ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Master-Thesis

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)
Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

Schwarmroboter (vorläufiger Arbeitstitel)

vorgelegt von
Sven Manier

betreut und begutachtet von
Prof. Dr. Markus Esch
Prof. Dr. Martina Lehser

Saarbrücken, 10. 05. 2018

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, 10. 05. 2018

Sven Manier

Zusammenfassung

Lorem Ipsum

Inhaltsverzeichnis

1 Einleitung

Diese Thesis wird im Rahmen eines Informatik-Studiengangs mit dem Abschluss eines Master of Science geschrieben. Worum es genau geht und welche Bedingungen ich voraussetze wird im folgenden Kapitel beschrieben.

1.1 Motivation

Schaut man in die Natur sieht man sehr häufig, dass sich Tiere in riesigen Gruppen organisieren die sich meist ohne zentralen Koordinator bewegen. Bei Fischschwärmen handelt jedes Individuum vollkommen autonom und nur anhand dessen was es in seiner (unmittelbaren) Umgebung erkennen kann. Dennoch können komplexe Handlungen wie die Flucht vor einem Räuber oder die gemeinsame Wanderung zu einem neuen Futterort koordiniert werden. In manchen Herden dagegen besteht ein 'Schwarm' aus Untergruppen, welche zwar einen Anführer haben, der die Richtung für seine Gruppe angibt, diese Führer sich jedoch am Rest der Herde orientieren und somit ein Schwarmverhalten entsteht, bei dem die Akteure aus koordinierten Einzelgruppen und nicht aus einzelnen Tieren bestehen. Ebenso wird es bei Vögeln vorgefunden, welche durch ihre Koordination in einer energiesparenden Formation fliegen können. Solch ein Schwarmverhalten hat den großen Vorteil, dass jeder Akteur nur für seine eigene Bewegung verantwortlich ist. Der Denkaufwand verteilt sich auf alle Mitglieder und die, meist langsame, Kommunikation untereinander wird auf ein Minimum reduziert.

Man fing früh an dieses Verhalten zu studieren und es auf Roboter anwenden zu wollen. Schwärme von Robotern können in vielerlei Hinsicht eingesetzt werden, darunter Sensornetzwerke die dynamisch einen bestimmten Raum überwachen oder Arbeitsroboter die ohne weiteres Zutun bestimmte Tätigkeiten wie Transport übernehmen. Zentrale Systeme mit Koordinator können dafür allerdings schnell unzulänglich werden. Ein großer Nachteil ist schon dadurch gegeben, dass es diesen einen Koordinator geben muss. Fällt er wegen einem technischen Defekt aus, steht der Schwarm still. Baut man einen Backup-Koordinator hat man einen vielfach erhöhten Aufwand beim Design der Schwarm-Architektur, einen größeren Aufwand in der Kommunikation, da der Backup mit einbezogen werden muss, und letztlich auch einen höheren Preis, weil der rechenstarke Computer gleich mehrmals gebaut werden muss. Auch die Reichweite des zentralen Koordinators kann dem Schwarm schnell zum Verhängnis werden. Hat ein Schwarm Beispielsweise die Aufgabe ein Gebiet zu erkunden um später eine Karte aus den Sensorwerten zu erstellen, muss der Koordinator entweder stets in Reichweite aller Roboter sein um mit ihnen kommunizieren zu können oder er muss die Kommunikation durch teures Flooding im Kommunikationsnetz aufrecht erhalten. Bei der Koordination von Drohnen kann allein die Erreichbarkeit zu einem sehr realen Problem werden, da der Koordinator aufgrund seiner Rechenleistung ein vielfaches der Arbeiter wiegen wird. Dies führt zu beträchtlich steigenden Kosten für die Motoren und den Akku des Koordinators, beeinflusst aber auch sein Verhalten, da er deutlich träger ist als die anderen. Wer das Bild eines Fischschwarms vor Augen hat, wird vermutlich auch direkt an die Anzahl der Mitglieder des Schwarms denken müssen. Zentrale Systeme sind nur schwer skalierbar und müssen von Anfang an für die jeweilige Größe konzipiert werden.

Schwarmverhalten zeichnet sich dadurch aus, dass die einzelnen Mitglieder des Schwarms ihre Bewegung selbst koordinieren. Dabei wird sich entweder nur nach der Bewegung der anderen gerichtet oder es werden sehr lokal Informationen ausgetauscht. Die Rechenleistung wird auf alle Mitglieder gleichmäßig aufgeteilt und jeder ist nur für sich selbst verantwortlich. Durch die lokale Kommunikation stellt die Größe oder Ausbreitung des Schwarms kein Hindernis dar, da ein Roboter nur Verbindung zu seinen nächsten Nachbarn braucht und auch auf teures Flooding verzichtet werden kann. Die Größenordnung des Schwarms spielt dabei auch eine untergeordnete Rolle, da ein Roboter aufgrund des lokalen Informationsaustauschs ohnehin nur wenige Kommunikationspartner hat. Ebenfalls ist solch ein Schwarm ausfallsicherer da eine ausgefallene Einheit (egal welche) letztlich nur sich selbst stoppt.

1.2 Aufgabenstellung

In dieser Thesis befasse ich mich mit der Thematik einen Roboterschwarm dafür nutzen zu können Transportaufträge zu bearbeiten. Ein Auftrag wird an beliebig viele Roboter eines Schwarms verteilt, woraufhin diese selbstständig eine Untergruppe bilden, die für die Ausführung des Auftrages zuständig ist. Da die zu transportierenden Objekte beliebiger Größe und Form sein können, gilt es unter Umständen eine Formation zu bilden und diese bis zum Ende des Auftrags einzuhalten. Dabei gilt es Kollisionen mit anderen Objekten zu vermeiden und selbstständig navigieren zu können. Eine eventuell benötigte Karte ist zunächst voreingestellt, wird im späteren Verlauf aber, von bereits im System der Roboter vorhandenen Funktionen, selbstständig erstellt werden.

Hauptgegenstand dieser Thesis ist die Erarbeitung des Konzepts und die Umsetzung der Koordination bzw. des Verhaltens der Roboter als Algorithmus, sodass dieser produktiv eingesetzt werden kann. Eine Implementierung findet prototypisch auf den Robotern selbst oder einer Simulation dieser statt und dient vor allem dazu, festzustellen ob die theoretischen Überlegungen auch in der Praxis angewendet werden können.

Rahmenbedingungen

Ich gehe bei dieser Arbeit davon aus, dass der Roboterschwarm aus AGVs (Autonomous Guided Vehicles) besteht, welche aktuell an der Hochschule für Technik und Wirtschaft des Saarlandes entwickelt werden. Diese Roboter bieten bereits eine Vielzahl an Technik und (Software-)Funktionen, welche alle uneingeschränkt genutzt werden können. Die AGVs und ihre Fähigkeiten werden in einem späteren Kapitel genauer vorgestellt.

ja?

Ebenfalls gehe ich davon aus, dass der Transportauftrag nicht erst verteilt werden muss. Beim Start des Systems wissen die jeweiligen Roboter bereits von ihrem Auftrag und müssen allenfalls noch ihre Gruppe selbstständig zusammenfinden.

1.3 Aufbau der Thesis

Zunächst werde ich im Kapitel Grundlagen und Hintergründe auf notwendige Grundlagen zum Thema Schwarmverhalten eingehen. Darunter verschiedene Definitionen, Abgrenzungen und Einstiege in für Schwarmverhalten wichtige Themen. Danach folgt eine Analyse der Aufgabenstellung im Detail. Es wird darauf eingegangen welche Probleme sich bei der Aufgabenstellung auftun und die es zu lösen gilt und welche Fähigkeiten bei den Robotern genau gefordert werden. Im darauf folgenden Kapitel wird bereits vorhandene Forschung zum Thema Schwarmverhalten untersucht. Diese Forschung wird

sich nicht ausschließlich um den Bereich der Roboter-Technik drehen, zumal es für das behandelte Thema noch keine ausreichende Literatur gibt. Die Forschung wird darauf geprüft, welche Themenbereiche meiner Thesis bereits abgedeckt werden und welche noch offen sind. Als nächstes folgen die Kapitel der Konzeption und der Implementierung. In diesen wird das Konzept für die Roboter zunächst theoretisch, dann praktisch entwickelt werden wird. Erkenntnisse aus der prototypischen Entwicklung werden wieder zurück in das Konzept fließen und es verbessern. Am Ende der Thesis findet eine Evaluation der erreichen Ziele statt und, sollten einige nicht erreicht worden sein, eine Erläuterung woran es gelegen hat. Außerdem wird es einen Überblick darüber geben, was mit mehr Zeit und Ressourcen noch hätte erreicht werden können, bzw. wo man Ansetzen könnte um die Entwicklung weiterzuführen.

2 Grundlagen

In diesem Kapitel werden einige grundlegende Begriffe erklärt und voneinander abgegrenzt, welche für das Verständnis dieser Thesis wichtig sind und sonst für Verwirrung sorgen könnten.

2.1 Begrifflichkeiten und Definitionen

Im Verlauf der Thesis werden einige Begriffe immer wieder verwendet werden die eine feste Definition innerhalb dieser Thesis haben und nicht verwechselt oder missverstanden werden sollten. Solche Begriffe werden im folgenden Abschnitt definiert.

Nachbarschaft Die Nachbarschaft einer Einheit ist definiert durch einen Kreis mit fixem Radius um die Einheit herum. Andere Einheiten die sich, bezogen auf ihre Position, innerhalb dieses Kreises befinden sind Teil der Nachbarschaft dieser Einheit, Einheiten außerhalb nicht.

Aufnahmeort Der Aufnahmeort ist der Ort an dem die Roboter die zu transportierende Ware aufnehmen.

Abgabeort Der Abgabeort ist der Ort an dem die Roboter die zu transportierende Ware abgeben.

Einheit Wenn von der Einheit gesprochen wird, ist damit ein einzelnes Mitglied des Schwarms gemeint.

2.2 Selbstorganisation

Als Selbstorganisation bezeichnet man Prozesse, bei denen aus einer ungeordneten Menge mit Hilfe von lokaler Kommunikation ein global geordnetes System entsteht. Es entsteht oft durch zufälliges Verhalten, welches positives Feedback bekommt. In der Robotik versteht man unter Selbstorganisation Gruppen von Systemen die eigenständig, ohne zentralen Anführer arbeiten können.

Zentrale Systeme

Normale Systeme sind heterogen aufgebaut. Zentralisierte Systeme mit Robotern bestehen grundsätzlich aus einer zentralen Einheit, welche als das Rechenzentrum dient und mehreren Robotern die die Arbeiter darstellen. In der Praxis kommen schließlich noch viele weitere Systeme hinzu die unter Anderem der Ausfallsicherheit und Informationsaufzeichnung dienen. Die zentrale Einheit bekommt von den Arbeitern Informationen wie Sensorwerte zu gesendet, die zentrale Einheit berechnet daraufhin das weitere vorgehen und die Aktionen für die einzelnen Arbeiter und sendet sie diesen schließlich zu. Solche

zentralisierten Systeme sind wenig skalierbar und sehr kompliziert in der Implementierung, da die additionalen Systeme eingebunden werden müssen. Auch die Komplexität der Nachrichten innerhalb des Netzwerks nimmt zu, da letztlich nicht nur von/zu der zentralen Einheit kommuniziert werden muss, sondern auch die Kommunikation zu Backup, Datenbanken und anderen Hilfssystemen integriert und ausgeführt werden muss.

Selbstorganisierte Systeme

Systeme mit Selbstorganisation sind homogen aufgebaut. Jede Einheit ist für sich selbst verantwortlich und muss, gegeben der Abwesenheit der zentralen Steuereinheit, gezwungenermaßen selbst entscheiden was sie zu tun hat. Durch den homogenen Aufbau und die fehlenden Hilfssysteme gibt es keinen Grund für unterschiedliche Programme/Algorithmen welche aufeinander abgestimmt werden müssen, sondern man braucht nur ein einziges Programm, welches auf jedem Roboter gleichermaßen eingespielt wird und sich nur durch den darunterliegenden Hardware-Layer und einer einzigartigen Identifizierung von den andere Unterscheidet. Entscheidungen werden entweder für sich selbst oder in Gruppen mithilfe verteilter Algorithmen getroffen. Innerhalb von Abstimmungen oder kleinerer Tätigkeiten kann es zur Bildung von Hierarchien und der Erstellung von Anführern kommen, diese Konstrukte sind aber meist wieder verworfen, sobald die entsprechende Abstimmung oder auszuführende Tätigkeit erledigt wurde. Dadurch dass jede Einheit für sich selbst rechnet und keine permanente Kommunikation zu einer zentralen Stelle notwendig ist, lassen sich homogene Gruppen besser skalieren, wenn auch die Algorithmen für die Kommunikation insgesamt aufwendiger sind.

2.3 Schwarmverhalten

Bei Schwarmverhalten geht es darum, dass sich eine Gruppe von (meist homogenen) Einheiten ohne zentrale Kontrolle gemeinsam organisiert und eine geordnete Bewegung entsteht.

Die 4 Regeln Schwarmverhalten lässt sich grob auf 4 Regeln zurückführen:

1. Zusammenhang: Versuche deinen Nachbarn nahe zu sein
2. Ausrichtung: Passe deine Bewegungsrichtung deinen Nachbarn an
3. Abschottung: Vermeide Kollisionen mit deinen Nachbarn
4. Flucht: Fliehe vor Dingen, die eine potentielle Gefahr darstellen
5. **https://link.springer.com/article/10.1007%2Fs00354-007-0009-5**

Die Grundprinzipien von Schwarmverhalten lassen sich ohne jegliche Kommunikation umsetzen. Sie lassen sich durch reine Beobachtung der Nachbarschaft und entsprechender Reaktion auf das Verhalten der Nachbarn durchsetzen. Verfügen die Einheiten nicht über die notwendige Sensorik um die Nachbarschaft beobachten zu können, lässt sich dies aber durch entsprechende Kommunikation der eignen Werte ausgleichen.

Kommunikation innerhalb eines Schwarm findet, wenn überhaupt, nur mit der Nachbarschaft statt. Experimente mit Drohnen innerhalb eines Vogelschwarms zeigten, dass die Reichweite der Kommunikation recht gering ist und der Informationsfluss mit der Entfernung überproportional abnimmt. Das bedeutet, Informationen werden nie vollständig

weitergeben, was dazu führt dass der Fluss letztlich zum erliegen kommt und eine Information somit nur lokal verfügbar ist. **https://academic.oup.com/beheco/article/22/6/1304/220324**

Wer sich die 4 Regeln anschaut wird auch bemerken, dass es keine Regel gibt die einen Roboter normal dazu veranlassen würde still zu stehen. Ein Stillstand im Schwarm ist daher immer auf besondere Bedingungen (zum Beispiel fehlender Platz für Bewegung) oder Fehler im System zurückzuführen. Ein Schwarm der wartet, ist gut vergleichbar mit dem Rauschen bei älteren Fernsehern, bei dem überall zwar Bewegung zu erkennen ist, aber keine die in eine bestimmte Richtung führt. Der Schwarm steht also als Gesamteinheit still, die einzelnen Einheiten bewegen sich aber weiterhin kontinuierlich.

2.3.1 Bewegung innerhalb des Schwarms

Schwarmverhalten zeichnet sich dadurch aus, dass sich der Schwarm eher passiv bewegt. Beim Vicsek Model **(http://sci-hub.tw/https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.75.1226)** zum Beispiel entsteht eine koordinierte Bewegung durch Wiederholung drei simpler Schritte:

1. Berechne die durchschnittliche Ausrichtung innerhalb deiner Nachbarschaft
2. Passe deine Ausrichtung der berechneten Ausrichtung an (+ Zufallsfaktor bestimmter Größe)
3. Bewege dich um x Einheiten nach vorne

Nach einigen Iterationen dieser drei Schritte stellt sich eine kollektive Bewegung ein und der gesamte Schwarm bewegt sich in die selbe Richtung.

