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 28. Mai 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

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.

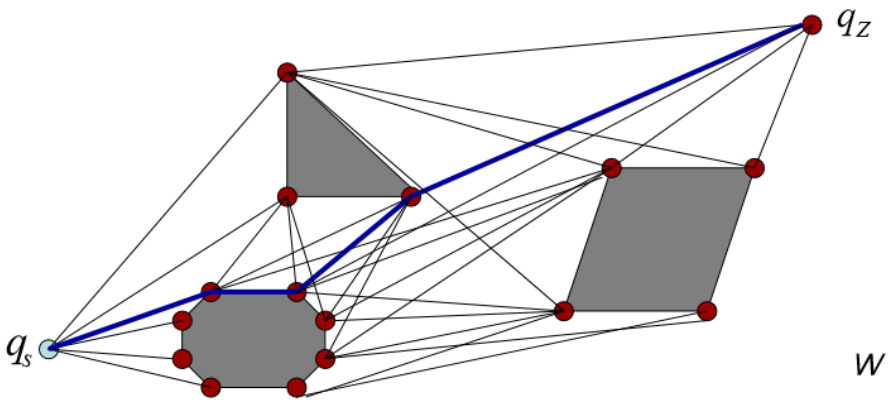

Abbildung 3.1: Beispiel Sichtbarkeitsgraph, Graue Felder sind Hindernisse, blaue Linie ist optimaler Pfad

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.

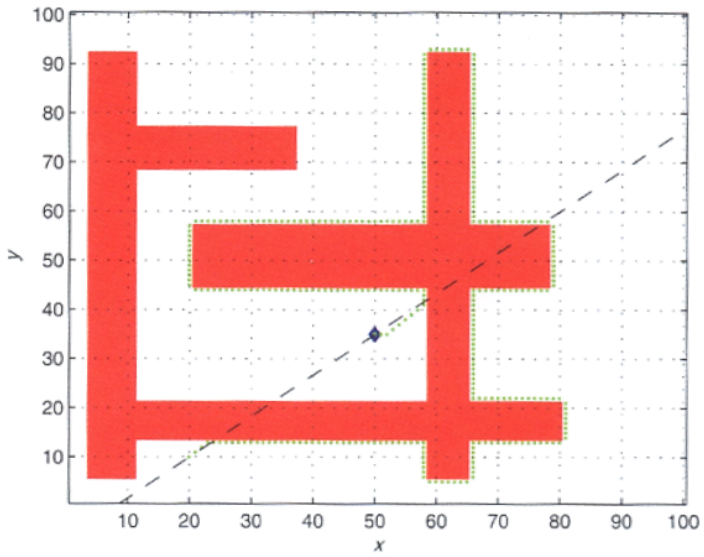

Abbildung 3.2: Beispiel Occupancy Grid, Rot-Hindernisse, Weiß-befahrbarer Raum

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.

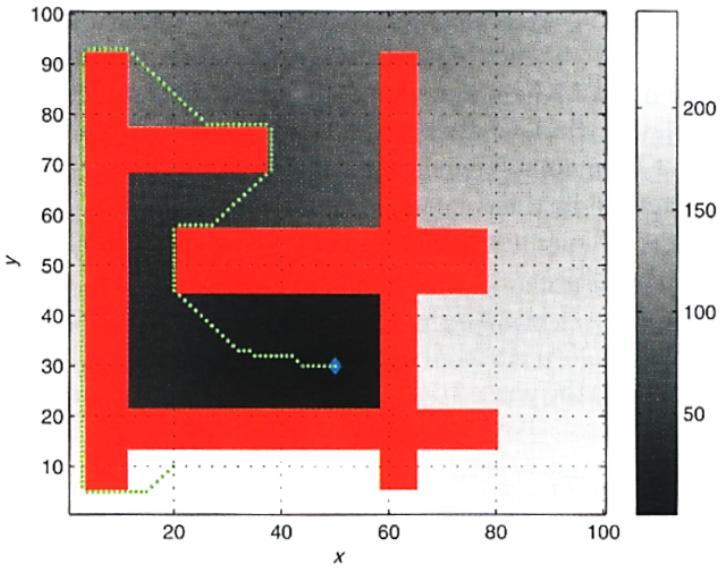

Abbildung 3.3: Beispiel Occupancy Grid mit D*, Rot-Hindernisse, Intensität des Graus im Hintergrunds repräsentiert die Entfernung zum Ziel; blauer Punkt entspricht Ziel

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.

