweisen ist der zugehörige Warteplatz zu einer Station immer eindeutig zuzuweisen. Im Kapitel 6 wird dies weiter erläutert.

Mit Hilfe des UDP-Datenpaketes der Kameradaten ist es für einen Robotino möglich

die Positionen der anderen Robotinos zu erhalten. Ist nun die Zielstation bereits von einem anderen Robotino belegt wird mit Hilfe der Selbstorganisationssoftware die erste mögliche Wartepostion als Ziel übernommen. Ist diese auch bereits belegt wird die Warteposition um einen Platz erhöht.
Dem gesamten Anstellporzess liegt das First In First Out-Prinzip zu Grunde.(Im weiteren nur noch FIFO genannt.) Dies bedeutet: Wer sich als erstes anstellt darf auch als erstes wieder herausfahren.
Damit auch die Warteschlange beim herausfahren eingehalten wird, wurde die Software ein weiteres mal erweitert. Die Position der anderen Roboter wird in einem Merker gespeichert und erst zurückgesetzt wenn der Roboter an der nächsten Warteposition bzw. an der Zielposition angekommen ist. Besonders das freischalten der ersten Wartepostion ist kritisch. Wenn ein anderer Robotino die Zielstation verlassen hat, für diese Station aber mehr als ein Robotino wartet, würde beide Robotinos direkt zur Station fahren. Mit Hilfe des Merkers wird die erste Warteposition erst freigeschaltet wenn der Robotino der vorher in der ersten Wartepostion war an der Zielstation angekommen ist. Das bedeutet, dass auch während der längeren Strecke zwischen Wartepositionen und Stationen gesichert ist, dass nur ein Robotino gleichzeitig die Station anfährt. Ist der Roboter dort angekommen wird die erste Wartepostion freigeschaltet und der Robotino auf dem zweiten Platz darf auf den ersten Platz vorfahren.

Diese Warteschlange wird noch ein weiteres mal im Progrmmablauf abgefragt. Hat der Robotino seinen Auftrag abgearbeitet, also sein Werkstück abgeholt bzw. abgelegt, wird die Belegung der Warteschlange zur zugehörigen Station abgefragt. Wartet dort bereits der nächste Roboter fährt der Robotino nach Hinten aus der Station heraus. Von dort gelangt er über die Einbahnstraßen wieder in das freie Feld.

Bei der Erweiterung des Projektes um einen fünften Roboter wie in Kapitel **??** erklärt, wird die vierte Anstellposition nicht neben den anderen Wartepostionen generiert sondern auf Grund von Platzmangel wird als Ziel die Standbyposition einprogrammiert.

4.2.2 Bekannte Hindernisse

Um generelle Begegnungen anderen Robotinos in der normalen Fahrzone zu vermeiden wurde ein Ausweichmanöver entworfen. Befindet sich ein anderer Roboter im Bereich zwischen den gesteuerten Roboter und seinem Ziel und in einer Entfernung von 20 cm wird auf den Zielvektor eine $90°$-Drehung addiert. Der Robotino fährt somit für eine kurze Zeit mit gleichbleibenden Abstand zum Ziel. Ist der behindernde Robotino nicht mehr im gefÃhrdeten Bereich wird wieder der normale Zielvektor eingestellt.

Für eine hohe Dynamik im Gesamtsystem unterscheidet die Software zwischen sich bewegenden Robtinos und stehenden. Anhand der Kameradaten werden die aktuellen Positionen mit deren vorherigen mittels Merker verglichen. Ist die Differenz größer der Toleranz bewegt sich der Robotino. Sich bewegende Robotinos stelle eine größere Kollisionsgefahr dar. Deshalb werden die Potentialfelder jener Robotinos vergrößert. Der gesteuerte Roboter wird somit in einem größeren Abstand vorbei fahren.

4.2.3 Unbekannte Hindernisse

Das verbaute Kamerasystem ist nicht in der Lage andere Objekte, auÃer die mit Aruko-Markern versehenden Robotinos, zu erkennen. Um jedoch auch eine Kollision mit Menschen zuverhindern besitzt der Robotino 9 Infrarot-Sensoren gleichmäßig um den untersten Rand des Roboters verteilt. Detektiert einer der Sensoren ein Objekt ermittelt die Software an welcher Stelle das Objekt erkannt wurde und manipuliert den Zielvektor. Es wird ein um 180° gedrehter Zielvektor ein programmiert. Hat sich das unbekannte Hindernis entfernt wird wieder der normale Zielvektor geschaltet.