2.3.1.1 Auswirkungen von Rauschen

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2.3.2 Abgrenzung: Abgesprochene Bewegung

In der kollektiven Bewegung mit zentraler Steuereinheit berechnet diese die Positionen der einzelnen Arbeiter und teilt ihnen mit wie sie sich in der nächsten Iteration auszurichten haben. Dadurch ist es leicht möglich eine Gruppe von Einheiten in eine gewollte Richtung zu lenken und die einzelnen Einheiten sowie die gesamte Gruppe dadurch zu steuern. Solch eine 'abgesprochene Bewegung' lässt sich auch dezentral realisieren. Die einzelnen Mitglieder des Schwarms können ihre Daten den anderen mitteilen und in gemeinsamen Absprachen abstimmen, wohin sich der Schwarm bewegen soll und sich dementsprechend ausrichten. Diese Absprachen erfordern viel Kommunikation und richten sich entgegen typischem Schwarmverhalten.

Im typischen Schwarmverhalten ist die Bewegung, vor allem die Ausrichtung der Bewegung, etwas dass sich, wie in Vicseks Modell, passiv durch Beobachtung der Nachbarschaft automatisch synchronisiert. Es gibt keinerlei Absprachen innerhalb des Schwarms wohin sich einzelne Einheiten bewegen sollen oder wohin es mit dem Schwarm im gesamten gehen soll. Entsprechend ist es schwer einen Schwarm in eine bestimmte Richtung zu steuern, da man den einzelnen Einheiten nicht einfach mitteilen kann wohin sie sich ausrichten sollen.

QUELLEN

2.3.3 Steuern eines Schwarms

https://hal.elte.hu/flocking/browser/trunk/public/references/vasarhelyi/Tarcai2011.pdf?format=raw

Möchte man einen Schwarm dazu bringen sich in eine bestimmte Richtung zu bewegen, darf man es ihm nicht kommunizieren, da es das Paradigma des Schwarmverhaltens brechen würde. Um dieses Ziel zu erreichen muss man ihn passiv beeinflussen und ihn dazu bringen seine Ausrichtung selbständig in die gewollte Richtung zu ändern.

Anführer innerhalb eines Schwarms

Eine populäre Möglichkeit einen Schwarm zu lenken ohne direkt mit ihm zu kommunizieren ist der Einsatz von Anführern ('Leader'). Diese Anführer wissen wo der Schwarm sich hinbewegen soll oder haben einen Auftrag und bewegen sich entsprechend diesem. Im laufe der Iterationen richten sich die Einheiten immer am Durchschnitt der Nachbarschaft aus, wobei die Anführer eben dies nicht tun, sondern sich entsprechend der Aufträge ausrichten. Dadurch bildet ihre Ausrichtung eine Art Konstante innerhalb des Schwarm die sich nicht relativ zu den anderen Verändert. Der Schwarm nimmt dadurch allmählich die Ausrichtung dieser Konstanten an und er bewegt sich in die Richtung die von dem Anführer (auch mehrere sind möglich) vorgegeben wird. Mit Hilfe der Technik dieser Anführer, lässt sich der Schwarm letztlich steuern ohne dass das Paradigma des Schwarmverhaltens gebrochen werden muss.

Stigmergie

https://web.archive.org/web/20131104125931/http://www.eecs.harvard.edu/ rad/courses/cs266/papers/alife94.pdf

Eine besondere Form des passiven Nachrichtenaustauschs innerhalb eines Schwarm ist Stigmergie. Stigmergie beruht auf passiven Nachrichten die von Einheiten in der Umgebung platziert und von anderen Einheiten wahrgenommen werden. Ein Beispiel von Stigmergie findet man im Tierreich bei Termiten. Diese rollen ihre Schlamm-Kugeln zusammen, platzieren eine Pheromon-Spur oben drauf und lassen sie anschließend zunächst zufällig irgendwo in der Umgebung liegen. Termiten mögen die Pheromone von anderen Termiten sehr und sind gewillt ihre Kugeln auf denen von anderen zu platzieren, wenn diese sich nicht zu weit entfernt befinden. Je mehr Kugeln sich auf einem Haufen befinden, desto attraktiver ist dieser Haufen dafür die nächste Kugel darauf zu platzieren. Auf diese Weise bilden sich die bekannten Termiten-Hügel die wie Spitzen aus dem Boden ragen.

Diese Methode lässt sich auch auf Roboter übertragen. https://www.mitpressjournals.org/doi/abs/10.1 So gab es bereits erfolgreiche Experimente in denen ein Schwarm von Robotern durch Stigmergie dazu gebracht wurde einen Haufen von Frisbees zu sortieren. Dieses Verhalten ist von Ameisen als "brood sorting"bekannt und verzichtet bei der Arbeit vollkommen auf direkte Kommunikation. Die einzelnen Einheiten reagieren dabei nur auf das Umfeld, welches von anderen Einheiten stetig verändert wird und bilden so nach und nach die Frisbee-Haufen mit der entsprechenden Farbe, ohne sich untereinander absprechen zu müssen.

2.4 Phase Transitions

2.5 ROS

Turtlesim

In dieser wird ein visuell sichtbares Feld von 11.088889 x 11.088889 Einheiten erzeugt, welches auf Koordinaten statt auf Feldern gestützt ist. Somit ist auch eine Bewegung mit reelen Zahlen möglich. Eine Turtle (die Einheit die sich in der Turtlesim bewegt) kann mit Hilfe verschiedener Nachrichten gesteuert werden, sowohl fließende Fahrbewegungen als auch Teleportationen sind möglich. Da die Turtlesim genau weiß wo sich ein Roboter befindet, ist auch die Auswertung der Algorithmen wesentlich einfacher als bei Robotern im realen Einsatz. Die Daten können einfach ausgelesen und life

3 Analyse

Im folgenden Kapitel wird es eine Analyse der Aufgabenstellung geben, darunter die Analyse der Probleme die mit der Aufgabenstellung einher gehen und die Anforderungen des späteren Systems.

Problembeschreibung, potentielle Lösungswege, Entscheidungen wieso welcher Weg gegangen wird, wieso die anderen nicht etc

Analyse des Themenbereichs bzw der bestehenden Probleme und Ermittelung der Anforderungen gemäß passender Methoden

3.1 Problembeschreibung

Laut Aufgabenbeschreibung (Abschnitt 1.2) ist das Ziel dieser Thesis, mit Hilfe von Einheiten, die im Schwarm agieren, Transportaufträge zu verrichten. Als Einschränkung war gegeben, dass die Einheiten nicht erst den Auftrag verteilen müssen, sondern die entsprechenden Einheiten ihn bereits haben und die Verteilung des Auftrags, bzw. die Auswahl der geeignetsten Einheiten somit bereits abgeschlossen ist.

Die nachfolgenden Abschnitte bilden den Verlauf des typischen Transports ab und zeigen die verschiedenen Herausforderungen die es dabei zu bewältigen gibt und die Probleme die dabei auftraten können.

Allgemeine Probleme

In diesem Abschnitt sind die Probleme aufgelistet, die generell in jeder Phase des Transports auftreten können.

Erlaubtes Gelände

Generell ist immer darauf zu achten, dass es Gebiete gibt in denen der Schwarm sich bewegen darf und solche, zu denen er keinen Zutritt hat. Dabei geht es nicht unbedingt darum gefährliche Orte, und damit Beschädigung, zu vermeiden, sondern auch darum dass er nicht überall erwünscht ist. Lässt man den Schwarm Transportaufträge innerhalb eines offenen Geländes erfüllen, besteht die Möglichkeit dass er es auf der Suche nach dem kürzesten Weg kurzerhand verlässt und damit auch Eigentumsgrenzen. Andererseits kann es aber auch Orte geben die beispielsweise besonders sauber bleiben sollten und in denen die Einheiten mit ihren (eventuell) dreckigen Rädern nicht erwünscht ist. Idealerweise lassen sich solche Bereiche dynamisch in die Gebietskarte einzeichnen oder vor Ort markieren, sodass auch auf Gebiete geachtet werden kann die erst kürzlich zu einer Gefahrenzone wurden, wie Beispielsweise eine ausgelaufene Flüssigkeit.

Gruppe finden

Die neue Gruppe die für den Auftrag zuständig ist muss sich selbst finden. Die Vergabe des Auftrags schließt nicht mit ein den Einheiten Erklärungen mitzugeben wie sich zu arbeiten haben, sondern lediglich die Daten für den Auftrag, darunter wie die Ware aussieht, wo sich der Aufnahmeort befindet und wo der Abgabeort. Nachdem eine Einheit also festgestellt hat dass er einen neuen Auftrag besitzt, müssen die anderen Gruppenmitglieder innerhalb des Netzwerks gefunden werden um direkt und später Informationen untereinander auszutauschen.

Standby im Schwarm

Ohne Auftrag gehen die Einheiten ihrem typischen Schwarmverhalten nach. Praktisch heißt das, dass sie sich bewegen und dabei die 4 Regeln von Schwarmverhalten (Abschnitt 2.3) beachten. Die ständige Bewegung mag aus energietechnischer Sicht nicht die optimale Lösung darstellen, da ein Schwarm selbst aber nie still steht, bildet sie aus Sicht eines Schwarms letztlich die Standard-Lösung für 'warten'. Wie in Abschnitt 2.3.1.1 beschrieben, spielt das Rauschen bei der Bewegung eines Schwarm einen wichtigen Faktor. Ist der Schwarm im Standby, hat also keinen Auftrag, muss der Faktor des Rauschens so gewählt werden, dass die einzelnen Einheiten des Schwarm zwar noch in Bewegung bleiben, der Schwarm als großes ganzes aber keine Bewegung zu erkennen gibt und augenscheinlich auf der Stelle steht.

Dichte des Schwarms

Da sich der Schwarm auf einem Gelände, wahrscheinlich sogar innerhalb eines Gebäudes oder einer Fabrikhalle, befinden wird, muss auch darauf geachtet werden, dass der Schwarm während des Wartens kein Hindernis für andere Beschäftigte oder Fahrzeuge darstellt. Er sollte im Hintergrund verschwinden und kein aktiver Teil der Umgebung werden auf den im besonderen Maße geachtet werden muss. Der Schwarm sollte sich daher nirgendwo (partiell) örtlich zusammenballen, sondern, wenn man ihn sich als einen Festkörper vorstellt, seine Dichte so gering wie möglich werden lassen, indem er sich auf dem (ihm erlaubten) Gelände so gut es geht ausbreitet.

Bildung der Untergruppe

Es muss beachtet werden, dass die neue Untergruppe nach wie vor Teil des großen Schwarms ist und alle Regeln des großen Schwarms erbt, aber auch neue dazu bekommt. Die neuen Regeln sorgen dafür, dass der Auftrag ausgeführt werden kann, dürfen sich aber nicht mit den 4 Hauptregeln (siehe Abschnitt 2.3) stören.

Zusammenfinden am Aufnahmeort

Nachdem die Einheiten ihren Auftrag erhalten haben, gilt es sich am Aufnahmeort zusammenzufinden. Die Einheiten waren gerade noch in Standby ein Teil des Schwarms und müssen ab sofort eine Untergruppe innerhalb des Schwarms bilden die für den Auftrag zuständig ist.

Als Untergruppe müssen sie sich dann am Aufnahmeort zusammenfinden. Das Problem in dieser Situation ist, dass die Untergruppe nach wie vor als Schwarm agieren muss. In Abschnitt 2.3.2 wurde bereits erläutert, dass Schwarmverhalten nicht mit 'Abgesprochener Bewegung' verwechselt werden darf und Einheiten innerhalb eines Schwarms

sich nicht aktiv darüber absprechen, welche Einheit sich wohin bewegen soll. Das bedeutet, dass die Untergruppe sich ausschließlich durch passive Beeinflussung und die Informationen des gegebenen Auftrags gemeinsam am Aufnahmeort zusammenfinden muss. Natürlich darf dabei aber nicht davon ausgegangen werden, dass die Gruppe die den Auftrag erhält auch eine erkennbare Gruppe war und die Einheiten daraus örtlich nahe zusammen waren, sondern es muss davon ausgegangen werden dass die Einheiten zufällig ausgewählt werden können.

Einnehmen einer Formation

Damit die Ware sicher gelagert ist, muss die Untergruppe eine Formation einnehmen, die für den Transport der Ware geeignet ist. Beispiele für solche Formationen wären z.B. ein Viereck, wenn eine große viereckige Kiste transportiert werden muss oder eine Linie, sollte ein Stahlträger transportiert werden müssen. Wichtig für die Formation ist jedoch nicht nur die Form der Ware, sondern auch die Anzahl der Einheiten die für den Transport zugeteilt wurden. So macht es bei einer Kiste für vier Einheiten am meisten Sinn, wenn sich die Einheiten an den vier Ecken/Kanten verteilen. Eine fünfte Einheiten würde zunächst in der Mitte am meisten Sinn machen, wohingegen es bei sechs Einheiten wieder mehr Sinn machen würde, wenn der Einheiten aus der Mitte zu einer Ecke/Kante wandert die noch nicht belegt ist und der sechste dann gegenüber.

Das Einnehmen einer Formation ist für einen Schwarm eine besondere Herausforderung, weil Schwarmverhalten sich eben signifikant von abgesprochener Bewegung (siehe Abschnitt 2.3.2) unterscheidet. Ein schneller erster Gedanke wie Einheiten eine Formation einnehmen, wäre die Kontur der Ware zu zeichnen und die Einheiten an den signifikanten Stellen, möglichst symmetrisch, zu verteilen. Für solch eine Koordination braucht es aber letztlich auch einen Koordinator. Einen Koordinator zu wählen, wäre für die Untergruppe grundsätzlich kein Problem, ein einfacher Echo-Algorithmus wäre vollkommen ausreichend, jedoch gilt es abgesprochene Bewegung von Schwarmverhalten abzugrenzen. Es muss für dieses Problem also ein Algorithmus gefunden werden, mit dessen Hilfe sich ein Schwarm in eine bestimmte Formation bringen lässt, ohne dass es einen Koordinator gibt der den einzelnen Einheiten mitteilt wo sie sich hinbewegen müssen. Grundlage für diese Formation muss ein simpler Austausch von Informationen sein, darunter vor allem die Größe der Gruppe.

Stillstand am Aufnahmeort

Am Aufnahmeort angekommen, müssen die Einheiten still stehen bleiben. Der Grund dafür ist, dass es eine gewisse Zeit dauert bis ein Mitarbeiter oder eine andere Maschine die zu transportierende Ware auf die Einheiten verladen hat. In dieser Zeit darf die Untergruppe nicht in den typischen Standby des großen Schwarms verfallen, denn dabei würden sich die Einheiten leicht bewegen und das verladen schwierig machen. Statt dessen müssen sie eher in eine Art Starre verfallen bis ein bestimmtes Ereignet eintritt, welches ihnen signalisiert dass die Ware vollständig verladen wurde und sie ihren Weg antreten können.

Gemeinsamer Transport zum Abgabeort

Nachdem die Ware erfolgreich auf die Einheiten verladen wurde gilt es nun diese an ihren Abgabeort zu transportieren und den Transportauftrag dadurch zu beenden.