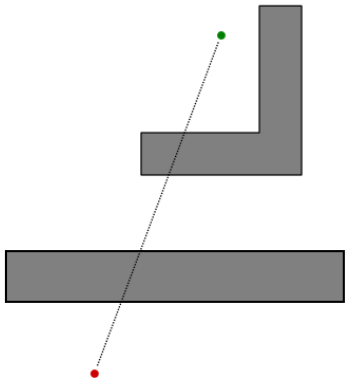

Abbildung 3.4: Bu2-Algorithmus, Grauen Flächen: Hindernisse, Roter Punkt: Startpunkt, Grüner Punkt: Ziel, Verbindungslinie: m-Linie

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

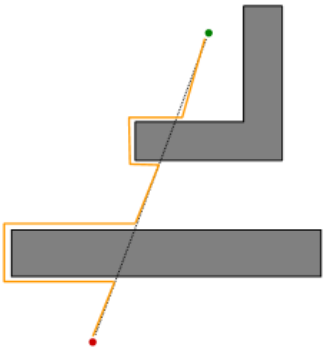

Abbildung 3.5: Bu2-Algorithmus, Fahrweg des Roboters ist in orange dargestellt

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

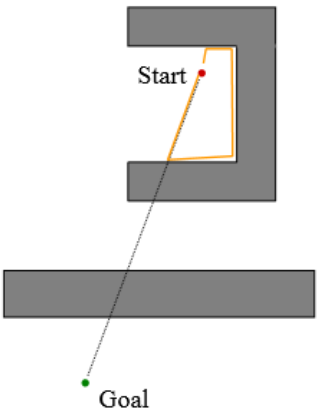

Abbildung 3.6: Bu2-Algorithmus, möglicher Deadlock, Fahrweg des Roboters ist in orange dargestellt

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.

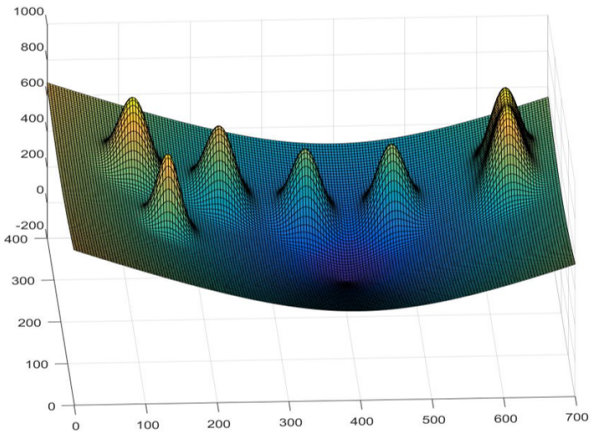

Abbildung 3.7: Beispiel Potentialfeld

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.

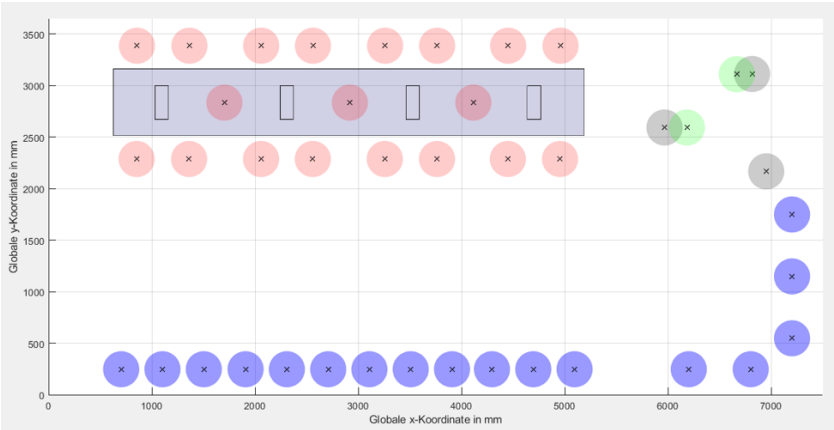

Abbildung 3.8: Konfigurationsraum

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.

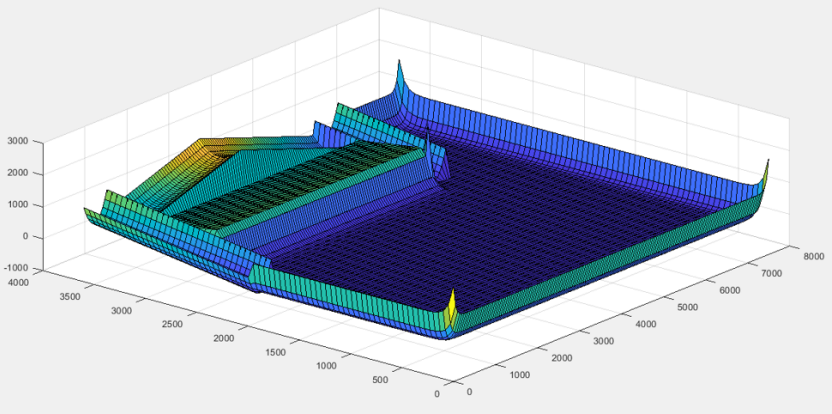

Abbildung 3.9: Potentialfeld des Konfigurationsraumes eingezäunt in hohe Potentiale zur eindeutigen Begrenzung des Raumes, Stationen als ein hohes Potential dargestellt, Rampen und Fahrrinnen zum leiten der Robotinos,

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 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, 28. Mai 2018	
Ort, Datum	Unterschrift