4.3 Schnittstellen

Im gesamten Projektablauf ist die Kommunikation mit anderen Gruppen ausschlaggebend für eine perfektes Gesamtkonzept. In der Bahnplanung wurde vorallem mit der Gruppe der Bahnregelung und mit der Gruppe der Fertigungsplanung eng zusammengearbeitet.

4.3.1 Bahnregelung

In sehr engen Bereichen in denen auch noch eine hohe Präzision erforderlich ist stößt die Potentialfeldmethode an ihre Grenzen. Aus diesem Grund wurde der gesamte Bereich direkt am Anfang des Projektes aufgeteilt. Das Potentialfeld wird in den freien Bereiche benutzt und an den Randbereichen des Feldes. Das Anfahren der einzelnen FÃcher innerhalb der Stationen sowie das WerkstÃ$\frac{1}{4}$ckhandling ist an die Bahnregelung übergeben worden.

In Abbildung 4.8 stellen die 8 roten Kreise vor den Stationen die Übergabepunkte da. Befindet sich der Robotino innerhalb eines solchen Kreises, inklusive einer gewissen Toleranz, wird ein Übergabe Bit gesetzt und die Bahnregelung übernimmt das präzise Anfahren zuerst des RFID Lesegerätes und anschließend das Ablegen in ein Stationsfach. (Beim Abholen ist der Prozess andersherum.) Somit gehören zur Schnittstelle nicht nur der Übergabebit. Die Gruppe der Bahnregelung benötigt darüber hinaus auch noch den genauen Auftrag der Fertigungsplanung. Hat die Bahnregelung den Auftrag abgearbeitet wird der Übergabebit am Übergabepunkt getoggelt und der Robotino fährt wieder im Potentialfeld. Dieser Übergabepunkt ist variabel wie bereits in dem Unterkapitel 4.2.1 erläutert.

Da die Bahnregelung sehr präzise die Position des Robotinos für eine hoch genaue Fahrt benötigt, wurde von der Gruppe ein Kalman-Filter für die Robotino Position entworfen. Ohne Beobachter sendet die Kamera bei nicht Erkennung eines Robotinos die Positionsdaten [0,0]. Diese falsche Postion kann zu einigen Fehlern in den Berechnungen führen. Dieser Beobachter ermöglicht eine Fehlerfreie Positionsdarstellung des Roboters auch bei kurzzeitiger Verdeckung des Arukomarkers. Die Beobachterdaten werden auch in der Bahnplanung durchgehende verwendet.

4.3.2 Fertigungsplanung

Die Fertigungsplanung ist für die gesamte Simulation der Fabrikumgebung zuständig. In der Gruppe werden verschiedene Aufträge generiert und intelligent auf die vier Robotinos verteilt. Mittels UDP Kommunikation wird der jeweilige Auftrag an den jeweiligen Roboter gesendet. In Abbildung 4.9 ist der Informationsaustausch dargestellt. Die Informationen, die

Abbildung 4.9: Schnittstelle mit der Fertigungsplanung

die Bahnplanung an die Fertigunsplanung zurücksendet, wurden um einen weiteren Punkt erweitert. Die Ist-Position des Roboters wird ebenfalls gesendet. Diese Position wird von der Fertigungsplanung für die Fabriksimulation benötigt. Im Status Byte wird übermittel was der Robotino zu dem Zeitpunkt gerade ausfÃ¼hrt. Also ob er im Potentialfeld fÃhrt oder eine Stationanfährt, etwas ablegt bzw. etwas aufnimmt. Darüber hinaus werden im Status auf Fehlermeldungen weitergeleitet, dazu im Unterkapitel 4.3.3 mehr.

Von der Fertigungsplanung bekommt das Programm den genauen Auftrag den der Robotino abarbeiten soll. Jeder neue Auftrag erhÃlt auch eine neue Prozessnummer. Der Auftrag besteht aus der Startstation mit zugehörigem Fach, an dem ein Werkstück abgeholt werden soll, und in welches Fach von welcher Station es gebracht werden soll. Damit jedes individuelle Werkstück auch entsprechend erkannt wird, sind an jeder Station RFID Lese-/Schreibköpfe angebracht. Wird ein Werkstück in die jeweilige Station gebracht wird

es dort angemeldet. Beim Aufnehmen wird es abgemeldet. Hierfür werden zwei Bytes in der Schnittstelle verwendet. Die Information über die einzelnen Werkstücke werden von der Fertigungsplanung verarbeitet und in einer Cloud gespeichert.