Halten der Formation

Wichtig beim Transport der Ware ist, dass die Formation möglichst genau beibehalten werden muss. Ein abdriften der Formation könnte schnell dazu führen, dass die zu transportierende Ware herunterfällt und beschädigt wird oder dabei andere Gegenstände, insbesondere die Einheiten selbst, beschädigt werden. Ebenso ist eine Beschädigung der Einheiten oder der Ware möglich, wenn die Ware rutschfest auf den Einheiten liegt oder gar festgeschraubt wurde. In diesem Fall könnte das verlassen der Formation dazu führen, dass die Einheiten versuchen in eine Richtung zu fahren, aber von der Befestigung daran gehindert werden. Die Ware oder die Einheiten könnten durch die Zugkräfte beschädigt werden, insbesondere könnten die Einheiten umfallen oder die Motoren die für den Antrieb zuständig sind überhitzen und die Einheit ausfallen lassen.

Wieder ist darauf zu achten, dass die Formation ohne Koordinator eingehalten werden muss, allein durch Regeln die der Untergruppe als Schwarm allgemein inne herrschen. Die gleichen Algorithmen die dazu führten dass die Untergruppe am Aufnahmeort ihre Formation einnahm müssen nun dazu verwendet werden um die Formation während des Transports zu halten.

Finden des Weges

TODO

Standby am Abgabeort

Am Abgabeort angekommen muss die Untergruppe wieder zum Stillstand kommen und solange in einer Art Starre verharren bis die Ware vollständig umgeladen wurde. Die Starre muss erneut anhalten bis ein bestimmtes Event eintritt.

Anschließend muss die Untergruppe wieder in den Schwarm eingegliedert werden. Das heißt, die neuen Regeln die für die Erfüllung des Auftrags notwendig waren fallen nun weg und es werden wieder die Regeln des großen Schwarms übernommen, wie zum Beispiel das Verteilen um anderen Arbeitern nicht im Weg zu stehen.

3.2 Anforderungen an das System

Nachdem im vorigen Abschnitt Herausforderungen anhand des generellen Verhaltens der Einheiten dargestellt wurden, wird nun auf die Anforderungen des Systems selbst eingegangen.

Unabhängigkeit

Die einzelnen Einheiten des Schwarms oder der Untergruppen die gerade einen Auftrag erledigen müssen vollständig unabhängig agieren und dürfen nicht aktiv von einer anderen Einheit geleitet werden. Dies betrifft auch eventuelle Anführer die innerhalb der Untergruppen gewählt werden. Diese mögen zwar eine Sonderrolle einnehmen, dürfen den anderen Einheiten der Untergruppe aber nur passiv beeinflussen und ihnen nicht explizit mitteilen wo sie sich hinzubewegen haben.

Ausfallsicherheit

Einer der besonderen Vorteile des Schwarms ist es, dass die Einheiten generell unabhängig agieren und nicht auf die Kommunikation mit anderen Einheiten angewiesen sind. Dar-

aus resultiert dass ausgefallene Einheiten kein generelles Problem für den Schwarm sind. Betrachtet man einen Schwarm im allgemeinen, so ist eine ausgefallene Einheit vergleichbar mit einem Stein der herumliegt; er kommuniziert nicht und er bewegt sich auch nicht mehr. Er spielt also im Schwarm keine große Rolle mehr und dieser kann ungehindert weiter agieren, nur eben mit einer Einheit weniger.

Im besonderen Fall des Auftrags sind die Einheiten allerdings im gewissen Maße abhängig voneinander. Gerade wenn es um die Bildung der Formation geht, spielt die Größe der Untergruppe eine wichtige Rolle, da sie mitunter bestimmt wo sich die einzelnen Einheiten aufzuhalten haben. Sollen 2 Einheiten eine Stange bewegen und eine fällt unterwegs aus, fällt die Ware aufgrund des fehlenden Gleichgewichts zu Boden. Es muss daher darauf geachtet werden, dass die Untergruppe in ständigem Kontakt untereinander steht um sicherzugehen dass alle Einheiten nach wie vor aktiv und einsatzbereit sind. Fällt plötzlich eine Einheit aus, muss darauf reagiert werden indem die Formation entsprechend angepasst wird oder der Auftrag abgebrochen und eine Fehlernachricht an ein zuständiges System gesendet wird.

Örtlichkeit

Ein Schwarm zeichnet sich nicht nur dadurch aus, dass die Einheiten unabhängig voneinander sind, sondern auch dadurch, dass Nachrichten zur Kommunikation nicht den gesamten Schwarm belasten. Sendet eine Einheit eine Nachricht an seine Nachbarschaft (siehe Abschnitt 2.1) aus, so darf diese nicht den gesamten Schwarm durchqueren, sondern muss mit der Zeit 'kleiner' werden und letztlich verschwinden. Dies hat nicht nur den Grund näher am Vorbild der Natur zu sein **QUELLE** sondern ist auch deshalb notwendig, da die Nachrichten von mehreren hundert Einheiten sonst das Netzwerk lahmlegen würden und die Einheiten nur noch damit beschäftigt wären die Nachrichten zu verarbeiten und weiterzuleiten. Die Ressourcen für die eigentliche Aufgaben würden fehlen und der Schwarm würde still stehen.

4 Konzeption

In diesem Kapitel werde ich die Konzeption hinter den Schwarmrobotern genauer beschreiben. Dazu gehört die Struktur des Systems generell, aber auch wichtige Algorithmen die die Roboter selbst, aber auch deren Interaktion mit anderen beschreiben.

4.1 Generelles zur Konzeption

Die Konzeption dieser Thesis beruht darauf, dass ROS (Robot Operating System) genutzt wird. ROS ist ein Framework, dass die Erstellung von Robotern erleichtern soll und arbeitet mit Nodes, ähnlich Microservices. Diese Nodes können als Publisher fungieren, die Nachrichten an ein Topic senden und welche man dann abonnieren kann. Sie können aber auch als Services dienen, denen man Daten sendet und von denen man anschließend eine Antwort erwarten kann.

Die Konzeption ist in 3 Phasen unterteilt, die das Konzept zum letztlichen Ziel führen werden. Der Sinn hinter diesen 3 Phasen, ist eine Iteration. Die anderen Kapitel Implementierung und Evaluation werden parallel geführt werden und nach jeder Phase der Konzeption wird das System zuerst prototypisch gebaut und anschließend evaluiert. Dadurch kann die Konzeption in seinen weiteren Phasen die Ergebnisse der vorangegangenen eingehen auf den neuen Kenntnissen aufbauen. Außerdem wird dadurch sichergestellt, dass das Verhalten der Roboter nicht zu sehr auf die Aufgabe des Transports zugeschnitten wird. Das ist deswegen der Fall, weil das zugrundeliegende System ein natürliches Schwarmverhalten ähnlich dem von Tieren sein soll und keine Gruppe von Robotern die 'übernatürliche' Fähigkeiten haben.

4.2 Nachbau des Schwarms nach Craig Reynolds

Da der Transport des Schwarms auf den 3 (bzw. 4) Grundregeln des Schwarmverhaltens besteht, welche von Craig Reynolds erstmals modelliert wurden (**http://www.red3d.com/cwr/boids/**), galt es diese zunächst nachzustellen und zu analysieren. Der Schwarm wurde zunächst nachgebaut, ohne auf die spätere Verwendung für Transporte zu achten, da diese (möglichst) ohne Kompromisse auf dem Schwarm-Prinzip aufbauen sollten.

Ziel

Das Ziel der ersten Phase ist es grundsätzliches Schwarmverhalten nachzubauen und dieses zu messen. Es werden individuelle Roboter erstellt, die sich nur dadurch bewegen, indem sie sich an anderen Einheiten orientieren und dabei noch ihren freien Willen in die Entscheidung der Bewegungsrichtung einfließen lassen. Anschließend wird gemessen werden, auf welche Weise die verschiedenen Parameter das Verhalten der Roboter beeinflussen.

Messages

Die Roboter brauchen einige Informationen über die anderen Roboter, darunter vor allem die Positionen und Ausrichtungen derer die innerhalb einer bestimmten Reichweite liegen. Diese Informationen könnten über Sensoren kommen. Da diese Sensoren aber ungenau und kostspielig sind, wird davon ausgegangen, dass sie in der Praxis nicht zur Verfügung stehen. Da die Roboter ihre eigene Position stets kennen müssen, werden diese Informationen direkt mit den anderen Robotern geteilt werden. In ROS funktioniert dies über einen ROS-Master der im Sinne des Subscriber-Publisher-Prinzips Nachrichten zu einem bestimmten Topic entgegen nimmt und sie an diejenigen weiterverteilt, die das entsprechende Topic abonniert haben. Der ROS-Master ist dabei zwar eine zentrale Node, kann aber auf dem System eines Roboters untergebracht werden. Es ist also keine weitere Einheit notwendig, wodurch das Gesamt-System homogener bleibt, wenn auch immer eine direkte Verbindung zum Master bestehen bleiben muss. Es gibt Bemühungen ROS für mehrere, parallel laufende, Master-Nodes zu öffnen, bisher sind die Packages aber noch in der Entwicklung und es muss auf einem zentralen Master aufgebaut werden[1]. Jedem Roboter wird zudem eine eindeutige ID zugewiesen werden, die es erlaubt sie unterscheiden zu können.

Für eine einfache Schwarm-Simulation, ist nur eine Art von Nachricht notwendig. Diese enthält die Position und Ausrichtung der Roboter und wird nach jedem ausgeführten Bewegungs-Befehl gesendet. Um das System der Roboter einfacher zu halten, abonnieren die Roboter auch die Bewegungs-Daten von sich selbst. Dadurch ist es nicht notwendig diese gesondert zu behalten, sondern die Daten werden in das eigene System eingepflegt, wie die von allen anderen Einheiten auch. Möchte ein Roboter dann auf die eigenen Daten zugreifen, kennt er seinen Index und greift auf die Daten normal zu.

Der bisher einzig genutzte Nachrichten-Typ beinhaltet somit die Positionsdaten der einzelnen Roboter. Für die Deklaration der Nachrichten-Typen werden die Datentypen genutzt die ROS beinhaltet[2]. Dies erleichtert die Implementierung und die Namen der Datentypen sind außerdem selbsterklärend.

Nachrichten-Typ: Robot_Position

```
uint8 robot_id  // Zur Identifizierung des Absenders
float32 pos_x   // Position des Roboters entlang der X-Achse
float32 pos_y   // Position des Roboters entlang der Y-Achse
float32 angle   // Ausrichtung des Roboters im Winkel von [0, 360] Grad
```

Diese Art von Nachricht macht auch den Großteil der gesendeten Nachrichten aus. Da das Verhalten der Roboter überwiegens von den Positionen anderer gesteuert wird, sollten dieser Nachrichten-Typ so oft wie möglich gesendet werden. Da eine Nachricht dieser Art aber auch von allen Robotern empfangen wird, beträgt die Anzahl der Nachrichten die bei der Aktualisierung des gesamten Schwarms durch das Netzwerk gehen `(n+1)*n`. Um das Netzwerk nicht zu überfordern muss deshalb ein Mittelwert gefunden werden. Zumindest sollte die Aktualisierungsrate aber hoch genug sein, damit sich die Roboter nicht gegenseitig umfahren.

Einhalten der 4 Grundregeln

Die Roboter werden als autonome Einheiten konzipiert, die sich ausschließlich anhand von 3 fixen Parametern bewegen und dabei versuchen die 4 Grundregeln des Schwarm-

[1] `http://wiki.ros.org/multimaster_fkie`
[2] `http://wiki.ros.org/msg`

verhaltens einzuhalten.

Ausrichtung: Passe deine Bewegungsrichtung deinen Nachbarn an

Da ein Roboter die Position und Ausrichtung jedes anderen Roboters kennt, ist es ohne weiteres möglich die durchschnittliche Ausrichtung der Nachbarschaft zu ermitteln. Dafür wird die Liste der Roboter genommen und der Roboter der Nahe genug ist um relevant zu sein in eine neue Liste kopiert. Von diesen Robotern wird dann der Durchschnitt des Winkels berechnet. Der einfache Durchschnitt, über die Summe mehrerer Winkel, würde jedoch zu einem mathematischen Schnitt führen, statt zu einem, der in der Praxis relevant ist. Daher müssen die Winkel für die Berechnung zunächst in Vektoren umgerechnet werden. Diese Vektoren werden dann addiert und der resultierende Gesamtvektor wird wieder in einen Winkel zurück gerechnet.

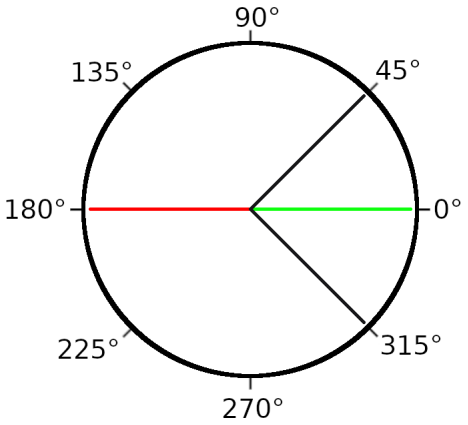


Abbildung 4.1: Winkel-Übersicht

Beispiel: Fehlerhafte Winkel-Berechnung Abbildung 4.1 zeigt ein Beispiel für einen Fehler, beim Berechnen des Durchschnitts zweier Winkel. Zwei Roboter mit den Winkeln 45° und 315° würden zu einem durchschnittlichen Winkel von 180° führen. Der richtige Winkel in Anbetracht der Ausrichtung wäre jedoch 0°.

Zusammenhang: Versuche deinen Nachbarn nahe zu sein

Dieser Abschnitt wird realisiert indem ebenfalls zuerst die Roboter in eine Liste kopiert werden, die nahe genug sind um zur Nachbarschaft zu gehören. Anschließend wird ein simpler Durschnitt über alle Positionen (x/y - Koordinaten), der vorhandenen Roboter berechnet. Die resultierenden Koordinaten werden von den eigenen Koordinaten abgezogen, um die relative Position des Schwarm-Mittelpunkts zu bekommen. Indem man nun die relativen Koordinaten als Vektor behandelt und dadurch zu einem Winkel umrechnet, erhält man letztlich den relativen Winkel, den der Mittelpunkt der Nachbarschaft zum Roboter hat. Dieser Winkel wird mit normaler Geschwindigkeit angefahren.

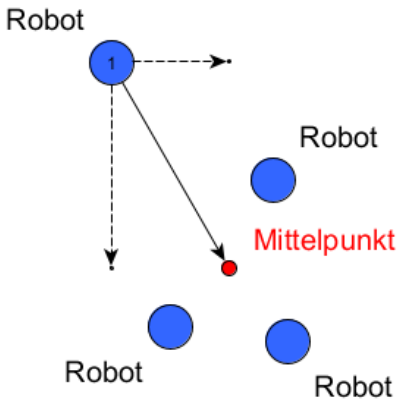


Abbildung 4.2: Berechnung des Mittelpunkts

Dabei ist der Mittelpunkt der Nachbarschaft wie er in Abbildung 4.2 zu sehen ist, allerdings nicht zwingend der Mittelpunkt der konvexen Hülle die die Nachbarschaft umgrenzt. Mehrere Roboter ziehen den Mittelpunkt der Nachbarschaft mehr in ihre Richtung. Der Vorteil dieser Methode ist schlicht, dass es den Roboter eher zu Gruppen zieht anderer Roboter zieht. Würde es den Roboter zur Mitte der konvexen Hülle ziehen, wäre die Gefahr groß, dass er alle Roboter verlieren würde, da sie zu allen Seiten gleich weit entfernt sind. Durch die einfache Methode des Mittelwerts bleibt der

Roboter stets näher an Gruppen und lässt notfalls einzelne Roboter ziehen um die Gruppe zu behalten.

Abschottung: Vermeide Kollisionen mit deinen Nachbarn

Damit die Roboter nicht beschädigt werden, müssen sie eine Größe erhalten und ihre Position mit der der anderen abprüfen. Eine Bewegung wird als gültig befunden, wenn der Roboter an seinem Zielpunkt mindestens seine Größe als Abstand zu allen anderen Robotern hat. Praktisch wurde die Größe um 10% erweitert, um eine gewisse Fehlertoleranz zu erlauben. Auch die Frequenz der in Abschnitt 4.2 erwähnten Positions-Nachrichten spielt hier wieder eine Rolle. 2 Roboter können dann aufeinander zufahren, wenn der Abstand beider Roboter nicht innerhalb der nächsten Aktualisierung der Positionsdaten überwunden werden kann.

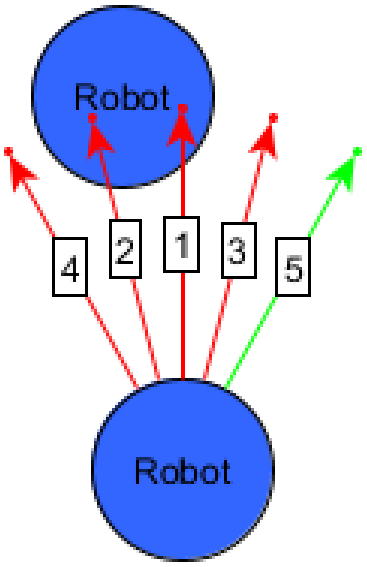


Abbildung 4.3: Ausweichen gegenüber Hindernissen

Ob eine Kollision am Ziel (oder unterwegs) stattfinden wird, wird mittels interner Simulationen überprüft, die die letzten Positionsdaten der Roboter nutzt. Findet eine Kollision statt, prüft der Roboter Ausweichmöglichkeiten. Der Algorithmus wird dabei angewendet, egal ob das Hinderniss ein anderer Roboter ist oder eine Gefahrenzone. Ist der Weg geradeaus nicht möglich, wird abwechselnd versucht nach links und rechts auszuweichen, wie in Abbildung 4.3 gezeigt wird. Dabei wird der Winkel, der zum Ausweichen genutzt, wird stufenweise erhöht, bis man letztlich 180° zu beiden Seiten erreicht hat, was einem Schritt rückwärts entspricht. Führt auch dieser letzte Winkel nicht zu einem erfolgreichen Ergebnis, wird der Algorithmus erneut mit einer leicht geringeren Geschwindigkeit (was ebenfalls einer leicht geringeren Bewegungsdistanz entspricht) gestartet. Erst wenn eine Simulation eine erfolgreiche Bewegung generieren konnte, wird diese auch ausgeführt.

Flucht: Fliehe vor Dingen, die eine potentielle Gefahr darstellen

Gefahren wie Hindernisse, Löcher im Boden oder Zonen mit sich bewegenden Maschinen können die Roboter beschädigen, weshalb es gilt diesen Auszuweichen. Bei jeder Bewegung muss geprüft werden, ob man mit der angestrebten Bewegung eine Gefahrenzone betreten würde. Wenn ja, wird versucht mit dem Algorithmus des vorigen Abschnitts auszuweichen.

Um Gefahrenzonen abbilden zu können, werden Objekte erzeugt die sich aus verschiedenen Geometrien, mitsamt Positionen, zusammensetzen. Betritt ein Roboter eine der, im Objekt inneliegenden Geometrien, ist das Objekt als ganzes betreten worden und der Roboter befindet sich in einer Gefahrenzone. Um ein Objekt zu definieren, ließen sich auch andere Methoden nutzen. Punkte die die Zone umspannen und eine konvexe Hülle bilden oder der Reihenfolge nach mit Geraden verbunden werden, wären ebenfalls eine Option. Allerdings wären diese viel komplexer zu berechnen und es würde unnötig Rechenleistung verbrauchen. Die Berechnung, ob sich ein geometrischen Objekt innerhalb eines anderen geometrischen Objektes befindet, ist außerdem ein häufiges und gut

gelöstes Problem von Videospielen. [3]

Parameter der Bewegung

Konfiguriert wird das Schwarmverhalten der Roboter durch 4 Parameter, die in den nachfolgenden Abschnitten näher erläutert werden.

Geschwindigkeit Gibt an, wie schnell ein Roboter sich bewegt. Ein kleinerer Wert sorgt für langsamere, aber auch 'vorsichtigere' Roboter. Optimal wäre ein kleiner Wert in Geschwindigkeit, aber eine hohe Frequenz an Bewegungs-Befehlen.

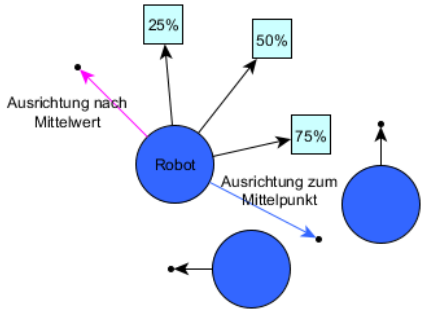


Abbildung 4.4: Berechnung des Drangs zur Gruppierung

Lokale Reichweite Da ein Roboter nur seine unmittelbare Umgebung imitiert, braucht es dafür einen Grenzwert. In dieser Thesis wurde sich für eine Begrenzung in der Reichweite (im Gegensatz zu einer Begrenzung in der Anzahl der Roboter) entschieden, da die unmittelbare Umgebung wichtiger ist, als eine feste Anzahl an Robotern an denen sich orientiert wird. Einheiten die weiter weg sind als dieser Wert (in der entsprechenden Maßeinheit), werden für Berechnungen ignoriert. Ein Wert kleiner als die eigene physikalische Größe, führt zwangsläufig dazu, dass der Roboter nur sich selbst beachtet.

Freier Wille Einheiten im Schwarm orientieren sich nicht nur an seinen Nachbarn, sondern haben auch in gewisser Weise einen eigenen Willen. Nachdem die Berechnung der Orientierung am Schwarm abgeschlossen ist wird der eigene Wille einberechnet. Das ist ein Winkel der zufällig zwischen [-x/2, x/2] ausgesucht wird und dem bisherigen Winkel aufgerechnet wird.

Nähe zum Schwarm Dieser Wert ist verbunden mit der Regel 'Zusammenhang: Versuche deinen Nachbarn Nahe zu sein'. Ein Roboter hat den Drang seinem Schwarm nahe zu bleiben und versucht sich, bis zu einem gewissen Grad, in dessen Richtung zu bewegen. Da der Drang zum Schwarm nahe zu bleiben als Teil des freien Willens gezählt wird, wird dieser Wert prozentual angegeben und nach dem freien Willen auf den endgültigen Winkel aufgerechnet. Wie genau der Drang zur Gruppierung in Prozente eingeteilt ist, wird in Abbildung 4.4 skizziert dargestellt.

Beispiel: Berechnung einer Bewegung

In Abbildung 4.5 ist die Berechnung einer Bewegung skizziert. Die Pfeile geben dabei nur Richtungen an und sind nicht als Vektoren zu verstehen, die eine Aussage über die bewegte Entfernung treffen.

Der pinke Pfeil ist das Resultat der Berechnung des Mittelwerts der Winkel, den die anderen Roboter beim letzten Update ihrer Position haben.

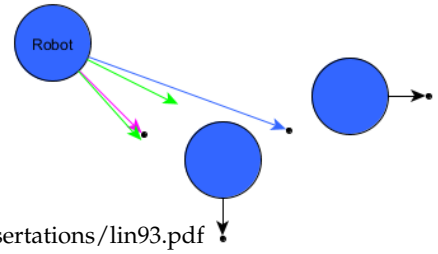


Abbildung 4.5: Berechnung einer Bewegung

[3] urlhttps://wwwx.cs.unc.edu/ geom/papers/documents/dissertations/lin93.pdf

Der blaue Pfeil ist der Mittelpunkt des Schwarms (in diesem Fall sind beide Roboter ein Teil der Nachbarschaft), auf den der Roboter sich aufgrund seiner Gruppenzugehörigkeit zubewegen möchte.

Die beiden grünen Pfeile sind letztlich die obere und untere Grenze des Spielraums, den der Roboter durch seinen eigenen freien Willen bekommt. Dieser wurde durch die Ausrichtung zum Schwarm-Mittelpunkt zusätzlich leicht verschoben. Aus diesen beiden Grenzen wird letztlich ein Winkel zufällig bestimmt und anschließend auf Gültigkeit (fehlende Kollisionen) geprüft um dann schlussendlich ausgeführt zu werden.

4.3 Anführer

Um später Transportaufträge erledigen zu können, muss der Schwarm in irgendeiner Form gesteuert werden können. Bei einer direkten Steuerung aller Einheiten des Schwarms, wäre das Schwarmverhalten allerdings außer Kraft gesetzt und man hätte schlicht eine Menge gelenkter Roboter. Deswegen wurden Mittel gesucht den Schwarm indirekt zu lenken und fand diese Methode in einem Paper **QUELLE** in dem beschrieben wurde, wie ein Schwarm über 'eingeweihte Anführer' gesteuert werden kann.

Der Anführer bietet eine passive Möglichkeit einen Schwarm zu lenken, ohne dass die normalen Einheiten angepasst werden müssen oder in ihrem Verhalten speziell abweichen müssen. Die normalen Einheiten müssen dadurch nicht wissen, dass sie ein bestimmtes Ziel haben, es muss ihnen nicht einmal bewusst sein, dass sie passiv gesteuert werden. Dadurch, dass er stets in die Richtung des Ziels zeigt, aber auch aufgrund des Gruppendrangs durch die Nachbarn, pendeln sich die anderen Einheiten des Schwarms, langsam auf das Ziel ein. Ein Anführer ist somit eine Art Leuchtturm, der permanent eine Richtung angibt ohne sich von der Umwelt beeinflussen zu lassen und der den anderen Robotern eine Orientierung gibt.

Als Anführer versucht die entsprechende Einheit außerdem den Kontakt zu seiner Herde nicht zu verlieren. Damit ist nicht nur der Drang gemeint die Gruppenmitte aufzusuchen. Es ist eine bewusstere Art auf den eigenen Schwarm acht zu geben.

Ziel

Ziel dieser Phase ist es, den Schwarm von einem Ort zu einem anderen zu lenken, ohne ihn wissen zu lassen dass er gelenkt wird. Dies soll mit Hilfe von (möglichst wenigen) Anführer geschehen die über das Ziel Bescheid wissen. Auf diese Weise muss nur wenig in das ursprüngliche Verhalten der Roboter eingegriffen werden und der Schwarm bleibt möglichst natürlich.

Einbindung des Anführers in ROS

Der Anführer wurde umgesetzt, indem einer normalen Einheit ein Ziel gegeben wird und er damit beginnt seinen Schwarm an einen bestimmten Ort zu lenken. Dazu werden die Roboter ein neues Topic abonnieren. Über dieses kann eine einfache Nachricht an einen bestimmten Roboter versendet werden. Dieser Roboter wird zum Anführer und hat versucht seinen Schwarm an die entsprechenden Koordinaten zu lenken.

Nachrichten

Um den Transport einzuleiten wird es einen neuen Nachrichten-Typ geben. Die Nachricht kann an den Schwarm rausgesendet werden und durch die eindeutige ID weiß der entsprechende Roboter, dass er als Anführer ausgewählt wurde.

Nachrichten-Typ: New_Mission

```
uint8 leader_id // Die ID des ausgewaehlten Anfuehrers
float32 pos_x   // Position des Ziels entlang der X-Achse
float32 pos_y   // Position des Ziels entlang der Y-Achse
```

Generelles Verhalten

Er steuert direkt auf das Ziel zu und wartet gegebenenfalls, wenn er sich zu weit vom eigenen Schwarm entfernt. Ab einer Entfernung von 75% , der lokalen Reichweite zum Mittelpunkt seiner Herde, bleibt er stehen, bis er wieder bei 50% Entfernung angekommen ist. Anschließend setzt er seinen Weg mormal fort. Ausweichmanöver würden dafür sorgen, dass der Anführer nicht mehr auf das Ziel zeigt und damit auch die anderen Roboter dazu bringen der neuen Ausrichtung zu folgen. Aus diesem Grund wird anderen Robotern nicht ausgewichen, wie es für Schwarmroboter üblich ist. Stattdessen wird der Anführer seine Geschwindigkeit verlangsamen, wenn er mit der normalen Geschwindigkeit eine Kollision verursachen würde. Kann er sich, trotz geringerer Geschwindigkeit nicht bewegen, bleibt er stehen und wartet eine kleine Zeitspanne ab.

Einfangen eines verlorenen Schwarms

Sollte der Schwarm den Grenzbereich der lokalen Reichweite ganz verlassen, wäre es außerdem möglich, den Anführer seinen Schwarm wieder einfangen zu lassen. Dies ist in Abbildung 4.7 dargestellt. Im ersten Abschnitt ist der Schwarm verloren gegangen, obwohl der Anführer gewartet hat. Anschließend ist er mit erhöhter Geschwindigkeit hinter den Schwarm gefahren, sodass der Schwarm sich nun genau zwischen Ziel und Anführer befindet. Nun nimmt er wieder sein normales Verhalten an und versucht den Schwarm in die Richtung des Ziel zu lenken. Beim umdrehen, um den Schwarm wieder einzufangen, dreht sich der Anführer allerdings in die entgegengesetzte Richtung. Die anderen Roboter wissen nichts von einem Manöver und orientieren sich normal am Anführer. Das kann dazu führen, dass der Schwarm zunächst noch weiter abgelenkt wird. Aus diesem Grund sollte das Einfangen so schnell wie möglich beendet werden. Die Geschwindigkeit des Anführers sollte also möglichst hoch sein.

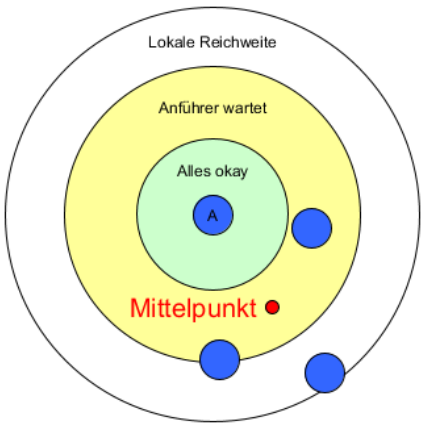


Abbildung 4.6: Entfernungen des Anführers

4.4 Transport von Waren mit Hilfe eines Schwarms

Das letztliche Ziel dieser Arbeit ist es zu prüfen, ob es möglich, und sinnvoll, ist Gegenstände mit Hilfe eines autonomen Schwarms zu transportieren. Dazu musste der Schwarm nun dazu gebracht werden Waren zu bewegen, ohne die einzelnen Roboter zu sehr zu beeinflussen. Da generell die Roboter nicht zentral gesteuert werden sollen und auch die Logik möglichst simple bleiben soll, sollte das Programm der einzelnen Einheiten möglichst wenig verändert werden und der 'natürliche' Trieb der Einheiten genutzt werden.