4.3.3 Fehlererkennung

Eine weitere Übergabeinformation zwischen den drei Gruppen ist das Fehlerhandling. Es wird zwischen drei verschiedenen Fehlern des Robotinos unterschieden:

1. Kein Werkstück vorhanden
2. Stationsfach belegt
3. Werkstück verloren

Beim Anfahren eines Stationsfachs zur Aufnahme eines Werkstückes durch die Bahnregelung wird zwischen den Greifern eine Lichtschranke abgefragt. Ist diese Lichtschranke beim schließen des Greifers nicht durchbrochen wird der erste Fehler ausgelöst. Dieser Fehler wird anschließend an die Fertigunsplanung gesendet. Die Fertigungsplanung entscheidet dann ob ein falscher Auftrag gesendet wurde oder in der 'Fabrik' ein Fehler vorliegt, welcher per manuellen Eingriff behoben werden muss. Der zweite Fehler wird über einen Zähler der Anfahrversuche fpr ein Fach realisiert. Schafft der Robotino innerhalb von drei Anfahrversuchen es nicht das Werkstück abzulegen so wird der zweite Fehler gesendet. Verliert der Roboter während der Fahrt, auch im Potentialfeld, das Werkstück wird der dritte Fehler gesendet. Hier muss manuell das Werkstück aus dem Fahrbereich entfernt werden.

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 Funktionsfähigkeit geprüft. 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 erfüllt, 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 können und die Schnittstelle mit der Fertigungsplanung zu teseten wird ein UDP-Dummy in Simulink erstellt mit dem Aufträge an den Robotino per UDP in Simulink gesendet werden können. Dadurch können mehrere Ziele hintereinander Angefahren werden. Im nächsten Schritt wird die Schnittstelle mit der Bahnregelung gestestet dazu werden beide Simulinkmodelle kombiniert und auf den Robotino geladen. Dadurch können Schnittstellenprobleme zwischen der Bahnregeleung und der Bahnplanung frühzeitig 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 Aufträge zu senden. Im letzten Schritt wird der UDP-Dummy als Fertigungsplanungsersatz erweitert indem sich die Positionen der Werkstücke gemerkt wird und die zufallsgenerierten Aufträge auf ausführbare Aufträge gefiltert wird. Zum Ende des Projektes wird zusätzlich eine Simulation, wie zu Beginn, mit der Erweiterung auf 5 Robotinos ausgeführt um geringe Änderungen im Programm aufzuzeichnen, da die Aufzeichnung von Daten von mehreren Robotinos unter Simulinkrealtime einige Probleme verursacht. Unter Kapitel **??** wird eine Simulation des Ausweichmanövers dargestellt.

5.1 Simulation des Ausweichmanövers

In diesem Abschnitt wird die Simulation des Ausweichmanövers 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 fährt von der Position [500 1000] zur Station 7 welche sich ganz rechts befindet. Robotino 2 in blau

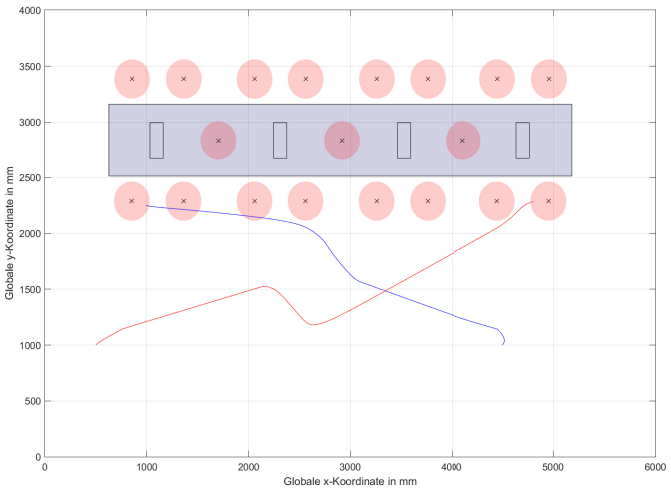

Abbildung 5.1: Simulation mit Ausweichfunktion

dargestellt fährt 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 Abstoßverhalten, 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