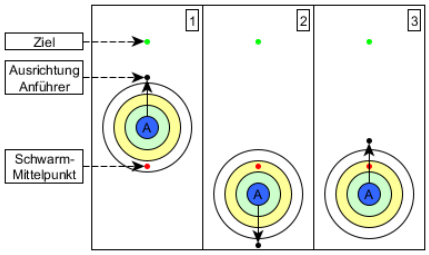


Abbildung 4.7: Anführer fängt seinen Schwarm wieder ein

Erteilen von Aufträgen

Eine der ersten Dinge im Ablauf eines Transports ist die Erteilung des Auftrags. Dazu muss der Nachrichten-Typ `New_Mission` im ROS-Netzwerk erweitert werden, sodass er nun die folgenden Felder hat:

```
uint8 robot_index_from    // Beginn der ID-Range
uint8 robot_index_to    // Ende der ID-Range

uint8 leader_number     // Anzahl der Leader die gebraucht werden
uint8 mission_id      // Die ID der Mission

float32 object_position_x // Die Start-Position des Transport-Objektes, X-Achse
float32 object_position_y // Die Start-Position des Transport-Objektes, Y-Achse

float32 object_size_x   // Die Laenge des Objektes, in X-Richtung
float32 object_size_y   // Die Laenge des Objektes, in Y-Richtung

float32 target_x      // Die Ziel-Position des Transport-Objektes, X-Achse
float32 target_y      // Die Ziel-Position des Transport-Objektes, Y-Achse
```

Der Nachrichten-Typ definiert eine Spanne von Robotern die den Auftrag ausführen sollen. Diese werden mit ihren IDs angesprochen. Das Intervall der IDs ist, gemäß Programmierstandards, als halboffenes Intervall definiert: [robot_index_from, robot_index_to[. leader_number gibt die Anzahl der Leader an die verwendet werden sollen und mission_id gibt dem derzeitigen Auftrag eine fixe ID um diesen und zugehörige Dinge genau identifizieren und verbinden zu können. Mit object_position_x/-_y ist die Startposition des zu transportierenden Objektes angegeben. Zusammen mit object_size_x/-_y, welche die Ausmaße des Objektes angeben. Die Position des Objekts ist als Mitte des Objekts definiert. target_x/-_y gibt die letztliche Position an die das Objekt einnehmen soll.

Der Auftrag wird von außen an das Topic 'flock/mission/new' gesendet. Der Ersteller des Auftrags muss keine ROS-Node sein, auch wenn dies verschiedene Vorteile hätte. Dadurch das ROS-Topics auch über das Terminal engesprochen werden können, ist das Senden der Nachrichten auch über jedes andere Programm möglich. Das Topic für die neuen Missionen wird von jedem aktiven Roboter abonniert. Entsprechend nimmt jeder Roboter Notiz von diesem Auftrag, auch wenn er nicht direkt mit seiner ID angesprochen wird.

Von Gefahrenzonen zu Sicherheitszonen

Um die betreffenden Roboter in die Zone zu senden, auf der ihnen später das Transportobjekt aufgesetzt wird, wird ein Mechanismus verwendet der bereits vorher Einzug in das ROS-System hielt: Gefahrenzonen. Diese wurden leicht angepasst um sie invertieren zu können und somit aus einer Gefahrenzone eine Sicherheitszone zu machen. Ist ein Bereich als Sicherheitszone definiert, ist jeder andere Bereich automatisch eine Gefahrenzone. Dieser Mechanismus birgt nur eine kleine Änderung im System, schafft es aber Roboter in einen bestimmten Bereich zu locken ohne sie aktiv steuern zu mpsse, indem einfach ihr 'natürlicher' Trieb verwendet wird, von Gefahren zu flüchten.

Wird ein Auftrag erteilt, so nehmen die Roboter die dem Auftrag zugeteilt sind, eine neue Sicherheitszone auf, die den Ausmaßen und der Position des Transportobjekts am Aufnahmeort entsprechen. Roboter die nicht dem Transport zugeteilt wurden, werden diese neue Zone als Gefahrenzone auffassen und versuchen sich diesem Gebiet fernzuhalten.

Ein Objekt kann grundsätzlich aus verschiedenen Geometrien zusammengesetzt werden. Dadurch ist es möglich nicht nur Objekte zu transportieren die eine einfache Form wie Rechtecke oder Kreise haben, sondern verschiedene Rechtecke können dann zu einem 'L' oder 'U' zusammengesetzt werden.

Implementierung Die Klassenhierarchie ist wie in Abbildung 6.12 dargestellt. Die Implementierung eines Objekts ist als Sammelklasse definiert, die verschiedene Geometrien unter sich vereint. Die Geometrien haben eine Hauptklasse von der sie sich ableiten. Die abgeleiteten Klassen müssen letztlich nur noch definieren, wann etwas innerhalb ihrer Fläche ist und was ihre Ausmaße im zweidimensionalen Raum sind. Wird ein Objekt auf eventuelle Kollisionen abgefragt, muss dieses letztlich nur noch über die inneliegenden Geometrien iterieren und sobald eine Kollision statt fand ist das Gesamtergebnis ebenfalls eine Kollision.

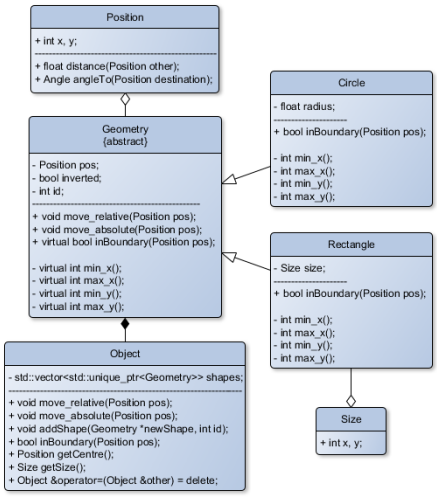


Abbildung 4.8: Die Klassenhierarchie des Objekt-Systems

Füllen eines Raums

Wird eine Sicherheitszone definiert, versuchen die Roboter, die sich in der Gefahrenzone befinden, auf möglichst schnellstem Wege die Sicherheitszone zu betreten. Dazu wird das Zentrum der Sicherheitszone ins Ziel genommen und (ohne mit anderen Robotern zu kollidieren) der direkte Weg darauf zu genommen. Dabei kann es dann allerdings vorkommen, dass eine komplexere Form nicht vernünftig gefüllt werden kann, oder dies sehr lange dauert. Da ein eintreffender Roboter so lange auf den Mittelpunkt zufahren würde, bis die Roboter darin sich genug verteilt haben um genug Platz für den neuen Roboter zu schaffen.

Ein Algorithmus wie man es schafft mit Roboter einen definierten Raum zu füllen lässt sich in (**HIER; QUELLE**) finden. Dieser Algorithmus lässt sich aufgrund anderer Fähigkeiten bei den Robotern nicht und dem gewünschten Schwarmverhalten nicht vollständig nachbilden. Eine andere, bereits implementierte Fähigkeit, lässt sich dagegen gut nutzen um den Algorithmus annähernd nachzubauen: Ausweichen.

Betritt ein Roboter die Sicherheitszone des Transport-Objekts, reduziert dieser seine Geschwindigkeit und fährt langsam weiter auf den Mittelpunkt zu. Neu ankommende Roboter werden durch die vorderen, viel langsameren Roboter zum Ausweichen gezwungen, was letztlich dazu führt, dass die Roboter sich seitlich verteilen, aber langfristig den Nachfolgern Platz machen. Ist ein Roboter nahe genug am Mittelpunkt angekommen oder findet in einem Umkreis von [-90°, 90°] keinen Platz mehr der näher am Mittelpunkt ist als der aktuelle, bleibt er stehen. Roboter die stehen bleiben senden den anderen Robotern ihres Schwarms ein entsprechendes Signal dass sie bereit sind. Haben alle Roboter erkannt dass die anderen bereit sind, ist die Phase des Eintreffens am Abnahmepunkt abgeschlossen.

Von hier an braucht es ein Event dass den Robotern zeigt dass sie mit dem Transportobjekt Richtung Ziel losfahren können. Möglich wäre eine Zeitsteuerung für streng automatisierte Prozesse. Aber auch interne Signale über ROS sind möglich. Durch die ID die jeder Auftrag hat, wäre eine einfache Nachricht '[TransportID#][Losfahren]' bereits zielführend.

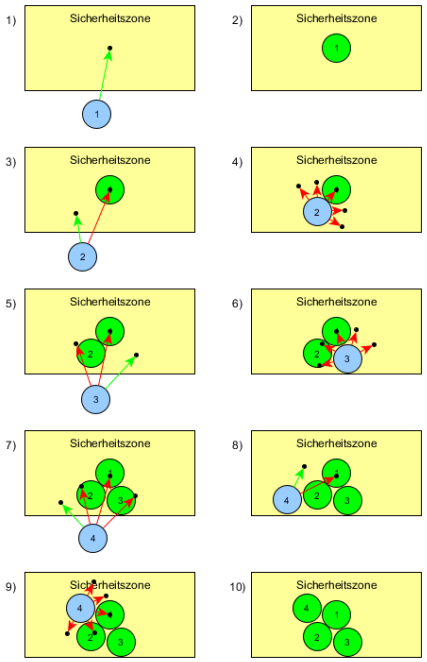


Abbildung 4.9: Roboter füllen das Transportobjekt

Der Algorithmus am Beispiel Abbildung 6.13 zeigt den Algorithmus anhand eines Beispiels mit 4 Robotern.

Der erste Roboter versucht zur Mitte des Objekts zu gelangen und findet sofort Zugang zum Objekt. Er reduziert seine Geschwindkeit und platziert sich langsam im Mittelpunkt des Objektes. Da seine Nähe zum Mittelpunkt klein genug ist, bleibt er letztlich stehen.

Der nächste Roboter kommt, wird allerdings von seinem Vorgänger leicht blockiert. Das führt dazu dass dieser versucht nach links auszuweichen und damit Erfolg hat. Er betritt die Zone, findet aber keinen besseren Ort und bleibt daraufhin stehen.

Der dritte Roboter wird ebenfalls von seinen Vorgängern blockiert, findet aber rechts eine Stelle. Nachdem er diese eingenommen hat findet auch er keine bessere Stelle und bleibt ebenfalls stehen.

Der vierte Roboter findet zunächst links eine Stelle und betritt daraufhin die Zone. Er verlangsamt seine Bewegung und sucht nun nach Stellen die ihn ohne Kollisionen näher zu seinem Ziel führen, wobei er einmal Roboter#2 umkreist. In der Lücke zwischen Roboter#2 und Roboter#1 findet er die beste Stelle und bleibt dort anschließend stehen. Der Algorithmus für diese 4 Roboter ist damit abgeschlossen. Sie stehen alle in der Fläche des Transportobjektes und stehen still, bereit die Lieferung entgegen zu nehmen.

Der Transport

Um den Transport selbst zu realisieren bedient man sich der Hilfe der Anführer. Diese richten sich dynamisch nach dem Winkel aus, den das Transportobjekt zum Ziel hat, wie Abbildung 6.14 zeigt. Danach fangen sie an sich langsam in die Richtung zu bewegen in die sie sich ausgerichtet haben. Die anderen Roboter werden sich aufgrund des Gruppenverhaltens ebenfalls mehr oder weniger nach ihren Anführern ausrichten und das Objekt

so langsam Richtung Ziel bewegen.

Einhalten der Richtung hat Priorität

Kann ein Anführer keine normale Bewegung ausführen, weil er sonst mit einem anderen Roboter kollidieren würde, darf er nicht versuchen auszuweichen, da dies sonst die Ausrichtung der passiven Roboter negativ beeinflussen könnte. Stattdessen drosselt er zunächst sein Bewegungstempo oder bleibt, falls notwendig, ganz stehen. Die richtige Richtung beizuhalten ist wichtig, da sich passive Roboter nicht unmittelbar nach ihren Anführern ausrichten. Gerade wenn es zahlenmäßig wenige Anführer im Vergleich zu passiven Robotern sind, kann es einige Zeit dauern, bis die Einheiten so weit beeinflusst wurden, dass sie in die gewünschte Richtung zeigen. Dreht sich ein Anführer in die verkehrte Richtung, vielleicht sogar in die entgegengesetzte, kann dies zu einer Kettenreaktion führen die alle Roboter betrifft und zusätzlich das Transportobjekt stark vom Weg abbringen.

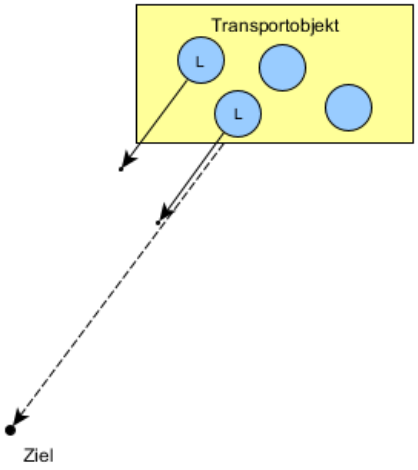


Abbildung 4.10: Der Transport des Objekts

Abschluss des Transports

5 Implementierung

Dieses Kapitel wird sich der Implementierung der Thesis widmen. Es wird gezeigt, wie die Konzeption als Software umgesetzt wurde und einige wichtige Code-Stücke vorgestellt. Die Implementierung wurde auf dem ROS-Framework aufgebaut.

5.1 Generelles zur Implementierung

Die Implementierung baut auf Ubuntu LTS 16.04 mit der ROS-Version 'Lunar' auf, da diese beiden Komponenten das derzeit stabilste Duo bilden. Aufgrund mangelnder Kapazitäten, befindet sich das Ubuntu-System in einer Virtuellen Maschine, was, bis auf einen höheren Ressourcen-Verbrauch, allerdings keinerlei erkennbare Nachteile mit sich zog.

Generell ist das ROS-Framework in C++ und Python verfügbar, wobei diese Sprachen gemischt werden könnten, wenn sie auf verschiedenen Nodes genutzt werden. Als Programmiersprache wurde letztlich C++ verwendet, da es deutlich besser mit den verfügbaren Ressourcen umgeht. In den späteren Auswertungen hat sich diese Wahl positiv bestätigt, da 25 gleichzeitige Roboter eine Menge Ressourcen verschlungen haben und den verfügbaren Computer (auch wegen der VM) bereits an die Grenzen seiner Leistung brachte.

Da ein experimentelles Vorgehen direkt an realen Robotern zu Aufwändig wäre, insbesondere was die Zeitkosten für die Durchführung und Auswertung einer Simulation angeht, beruht die Implementierung auf der Node Turtlesim [1] von ROS (siehe Abschnitt 2.5). Die Turtlesim wurde dahingehend angepasst, dass die Bilder der Turtles durch wesentlich kleinere Bilder schwarzer Pfeile ersetzt wurden die visuellen Aufschluss auf Position und Aurichtung erlauben. Dadurch wurde mehr Platz und Übersichtlichkeit gescahffen. Das Verhalten der Turtles selbst in der Simulation ist unverändert.

Generell ist die Turtlesim nicht dafür gemacht worden, aufwändigere Simulationen zu erproben. Der Zweck ist es einen einfachen Einstieg in das ROS-Framework zu geben und Neulinge durch die Tutorials zu begleiten. Sie bietet aber eine grafische Oberfläche, die bei der Implementierung sehr nützlich ist, insbesondere um Fehler besser entdecken zu können. Bei der Implementierung wurde viel Wert auf eine modulare Bauweise gesetzt. Die Bewegungsbefehle die aktuell an die Turtlesim gehen sind abstrahiert und lassen sich später schnell auf das System des jeweiligen Roboters umschreiben. Andere Nachrichten gehen nicht über die Turtlesim, wodurch die Bewegung das einzige Sub-System ist, dass es nachher anzupassen gilt.