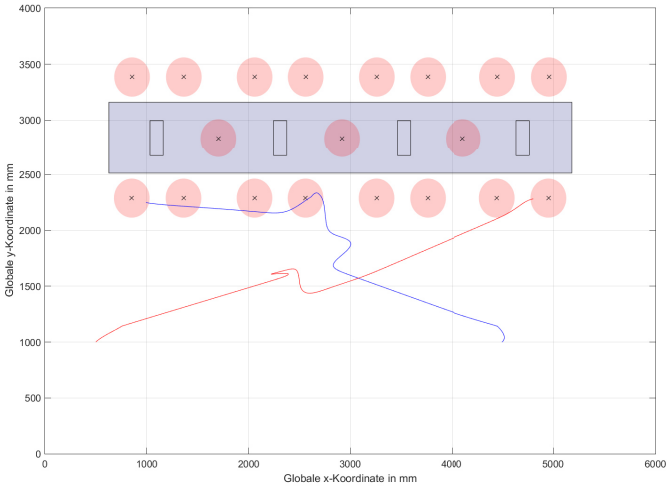

Abbildung 5.2: Simulation ohne Ausweichfunktion

6 Implementierung und Zielsysteme

In diesem Kapitel wird die Umsetzung der einzelnen Algorithmen des in Kapitel 4 erläuterten Konzepts mit Hilfe der Software Matlab/Simulink erklärt. Für einen besseren Überblick zum dem Konzept zeigt die Abbildung 6.1 ein Ablaufdiagramm des Programms der Bahnplanung.

Zunächst wird der Auftrag der Fertigungsplanung verarbeitet und parallel dazu wir die eigene Position des Robotinos sowie auch die Positionsdaten der anderen Robotinos bezogen. Nachdem die Selbstorganisation durch Wartebereiche abgearbeitet wurde wird das erhaltene Ziel weitergeleitet. Mit dem bekannten Ziel wird der Zielvektor bestimmt. Die Hinderniserkennung läuft nun gleichzeitig. Hat der Robotino sein Endziel erreicht kann ein neuer Auftrag abgearbeitet werden.

Hat ein Robotino zur Zeit keinen Auftrag wird als Ziel die Standby-Position einprogrammiert. Zunächst wird der Grundalgorithmus zur Zielgebung erläutert und anschließend um die einzelenen Unterpunkte des Konzeptes erweitert.

6.1 Potentialfeld

6.2 Selbstorganisation durch Wartebereiche

Die in dem Unterkapitel 4.2.1 erklärte intelligente Selstorganisation wird im folgenden erklärt und es wird gezeigt wie der Algorithmus in der Software Simulink umgesetzt wird.

Das Ablaufdiagramm in Abbildung 6.2 stellt den Ablauf des Algorithmus dar.

Die Positionen der anderen Robotinos werden über den ’FIFO Merker’ geprüft und es wird geschaut ob die anderen Roboter Positionen einnehmen die ein zugehöriger Warteplatz für die Zielstation darstellen.

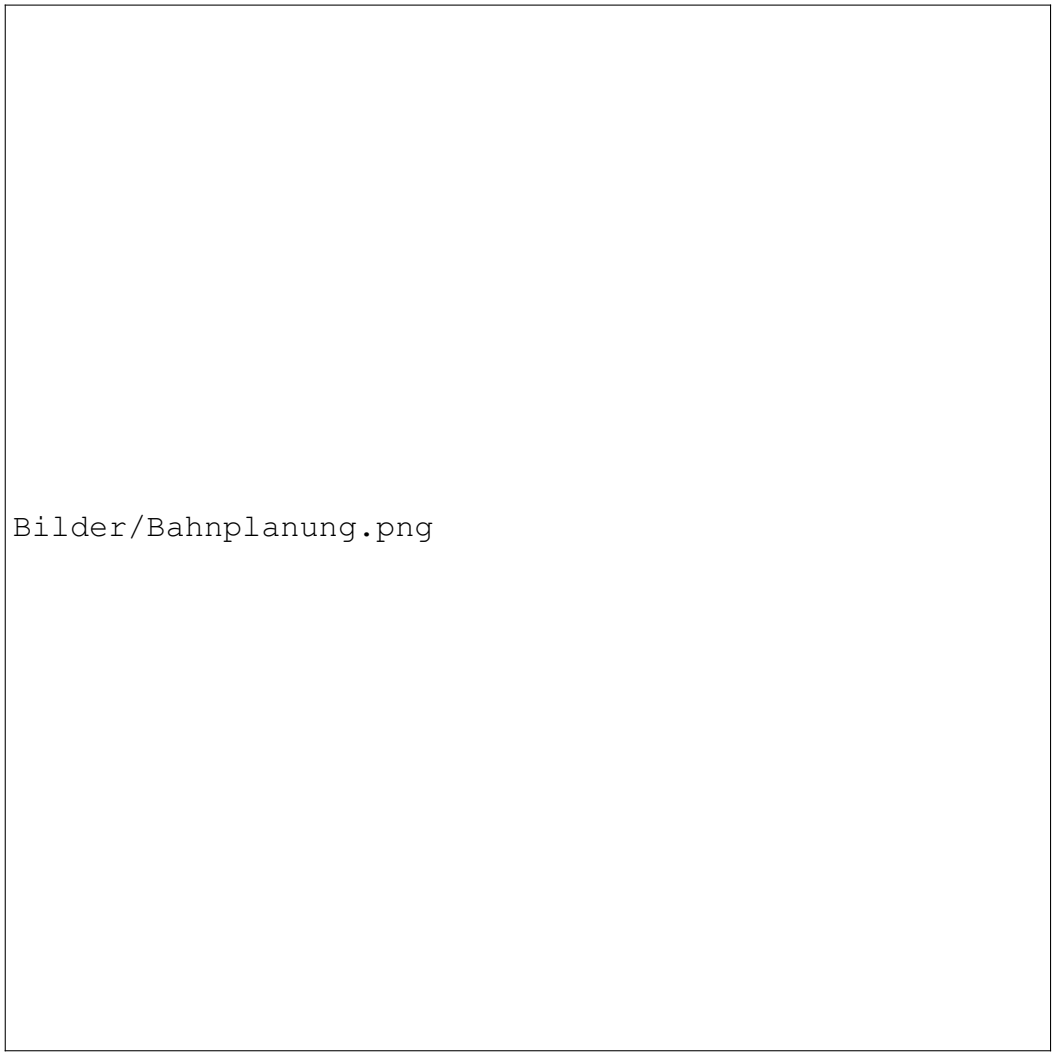

Abbildung 6.1: Ablaufdiagramm Konzept

Die Roboterpositionen werden in einem Array der X- Positionen und einem der Y-Postionen gespeichert. Die 12 bestehenden in gleichmäßigen Abstand aufgeteilten Wartebereiche können mit der Formel 6.1 in Zeile 8 des Matlabcodes **??** in X-Richtung berechnet werden.

$$N(n) = round((X(n) - 703)/400) + 1; \tag{6.1}$$

Stimmen die X-Positionen der Robotinos mit den Bereichen Ã$\frac{1}{4}$berein und sind die Y-Positionen am linken Rand des gesamten 'Fabrik' Bereiches, wird der Wartebereich an dem ein Robotino mit einer jedem Robotino spezifischen Nummer belegt. Damit die Wartebereiche auch in jedem Zyklus wieder neu berechnet werden wird der Merker zunÃchst auf 0 gesetzt und in jedem Zyklus mit neuen Werten beschrieben.