5.2 Nachbau des Schwarms nach Craig Reynolds

Sendet man einen Bewegungs-Befehl an die Turtlesim, wird diese den Befehl für 1 Sekunde ausführen und nach Abschluss die Position des Roboters automatisch an die Subscriber verteilen. Die Subscriber erhalten in ROS ihre Informationen immer über

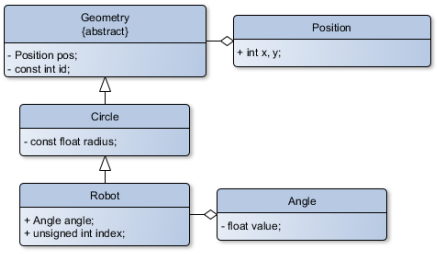


Abbildung 5.1: Die Klassenhierarchie eines Roboter-Objektes

[1] http://wiki.ros.org/turtlesim

Callback-Funktionen. Da diese Funktionen statisch sein müssen, ist es nicht einfach möglich die Daten in einem Objekt zu speichern, sondern sie müssen in globalen Variablen untergebracht werden. Die Informationen über die Position der Roboter werden in einem Array gespeichert. Da die Roboter IDs haben, ist es leicht diese bei 0 beginnen zu lassen und sie dann als Index zur Adressierung innerhalb des Arrays zu nutzen.

Da Roboter eine Position und eine Fläche haben die deren Körper darstellt, macht es am meisten Sinn ihnen einen Kreis als geometrische Form zu vererben, sodass letztlich die Klassen-Struktur aus Abbildung 5.1 entsteht (Funktionen wurden ausgelassen). Der Nutzen der Klasse `Angle` ist der, dass die Klasse automatisch darauf achtet im Bereich [0, 360] zu bleiben: `Angle(350) + Angle(20) == Angle(10)`.

Messages

Die Steuerung der Roboter innerhalb der Turtlesim geschieht nicht direkt durch das eigene Programm. Stattdessen müssen Nachrichten an die Turtlesim gesendet werden die dann letztlich die Roboter bewegt. Insgesamt werden dafür 5 Nachrichten verwendet. Die Nachrichten-Typen werden der Übersichtlichkeit halber in einer verkürzten Schreibweise vorgestellt, wie sie in ROS nicht erlaubt, von Programmiersprachen wie C++ oder Java, aber bekannt ist. Einige Nachrichten arbeiten mit dem Topic 'turtleX', was nichts anderes bedeutet als eine ID für die Roboter der Turtlesim (turtle1, turtle2, ...).

turtlesim/Pose.msg

```
// Empfangen von dem Topic: turtleX/pose
// Gibt Feedback zur Position und Ausrichtung des Roboters
// Wird in dieser Arbeit genutzt, um die Position der Roboter zu speichern
float32 x, y, theta, linear_velocity, angular_velocity
```

geometry_msgs/Twist.msg

```
// Gesendet an das Topic: turtleX/cmd_vel
// Fahrbefehl fuer den Roboter, wird 1 Sekunde lang ausgefuehrt
// Wird in dieser Arbeit genutzt, um die Roboer geradeaus zu fahren zu lassen
Vector3  linear, angular
```

turtlesim/TeleportRelative Service

```
// Gesendet an den Service: turtleX/teleport_relative
// Teleportiert den Roboter zu einem relativen Ort
// Wird in dieser Arbeit genutzt, um die Roboter vor dem Fahrbefehl zu drehen
float32 linear, angular
---
```

turtlesim/Spawn Service

```
// Gesendet an den Service: spawn
// Erschafft einen neuen Roboter in der Turtlesim
float32 x, y, theta
string name
```

```
---
string name
```

turtlesim/SetPen Service

```
// Gesendet an den Service: turtleX/set_pen
// Schaltet den Pen aus. Ein 'Stift' der den gefahrenen Weg markiert
uint8 r, g, b, width, off
---
```

System-Takt: Ticks

Der Ablauf eines Programms geschieht in Ticks (der Takt des Systems, periodische Zeitabstände), in denen zuerst eine Aktion gestartet und danach eine kurze Zeit gewartet wird um Nachrichten anderer Roboter zu empfangen und zu verarbeiten. Es ist damit impliziet nur eine Bewegungs-Aktion pro Tick möglich. Diese kann allerdings Drehung und Fahren kombinieren. Die Bewegung einer Turtle wird von ROS genau 1 Sekunde lang ausgeführt. Die Steuerung der bewegten Entfernung pro Tick lässt sich daher nur über die Geschwindigkeit der Bewegung steuern. In der Praxis hat sich aufgrund von weiteren Verzögerungen in der Turtlesim eine Tick-Länge von 1.1 Sekunden bewährt.

Entschließt sich ein Roboter dazu sich nicht zu bewegen, muss er einen 'leeren' Bewegungsbefehl senden, der dazu führt, dass sich der Roboter nicht bewegt und somit einen Tick abschließen, statt einfach zu warten und nichts zu senden. Der Grund hierführ liegt darin, dass die Systeme Single-Thread Architekturen sind und das Abfragen der Topics expliziet ausgelöst werden muss. Dies geschieht über eine Funktion in ROS, die aufgerufen wird und die automatisch alle abonnierten Topics abfragt, um dann die entsprechenden Callback-Funktionen auszuführen. Außerdem werden dadurch die eigenen Daten auf den Robotern aufgefrischt und eventuelle Informationen gesendet, die nichts mit der Bewegung zu tun haben. Das abschließen eines Ticks synchronisiert somit generell die Daten des eigenen Systems mit dem der anderen.

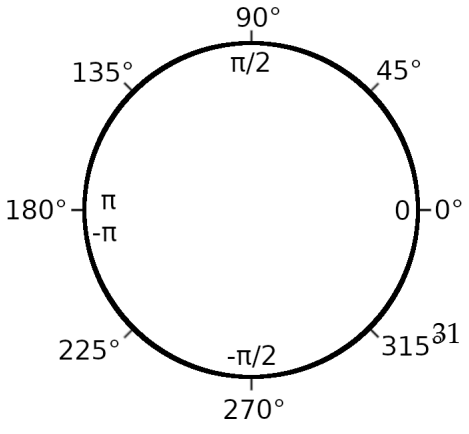
Die Ticks sind trotz allem keine globalen Ticks, die jeden Roboter gleichzeitig. Jeder Roboter arbeitet auf seiner Zeitlinie und agiert unabhängig der Ticks der anderen Einheiten.

Einhalten der 4 Grundregeln

Die Roboter wurden als autonome Einheiten konzipiert die sich ausschließlich anhand von 3 fixen Parametern bewegen und dabei versuchen die 4 Grundlegen des Schwarmverhalten einzuhalten. Grundsätzlich werden von einem Roboter nur andere Einheiten beachtet, die in einem gewissen Radius um ihn herum positioniert sind. Daher wird Am Anfang eine Liste derjenigen erstellt. Dies geschieht durch simples iterieren über die Liste aller Roboter und einer Berechnung über Pythagoras.

Ausrichtung: Passe deine Bewegungsrichtung deinen Nachbarn an

ROS selbst arbeitet bei Winkeln mit dem Bereich von [-3.14, 3.14], wohingegen die Steuerung der Roboter mit dem Bereich [0°, 360°] arbeitet, da dieser besser bearbeitet (und von Menschen gelesen) werden kann. Außerdem ist dadurch generell eine Abstraktions-Ebene entstanden die es später erlaubt



mit jedem bliebig anderen Winkel-System zu arbeiten. Eintreffende Positionsdaten, von anderen Robotern, müssen also erst in die interne Darstellung umberechnet werden. Ebenso wird der Winkel vor dem Senden eines Bewegungs-Befehls automatisch so umberechnet dass ROS ihn akzeptiert. Wird ROS ein zu großer Winkel gesendet, z.B.: 5, so wird sich der Roboter einfach sehr schnell drehen und wärend dieser Zeit schwer unkontrollierbar sein, sonst aber keine weiteren Fehler produzieren. In Abbildung 5.2 ist eine Übersicht über die Winkel zu sehen wie sie im System und von ROS genutzt werden.

Zusammenhang: Versuche deinen Nachbarn nahe zu sein

Für die Berechnung des Mittelpunkts der Nachbarschaft, wurde schlicht die Liste der Roboter, die nahe genug sind, genommen und ein mathematischer Durchschnitt gebildet. Die Koordinaten wurden danach von den eigenen Abgezogen und die resultierenden X/Y-Werte als Vektoren genutzt um den Winkel zu berechnen, den der Mittelpunkt zum Roboter hat. Anschließend wurde noch die Different dieses Winkels zu dem Winkel gebildet, den man fahren wollte. Ein prozentualer Wert dieser Differenz wurde dann vom Winkel abegzogen.

Abschottung: Vermeide Kollisionen mit deinen Nachbarn

Um Kollisionen mit anderen Robotern zu vermeiden, werden zur Kollisionserkennung interne Simulationen verwendet. Im Grunde bedeutet dies, dass berechnet wird wo der Roboter mit dem aktuell gewünschten Winkel und normaler Geschwindigkeit hinfahren wird. Ist dieser Ort weit genug von anderen Robotern entfernt, wird er akzeptiert. Ist er es nicht, wird der gewünschte Winkel in Schritten von 1° erst nach links und dann nach rechts verändert und erneut geschaut ob es am Zielort zu einer Kollision kommen wird.

Unzureichende Erkennung auf dem Weg Um die Berechnung einfacher zu gestalten und Ressourcen zu schonen wurde nicht der Weg selbst überprüft. Es ist also möglich, dass ein Roboter weder auf seinem Ursprungs- oder Zielort mit anderen Robotern kollidiert, wohl aber auf dem Weg dorthin. Ebenfalls ist es in der Turtlesim nicht möglich, eine angefangene Bewegung wieder abzubrechen. Kommt es also während der Ausführung zu einer Kollision, z.B. weil sich zwei Roboter aufeinander zubewegen, kann diese nicht mehr verhindert werden. Ein Ausgleich in der Simulation ist durch eine Abfrage gegeben, ob sich zwei Roboter 'ineinander' befinden. Die entsprechenden Roboter werden dann auf direkter Luftlinie voneinander weggeschoben, um wieder einen plausiblen physikalischen Zustand einzunehmen.

Im der Praxis sollte eine Kollision auf dem Weg kein Problem darstellen, da mit parallelen Threads nebenläufiges Verhalten ingesetzt werden kann. Es wäre also ohne weiteres möglich zu fahren und gleichzeitig die Positionen der anderen im Auge zu behalten. Außerdem ließe sich, im Falle einer Single-Thread-Architektur, die Frequenz der Ticks wesentlich schneller eingestellen, sodass sich die Roboter effektiv nur wenige Millimeter pro Tick bewegen, dafür aber mit über 100 Ticks pro Sekunde. Letztlich kommt es auf die Leistung des verwendeten Computers an.

Flucht: Fliehe vor Dingen, die eine potentielle Gefahr darstellen

Kollisionen mit Gefahrenzonen werden genau wie Kollisionen mit Robotern erkannt und verhindert. Um Gefahrenzonen darzustellen, wird auf Abstraktion und Vererbung gesetzt. Es gibt, wie in Abbildung 5.3 zu sehen ist, eine Hierarchie von Klassen aus denen sich die Zonen zusammensetzen lassen. Die `Geometry` -Klassen die sich in der `Object` -Klasse befinden werden alle von dieser gesteuert. Bewegt sich das Objekt, bewegen sich alle darin enthaltenen Geometrien. Das Hinderniss bleibt dadurch immer in einem plausiblen Zustand. Ebenfalls muss nur die `Object` -Klasse nach einer Kollision befragt werden, diese leitet die Anfrage an alle Geometrien weiter.

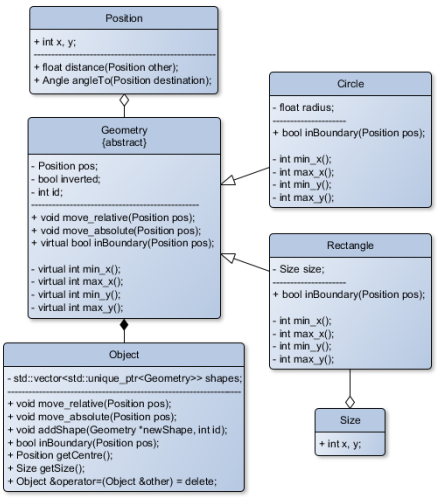


Abbildung 5.3: Die Klassenhierarchie eines Hindernisses

Berechnung der Bewegung

Die Berechnung der Bewegung geschieht letztlich in den Schritten die in Abbildung 5.4 gezeigt werden.

5.3 Anführer

Da der Anführer seinen Schwarm lenkt, verfällt die Einheit, die zum Anführer wurde, in ein spezielles Verhalten. Dieses Verhalten hat mit der bisherigen Implementierung wenig zu tun. Auf das Ziel zuzusteuern und den Schwarm dabei nicht zu verlieren, ihn sogar wieder einzufangen, wenn er abhanden kommt, kann mit den bisherigen Schwarmregeln nicht umgesetzt werden. Aus diesem Grund ist die Implementierung weniger ein umändern der bisherigen Implementierung, sondern es wird eher ein neues Verhalten hinzugefügt und eine Abzweigung im Code um dieses auszuführen.

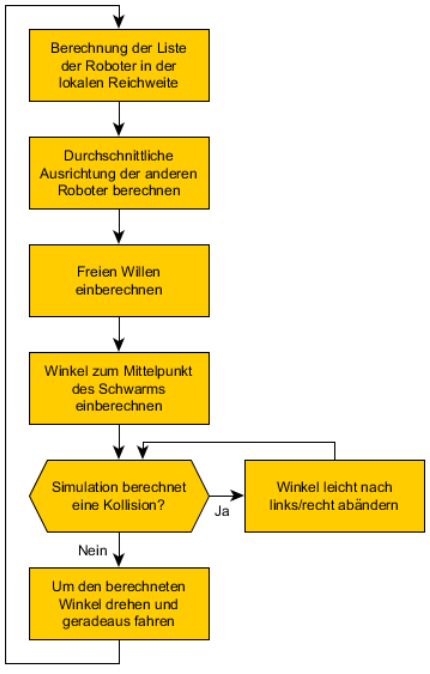


Abbildung 5.4: Ablaufdiagramm Schwarm

Generelles Verhalten

Der Anführer ist ein Zweig im normalen Verhalten der Roboter. Erhält ein Roboter eine Nachricht vom Typ `New_Mission` , richtet er sich nach diesem Ziel aus und fährt ihm entgegen. Behindert ein anderer Roboter, dass der Anführer geradeaus weiter kann, verringert er seine Geschwindigkeit oder bleibt letztlich 1 Tick lang stehen. Ist er nahe genug am Ziel angekommen, fällt er in sein altes Verhalten zurück. Abbildung 5.5 zeigt das Verhalten wie es umgesetzt wurde.

Einbindung des Anführers in ROS

Die Einbindung der Anführer in ROS geschah mit einem zusätzlichen Subscriber der das Topic `flock/mission/new` abonniert hat. Die Funktion des entsprechenden Callbacks nimmt die `New_Mission` an und speichert sie, wenn die ID in der Mission der ID des Roboters entspricht, in einer globalen FIFO-Struktur. Ein Roboter überprüft diese Struktur in seinem normalen Ablauf immer wieder und arbeitet die Mission ab, wenn er eine findet. Da die Implementierung als Single-Thread-Architektur implementiert ist, braucht es auch keine entsprechenden Vorsichtsmaßnahmen, wie sie bei Multi-Thread üblich sind. Die Struktur der Nachricht konnte für die Implementierung so beibehalten werden, wie sie in der Konzeption vorgegeben wurde.