[caption=FIFO Merker,label=Fifomerker,basicstyle=] function [FifoArray1,StationArray1,LadeArray1]= fcn(Robbi,FifoArray,LadeArray) X=[Robbi(1) Robbi(4) Robbi(7) Robbi(10)]; Y=[Robbi(2) Robbi(5) Robbi(8) Robbi(11)]; FifoArray1=zeros([1 12]); FifoArray1=FifoArray(1:12); N=zeros([1 5]); for n= 1 : 1 : 4 N(n)=round((X(n)-703)/400)+1; elseif (Y(n)<450 (X(n)>100 X(n)<5250) N(n)<=12 N(n)>0) for i=1:12 if FifoArray1(i)==n FifoArray1(i)=0; end end FifoArray1(N(n))=n; ... ... end Auch die Belegung der Stationen wird mit dem selben Algorithmus berechnet. Einzig die Bereichsabfrage ändert sich und die Formel 6.1 für die Berechnung der Belegung. Hierbei ändert sich allerdings nur die Unterteilung und der Offset der Formel.
Die Merkerwerte werden in einem Stateflowdiagramm weiterverarbeitet. Die EingÃnge des Stateflowdiagramms sind die Auftragsdaten, die Merkerdaten, und die genauen Positionsdaten des Robotinos aus Beobachter und Kamerawerten. Damit das System die Wartepositionen der zugehörigen Zielstationen für die Zuweisung verwendet wird die Formel 6.2 verwendet. Die 12 Wartebereiche können so explizit den 8 Stationen zugeordnet werden. Zwei Stationen teilen sich jeweils drei Wartebereiche.

$$ersterFifo = floor(Zielstation/2) \cdot 3 + 2 \cdot mod(Zielstation, 2) + 1 \quad (6.2)$$

$$zweiterFifo = floor(Zielstation/2) \cdot 3 + 1 + 1 \quad (6.3)$$

$$dritterFifo = floor(Zielstation/2) \cdot 3 + 2 - 2 \cdot mod(Zielstation, 2) + 1 \quad (6.4)$$

Ist der erste Bereich belegt wird der zweite Bereich abgefragt ist dieser belegt wird der Dritte abgefragt. Sind alles drei Bereich belegt gibt das Stateflow diagramm den höchsten Wert aus. Beim höchsten Wert wird die Standbyposition als Ziel einprogrammiert.

Wird im nächsten Zyklus der nächst kleinere Wartebereich frei so wird dieses als Ziel angefahren. In dem Fall wenn ein anderer Robotino den ersten Wartebereich verlässt und zur Zielstation fährt besteht ein Sonderfall. Da die Strecke zwischen den beiden Punkten ca. zwei Meter beträgt dauert diese Fahrt mehrere Zyklen. Damit ein dahinter stehender Robotino in dem Moment nicht auch direkt die Station anfährt, weil alle anderen Bereiche frei sind, wird eine weitere Position abgefragt. Die Stationsbelegung wird abgefragt. Die Station wird erst wieder freigegeben wenn sie im Stationsmerker nicht mehr belegt ist.

6.3 Bekannte Hindernisse

Für eine verbesserte Kollisionsvermeidung wurde das Ausweichmanöver, wie in Unterkapitel 4.2.2 bereits erklärt, verwendet. Die Aufgabe des Manövers ist es den Zielvektor des Robotinos so zu verÃndern, dass er einem sich im Weg befindenden Robotino mit einem größeren

Abstand ausweicht. Es werden die Position des Robotinos benötigt sowie auch die Positionen der anderen Robotinos, die Störpositionen. Darüber hinaus benötigt das Manöver den Zielvektor.
Zunächst wird analysiert in welchem Winkel die Störpositionen sich zum Zielvektor befinden. Liegt ein anderer Robotino innerhalb von 30 zum Zeilvektor wird ein Ausweichwinkel berechnet.

$$Ausweich(1,i) = +asin(Abstand(2,i)/Hyp) - pi/2 \tag{6.5}$$

Der erste Wert des Ausweichs-Array aus Formel 6.5 ist der Ausweichwinkel. über eine Trigonometrische Funktion wird der genaue Winkel des Zielvektors zum Ziel ausgerechnet und 90 addiert. Solange eine Störposition innerhalb der 30 liegen bleibt der Ausweichwinkel erhalten. Ist der im Weg liegende Roboter weitgenug umfahren wird der originale Zielvektor einprogrammiert.

6.4 Unbekannte Hindernisse

Unbekannte Hindernisse werden nur mit Hilfe der 9 verbauten Infrarotsensoren erkannt. Zyklisch werden diese Sensoren abgefragt. Detektiert einer der Sensoren ein Hindernis wird zunächst überprüft an welcher Position relativ zur Roboterausrichtung sich das Hindernis befindet. Der Ablauf wird im Code **??** dargestellt:
[caption=Infraroterkennung,label=infra,basicstyle=] for x=1:9 if (isnan(IRDaten(x))==0) (IRDaten(x) <= 150) IRWerte(1,x)=cos(((x-1)*40*pi/180)+RobPos(3))*(IRDaten(x)+200)+RobPos(1); IRWerte(2,x)=sin(((x-1)*40*pi/180)+RobPos(3))*(IRDaten(x)+200)+RobPos(2); else IRWerte(1,x)=NaN; IRWerte(2,x)=NaN; end end