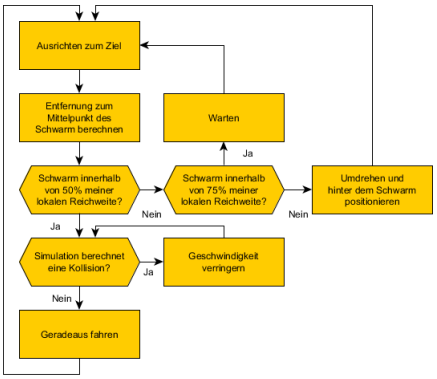


Abbildung 5.5: Ablaufdiagramm Schwarm

Nachrichten-Typ: New_Mission

```
uint8 leader_id // Die ID des ausgewaehlten Anfuehrers
float32 pos_x   // Position des Ziels entlang der X-Achse
float32 pos_y   // Position des Ziels entlang der Y-Achse
```

5.4 Transport von Waren mit Hilfe eines Schwarms

Das letztliche Ziel dieser Arbeit ist es zu prüfen, ob es möglich, und sinnvoll, ist Gegenstände mit Hilfe eines autonomen Schwarms zu transportieren. Dazu musste der Schwarm nun dazu gebracht werden Waren zu bewegen ohne die einzelnen Roboter zu sehr zu beeinflussen. Da generell die Roboter nicht zentral gesteuert werden sollen und auch die Logik möglichst simple bleiben soll, sollte das Programm der einzelnen Einheiten möglichst wenig verändert werden und der 'natürliche' Trieb der Einheiten genutzt werden.

Erteilen von Aufträgen

Eine der ersten Dinge im Ablauf eines Transports ist die Erteilung des Auftrags. Dazu wurde ein neuer Nachrichten-Typ Namens 'MissionNew' im ROS-Netzwerk eingeführt der die folgenden Felder hat:

```
uint8 robot_index_from
uint8 robot_index_to

uint8 leader_number
uint8 mission_id

float32 object_position_x
float32 object_position_y

float32 object_size_x
float32 object_size_y

float32 target_x
float32 target_y
```

Der Nachrichten-Typ definiert eine Spanne von Robotern die den Auftrag ausführen sollen. Diese werden mit ihren IDs angesprochen. Das Intervall der IDs ist, gemäß Programmierstandards, als halboffenes Intervall definiert: [robot_index_from, robot_index_to[. leader_number gibt die Anzahl der Leader an die verwendet werden sollen und mission_id gibt dem derzeitigen Auftrag eine fixe ID um diesen und zugehörige Dinge genau identifizieren und verbinden zu können. Mit object_position_x/-_y ist die Startposition des zu transportierenden Objektes angegeben. Zusammen mit object_size_x/-_y, welche die Ausmaße des Objektes angeben. Die Position des Objekts ist als Mitte des Objekts definiert. target_x/-_y gibt die letztliche Position an die das Objekt einnehmen soll.

Der Auftrag wird von außen an das Topic 'flock/mission/new' gesendet. Der Ersteller des Auftrags muss keine ROS-Node sein, auch wenn dies verschiedene Vorteile hätte. Dadurch das ROS-Topics auch über das Terminal engesprochen werden können, ist das Senden der Nachrichten auch über jedes andere Programm möglich. Das Topic für die neuen Missionen wird von jedem aktiven Roboter abonniert. Entsprechend nimmt jeder Roboter Notiz von diesem Auftrag, auch wenn er nicht direkt mit seiner ID angesprochen wird.

Von Gefahrenzonen zu Sicherheitszonen

Um die betreffenden Roboter in die Zone zu senden, auf der ihnen später das Transportobjekt aufgesetzt wird, wird ein Mechanismus verwendet der bereits vorher Einzug in das ROS-System hielt: Gefahrenzonen. Diese wurden leicht angepasst um sie invertieren zu können und somit aus einer Gefahrenzone eine Sicherheitszone zu machen. Ist ein Bereich als Sicherheitszone definiert, ist jeder andere Bereich automatisch eine Gefahrenzone. Dieser Mechanismus birgt nur eine kleine Änderung im System, schafft es aber Roboter in einen bestimmten Bereich zu locken ohne sie aktiv steuern zu mpsse, indem einfach ihr 'natürlicher' Trieb verwendet wird, von Gefahren zu flüchten.

Wird ein Auftrag erteilt, so nehmen die Roboter die dem Auftrag zugeteilt sind, eine neue Sicherheitszone auf, die den Ausmaßen und der Position des Transportobjekts am Aufnahmeort entsprechen. Roboter die nicht dem Transport zugeteilt wurden, werden diese neue Zone als Gefahrenzone auffassen und versuchen sich diesem Gebiet fernzuhalten.

Ein Objekt kann grundsätzlich aus verschiedenen Geometrien zusammengesetzt werden. Dadurch ist es möglich nicht nur Objekte zu transportieren die eine einfache Form wie Rechtecke oder Kreise haben, sondern verschiedene Rechtecke können dann zu einem 'L' oder 'U' zusammengesetzt werden.

Implementierung Die Klassenhierarchie ist wie in Abbildung 6.12 dargestellt. Die Implementierung eines Objekts ist als Sammelklasse definiert, die verschiedene Geometrien unter sich vereint. Die Geometrien haben eine Hauptklasse von der sie sich ableiten. Die abgeleiteten Klassen müssen letztlich nur noch definieren, wann etwas innerhalb ihrer Fläche ist und was ihre Ausmaße im zweidimensionalen Raum sind. Wird ein Objekt auf eventuelle Kollisionen abgefragt, muss dieses letztlich nur noch über die inneliegenden Geometrien iterieren und sobald eine Kollision statt fand ist das Gesamtergebnis ebenfalls eine Kollision.

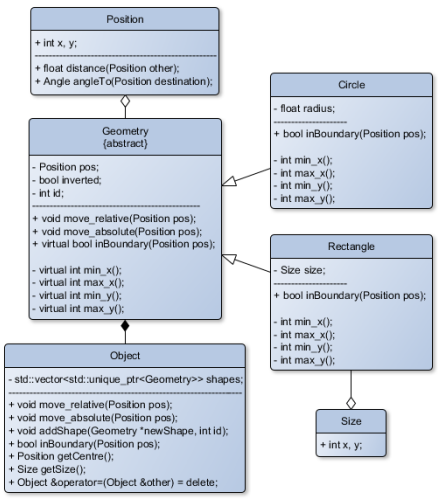


Abbildung 5.6: Die Klassenhierarchie des Objekt-Systems

Füllen eines Raums

Wird eine Sicherheitszone definiert, versuchen die Roboter, die sich in der Gefahrenzone befinden, auf möglichst schnellstem Wege die Sicherheitszone zu betreten. Dazu wird das Zentrum der Sicherheitszone ins Ziel genommen und (ohne mit anderen Robotern zu kollidieren) der direkte Weg darauf zu genommen. Dabei kann es dann allerdings vorkommen, dass eine komplexere Form nicht vernünftig gefüllt werden kann, oder dies sehr lange dauert. Da ein eintreffender Roboter so lange auf den Mittelpunkt zufahren würde, bis die Roboter darin sich genug verteilt haben um genug Platz für den neuen Roboter zu schaffen.

Ein Algorithmus wie man es schafft mit Roboter einen definierten Raum zu füllen lässt sich in (**HIER; QUELLE**) finden. Dieser Algorithmus lässt sich aufgrund anderer Fähigkeiten bei den Robotern nicht und dem gewünschten Schwarmverhalten nicht vollständig nachbilden. Eine andere, bereits implementierte Fähigkeit, lässt sich dagegen gut nutzen um den Algorithmus annähernd nachzubauen: Ausweichen.

Betritt ein Roboter die Sicherheitszone des Transport-Objekts, reduziert dieser seine Geschwindigkeit und fährt langsam weiter auf den Mittelpunkt zu. Neu ankommende Roboter werden durch die vorderen, viel langsameren Roboter zum Ausweichen gezwungen, was letztlich dazu führt, dass die Roboter sich seitlich verteilen, aber langfristig den Nachfolgern Platz machen. Ist ein Roboter nahe genug am Mittelpunkt angekommen oder findet in einem Umkreis von [-90°, 90°] keinen Platz mehr der näher am Mittelpunkt ist als der aktuelle, bleibt er stehen. Roboter die stehen bleiben senden den anderen Robotern ihres Schwarms ein entsprechendes Signal dass sie bereit sind. Haben alle Roboter erkannt dass die anderen bereit sind, ist die Phase des Eintreffens am Abnahmepunkt abgeschlossen.

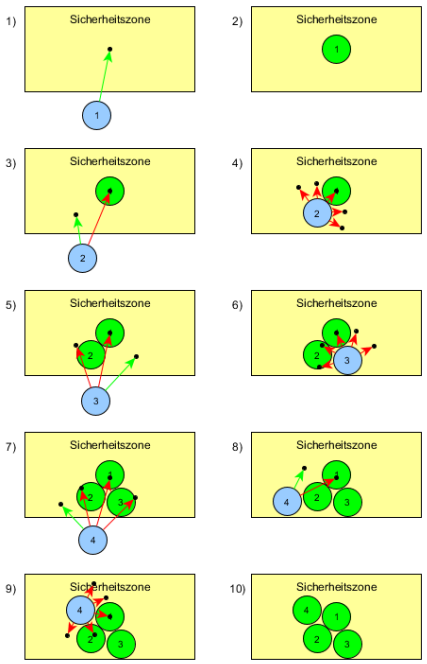


Abbildung 5.7: Roboter füllen das Transportobjekt

Von hier an braucht es ein Event dass den Robotern zeigt dass sie mit dem Transportobjekt Richtung Ziel losfahren können. Möglich wäre eine Zeitsteuerung für streng automatisierte Prozesse. Aber auch interne Signale über ROS sind möglich. Durch die ID die jeder Auftrag hat, wäre eine einfache Nachricht '[TransportID#][Losfahren]' bereits zielführend.

Der Algorithmus am Beispiel Abbildung 6.13 zeigt den Algorithmus anhand eines Beispiels mit 4 Robotern.

Der erste Roboter versucht zur Mitte des Objekts zu gelangen und findet sofort Zugang zum Objekt. Er reduziert seine Geschwindkeit und platziert sich langsam im Mittelpunkt des Objektes. Da seine Nähe zum Mittelpunkt klein genug ist, bleibt er letztlich stehen.

Der nächste Roboter kommt, wird allerdings von seinem Vorgänger leicht blockiert. Das führt dazu dass dieser versucht nach links auszuweichen und damit Erfolg hat. Er betritt die Zone, findet aber keinen besseren Ort und bleibt daraufhin stehen.

Der dritte Roboter wird ebenfalls von seinen Vorgängern blockiert, findet aber rechts eine Stelle. Nachdem er diese eingenommen hat findet auch er keine bessere Stelle und bleibt ebenfalls stehen.

Der vierte Roboter findet zunächst links eine Stelle und betritt daraufhin die Zone. Er verlangsamt seine Bewegung und sucht nun nach Stellen die ihn ohne Kollisionen näher zu seinem Ziel führen, wobei er einmal Roboter#2 umkreist. In der Lücke zwischen Roboter#2 und Roboter#1 findet er die beste Stelle und bleibt dort anschließend stehen. Der Algorithmus für diese 4 Roboter ist damit abgeschlossen. Sie stehen alle in der Fläche des Transportobjektes und stehen still, bereit die Lieferung entgegen zu nehmen.

Der Transport

Um den Transport selbst zu realisieren bedient man sich der Hilfe der Anführer. Diese richten sich dynamisch nach dem Winkel aus, den das Transportobjekt zum Ziel hat, wie Abbildung 6.14 zeigt. Danach fangen sie an sich langsam in die Richtung zu bewegen in die sie sich ausgerichtet haben. Die anderen Roboter werden sich aufgrund des Gruppenverhaltens ebenfalls mehr oder weniger nach ihren Anführern ausrichten und das Objekt so langsam Richtung Ziel bewegen.

Einhalten der Richtung hat Priorität

Kann ein Anführer keine normale Bewegung ausführen, weil er sonst mit einem anderen Roboter kollidieren würde, darf er nicht versuchen auszuweichen, da dies sonst die Ausrichtung der passiven Roboter negativ beeinflussen könnte. Stattdessen drosselt er zunächst sein Bewegungstempo oder bleibt, falls notwendig, ganz stehen. Die richtige Richtung beizuhalten ist wichtig, da sich passive Roboter nicht unmittelbar nach ihren Anführern ausrichten. Gerade wenn es zahlenmäßig wenige Anführer im Vergleich zu passiven Robotern sind, kann es einige Zeit dauern, bis die Einheiten so weit beeinflusst wurden, dass sie in die gewünschte Richtung zeigen. Dreht sich ein Anführer in die verkehrte Richtung, vielleicht sogar in die entgegengesetzte, kann dies zu einer Kettenreaktion führen die alle Roboter betrifft und zusätzlich das Transportobjekt stark vom Weg abbringen.

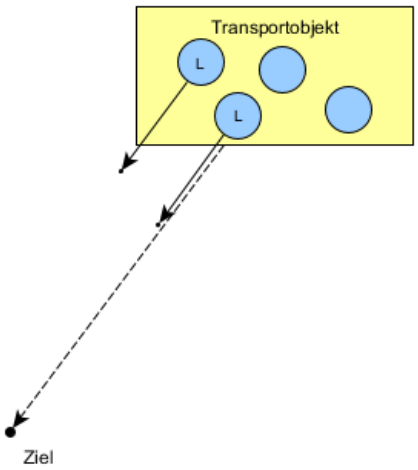


Abbildung 5.8: Der Transport des Objekts

Abschluss des Transports

6 Evaluation

In diesem Kapitel werde ich die Ergebnisse der einzelnen Phasen der Thesis beschreiben. Nach jeder Phase wurden Simulationen mit der Implementierung gestartet und verschiedene Informationen aufgezeichnet. Der Hauptgrund hierfür ist, dass durch die Ergebnisse zum einen ein allgemeines Gefühl für den Schwarm gegeben werden kann und wie er einzustellen ist. Zum anderen liefern die Beobachtungen wichtige Informationen für die späteren Phasen der Konzeption. In diesem Kaputel werden nicht alle Statistiken vorgestellt, sondern nur die, die als interessant empfunden wurden.

6.1 Generelles zur Evaluation

Da die Roboter des Schwarms einen gewissen freien Willen haben, der entscheidet wo sie hinfahren, und dieser schlicht Zufall ist, werden die Simulationen mit ihren Parametern immer mehrmals durchgeführt. Um Ausreißer in der Auswertung auszumerzen, wurde ebenfalls nicht mit einem normalen Durchschnitt gerechnet, sondern mit einem Durchschnitt über die mittleren Ergebnisse.

Da die Simulationen mit über 5 Parameter weit mehr als 100 verschiedene Einstellungen zulassen, konnte nicht jede Konfiguration getestet werden. Stattdessen wurden die Einstellungen für die Simulationen durch Gefühl und den Ergebnissen der vorangegangenen Simulationen ausgewählt, um die Anzahl im Bereich des zeitlich machbaren zu halten.

6.2 Nachbau des Schwarms nach Craig Reynolds

Nachdem die Implementierung des Schwarms abgeschlossen ist, galt es diese zu testen und Statistiken zu erheben. Dazu wurde die Simulation mit verschiedenen Parametern gestartet und 25 Roboter zufällig in der Simulation verteilt. Die Simulationen wurden für 1 Stunde laufen gelassen und in 10s-Abständen gemessen, wie viele Schwärme sich gebildet haben. Ein Schwarm war hierbei eine zusammenhängende Kette von Robotern, die nicht weiter als ihre lokale Reichweite voneinander entfernt sind. Das bedeutet, dass alle Roboter innerhalb eines Schwarm die anderen beeinflussen. Teilweise mag dies direkt geschehen sein, wenn sie innerhalb der lokalen Reichweite waren. Aber auch indirekte Beeinflussung ist es möglich, indem Roboter beeinflusst wurden, durch Roboter die von anderen beeinflusst wurden.