6.5 Geschwindigkeit bestimmen

Um die Sollgeschwindigkeit des Robotinos zu bestimmen, wird das Subsystem Vektorberechnung genutzt. Dieses Subsystem ist wie in Abbildung 6.3 dargestellt, aufgebaut. Das Subsystem hat dabei den in Abbildung 6.4 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 **??** näher erläutert. Wie die fahrenden Robotinos erkannt werden, wird in Kapitel **??** beschrieben. Anhand der IR Daten und der Informationen über die fahrenden Robotinos wird dann der Geschwindigkeitsvektor bestimmt. Die dabei genutzte 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 **??** näher beschrieben. Im nächsten Schritt wird dann das Ausweichmanöver ausgeführt. Die Implementierung des Ausweichmanövers wird in Kapiitel **??** beschrieben.

6.5.1 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 größer als 150 mm ist. Bei einem Abstand kleiner gleich 150 mm wird die Position der Hindernisse in globalen Koordinaten bestimmt und ausgegeben.

6.5.2 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.

6.5.3 Geschwindigkeitsvektor bestimmen

6.5.4 Ausweichrichtung bestimmen

6.5.5 Ausweichmanöver addieren

Abbildung 6.2: Ablaufdiagramm der Selbstorganisation

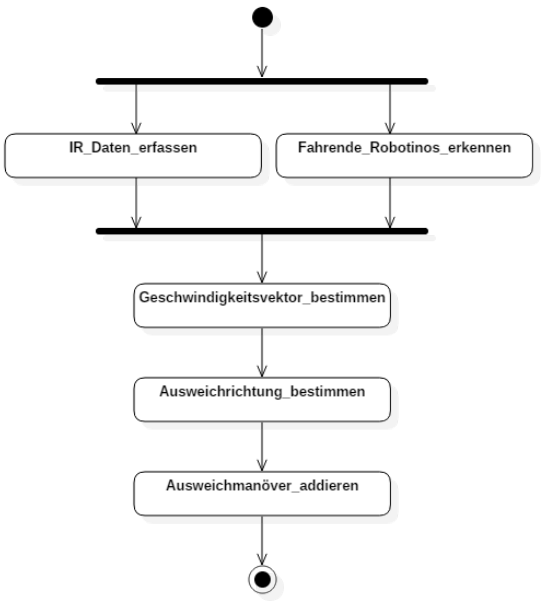

Abbildung 6.3: Subsystem Vektorberechnung

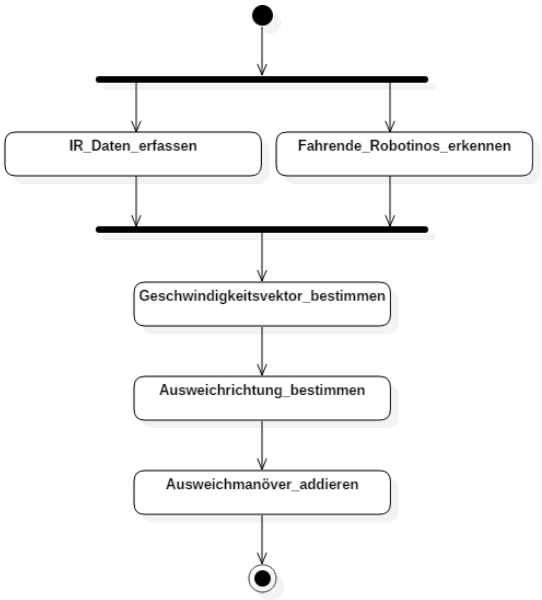

Abbildung 6.4: Ablaufplan des Subsystems Vektorberechnung

7 Robotino 2.0

In diesem Kapitel wird beschrieben welche Anpassungen am Programm vorgenommen werden müssen, um das Programm von den alten Robotinosauf dem neuen Robotino lauffähig zu bekommen. Um das Programm auf den Robotino 2.0 anzupassen wird eng mit der Bahnregelung und Gewerk4 zusammengearbeitet. Die hauptänderung 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.

8 Validierung

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