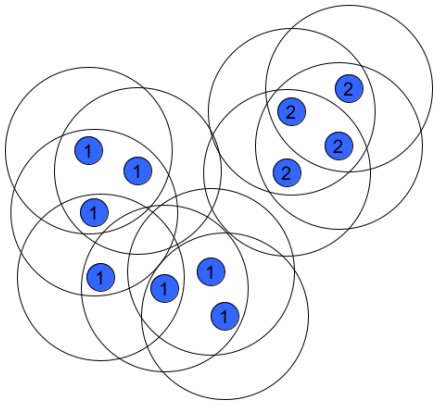


Abbildung 6.1: Einordnung der Roboter in Schwärme

Abbildung 6.1 zeigt ein Beispiel mit 2 Schwärmen. Die Roboter sind als blaue Kreise eingezeichnet, ihre jeweilige lokale Reichweite als schwarzer Kreis drum herum. Eine Zahl im inneren zeigt die Zugehörigkeit zum jeweiligen Schwarm.

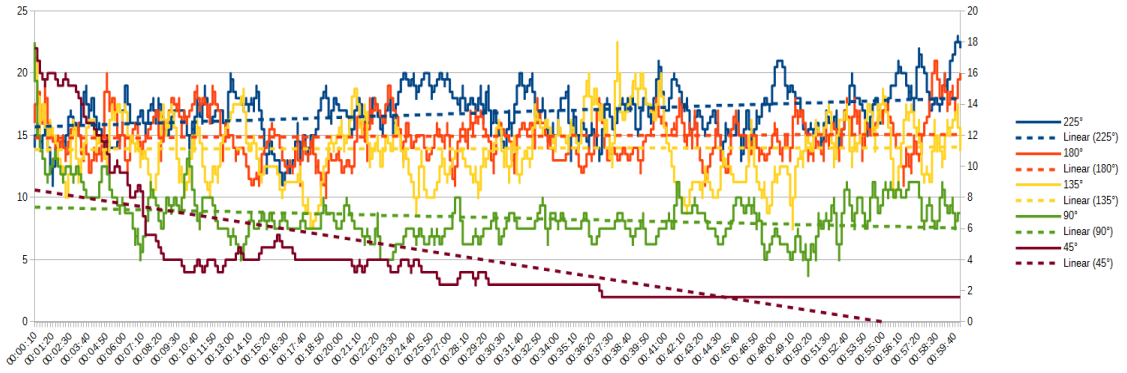


Abbildung 6.2: Entwicklung der Schwärme in Abhängigkeit ihres freien Willens

Da das Verhalten der Roboter, aufgrund ihres freien Willens und der zufälligen Platzierung in der Simulation, starken Schwankungen ausgesetzt ist, wurden immer 10 Simulationen gestartet und aus den mittleren 6 Werten ein Durchschnitt gebildet. Dadurch wurden Ausreißer eliminiert und die Statistiken können sinnvolle Mittelwerte zeigen. Der Rand der Simulation wurde als Hinderniss gewertet, was bedeutet, dass die Roboter nicht stumpf weiter gefahren, sind nachdem dieser erreicht wurde, sondern sie versucht haben diesem auszuweichen.

Statistik: Freier Wille

In Abbildung 6.2 zu sehen, ist eine Statistik die die Entwicklung von Schwärmen mit verändertem freien Willen zeigt. Über die X-Achse hinweg ist die Zeit der Simulation von 0-60 Minuten, die Y-Achse zeigt die Anzahl der vorhandenen Schwärme nach obiger Definition. Grundsätzlich wäre es innerhalb der Simulation aufgrund des verfügbaren Platzes möglich gewesen, dass jeder Roboter nur sich selbst als Schwarm hat. Neben den Graphen selbst, die die Entwicklung der Schwärme zeigen, ist ebenfalls ein linearer Trend in gleicher Farbe und gestrichelter Linie eingezeichnet worden.

Eigenschaften des Schwarms:

- Größe des Schwarms: 25 Roboter
- Lokale Reichweite: 5% der Größe der Simulation
- Geschwindigkeit: 10% der lokalen Reichweite
- Drang zur Gruppierung: 0%
- Freier Wille: Variabel

Zu sehen ist, dass ein Roboter mit einem freien Willen von 225° (112.5° in beide Richtungen) eher dazu neigt sich zu verteilen statt sich zu sammeln. Der Einfluss der Regel sich nach seinen Nachbarn auszurichten liegt bei unter 50% und unterliegt somit dem freien Willen. Dass es dennoch dazu kam dass sich Schwärme gebildet haben, resultiert einerseits daraus, dass aus der Menge [-112.5°, 112.5°] der letztliche Wert zufällig ermittelt wurde, der Durchschnitt also um 0° herum liegt, mit einer durchschnittlichen Abweichung von 56.25° zu beiden Seiten. Anderseits spielt aber auch der mangelde Platz zum verteilen eine Rolle und Roboter die zufällig aufeinander zugefahren sind im nächsten Tick als Schwarm erkannt wurden, unabhängig vom Grund warum sie zusammen waren.

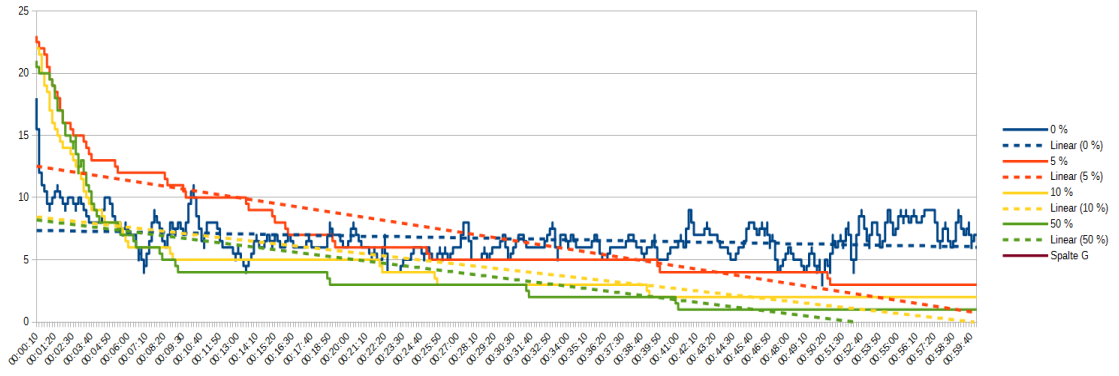


Abbildung 6.3: Entwicklung der Schwärme in Abhängigkeit ihres Dranges zur Gruppierung mit 90° eigenem Willen

Bei einem freien Willen von 180° und 135° zeigt sich der Trend nahezu konstant. Schwärme wurden gebildet und im gleichen Maße wieder aufgelöst.

Ab einem freien Willen von 90° und darunter zeigt sich dagegen ein Trend zur Gruppierung. Der Drang zur Gruppierung nimmt einen Großteil des Verhaltens ein und auf lange Sicht wäre die Zahl der Schwärme gegen 1 gesunken. Dies zeigt sich vor allem bei einem Wert von 45°, der bereits während einiger der einstündigen Simulationen dazu geführt hat, dass sich alle Roboter zu einem einzigen Schwarm versammelt haben.

Statistik: Gruppierungsdrang

In Abbildung 6.3 und Abbildung 6.4 zu sehen, sind Statistiken die die Entwicklung von Schwärmen mit verändertem Drang zur Gruppierung zeigen. Über die X-Achse hinweg ist die Zeit der Simulation von 0-60 Minuten, die Y-Achse zeigt die Anzahl der vorhandenen Schwärme. In der ersten Statistik behielten sie dabei einen freien Willen von 90°, bei der zweiten wurde ein freier Wille von 225° eingestellt.

Eigenschaften des Schwarms:

- Größe des Schwarms: 25 Roboter
- Lokale Reichweite: 5% der Größe der Simulation
- Geschwindigkeit: 10% der lokalen Reichweite
- Freier Wille: 90° bzw. 225°
- Drang zur Gruppierung: Variabel

Bei einem freien Willen von 90° ist zu sehen, dass die Anzahl der Schwärme bei einem Drang zur Gruppierung von 0% konstant bleibt. Diese Entwicklung war bereits in Abschnitt 6.2 zu sehen. Die Start-Anzahl war dabei in allen Simulationen annähern gleich. Sobald der Drang zur Gruppierung über 0% steigt, zeigt sich, dass die Roboter sich abhängig vom eingestellten Wert unterschiedlich schnell zu Schwärmen zusammenziehen und diese auch nicht wieder verlassen. Ein höherer Wert sorgte dabei analog dafür, dass sich die Schwärme schneller sammelten.

Bei einem freien Willen von 225° zeigt sich dagegen deutlich mehr Varianz. Die Entwicklung um 0% herum ist dabei wieder die gleiche wie in Abschnitt 6.2 bereits zu sehen war. Ein Wert von 5% neigt die Trendlinie zwar leicht nach unten, die Daten zeigen aber

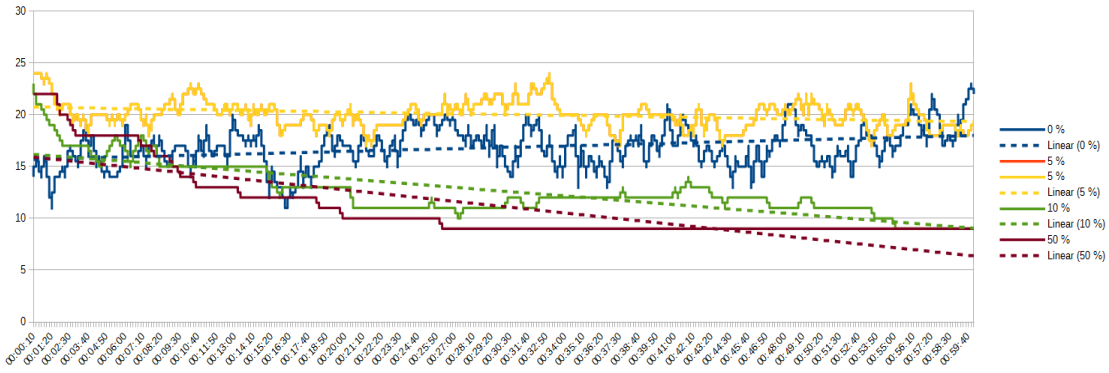


Abbildung 6.4: Entwicklung der Schwärme in Abhängigkeit ihres Dranges zur Gruppierung mit 225° eigenem Willen

nach wie vor hohe Fluktuationen und Schwärme die sich zunächst gebildet haben, brechen aufgrund des großen freien Willens wieder auseinander und die Roboter trennen sich. Bei 10% wird der Trend zwar umso deutlicher, die Fluktuationen sind aber nach wie vor zu sehen und Schwärme neigen noch immer dazu sich gelegentlich zu trennen. Erst ab einem Wert von 50% bleiben Schwärme stabil und langsam aber sicher fügen sich alle Roboter zu einem einzelnen Schwarm zusammen.

Statistik: Negativer Gruppierungsdrang

Zum Schluss wurde noch eine Statistik mit negativem Gruppierungsdrang erstellt. Hintergrund dieses Versuchs war es, Roboter verteilen zu lassen. Damit sie sich während des Leerlaufs nicht versammeln und so in ihrer Masse zu einem größeren Hinderniss werden, wurde versucht die Roboter dazu zu bringen sich zu meiden und damit auszulösen, dass sie sich verteilen.

Eigenschaften des Schwarms:

- Größe des Schwarms: 25 Roboter
- Lokale Reichweite: 5% der Größe der Simulation
- Geschwindigkeit: 10% der lokalen Reichweite
- Freier Wille: 45°
- Drang zur Gruppierung: Variabel

Wie in Abbildung 6.5 zu sehen ist, versuchen sich die Roboter bei einem freien Willen von 45° und einem Gruppierungsdrang von 0% zu Schwärmen zusammenzuschließen. Über die X-Achse hinweg ist die Zeit der Simulation von 0-60 Minuten, die Y-Achse zeigt die Anzahl der vorhandenen Schwärme. Dieses Verhalten ist nachvollziehbar, da ein derart geringer freier Wille dazu führt, dass die Bewegung größenteils davon abhängig ist die anderen zu imitieren. Roboter im Radius ihrer lokalen Reichweite fahren also meist in die selbe Richtung und gruppieren sich so auf natürliche Weise. Wenn sie gegen die Grenzen der Simulation stoßen und versuchen dieser auszuweichen, kommen sie dabei noch näher zusammen.

Ein negativer Gruppierungsdrang von bereits 0.05% reicht jedoch aus um die Trendlinie merklich nach oben zu verlagern und die Gruppierung in Schwärme zumindest

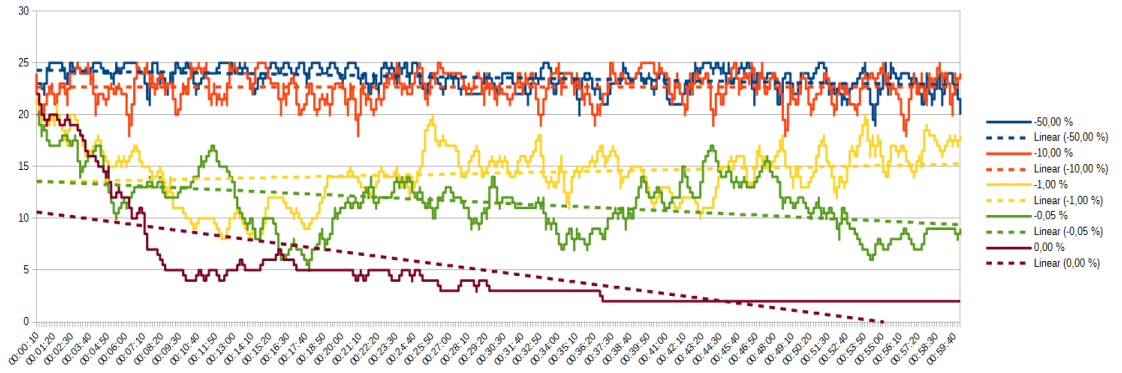


Abbildung 6.5: Entwicklung der Schwärme mit negativem Gruppierungsdrang

langsamer vollziehen zu lassen. Die Fluktuationen nehmen deutlich zu und die Schwärme scheinen nun deutlich instabiler zu sein. Zwar finden sich immer wieder Roboter in Schwärmen ein, sie brechen aber auch oft auseinander und die Einheiten gehen wieder getrennte Wege.

Schon ab 1% negativen Gruppendrangs zeigt die Trendlinie, dass die Gruppen eher dazu neigen sich aufzulösen statt sich neu zu bilden. Der Winkel zur Mitte des eigenen Schwarms kann maximal 180° (in beide Richtungen) groß sein, das heißt der negative freie Wille kann einen Einfluss von maximal 1.8° einnehmen. Trotzdem reicht bereits dieser Einfluss aus um die Roboter zu zu bewegen sich voneinander weg zu bewegen.

Ab einem Wert von 10% zeigt sich letztlich eine totale Streuung und die Trendlinie bleibt eine Konstante. Die Roboter schaffen es teilweise keinen einzigen Schwarm zu bilden (bzw. nur Schwärme in denen sie selbst als einziger Roboter vertreten sind).

Überraschenderweise zeigt ab einem Wert von 50% die Trendlinie wieder nach unten und die Roboter scheinen sich erneut zu gruppieren. Der Grund hierführ ist, dass die Roboter beim Ausweichen ihrer Kollegen überschießen und sich so weit in die andere Richtung drehen, dass sie letztlich vom Winkel wieder näher dran sind als vorher. Legt man Wert darauf, dass sich die Roboter verteilen, gilt es also einen passenden Prozentwert einzustellen und diesen nicht zu übertreiben, da man sonst das Gegenteil dessen erreicht was man als Ziel hatte.

6.3 Anführer

Nachdem der Anführer implementiert wurde, wurde das konzipierte Verhalten getestet und verschiedene Statistiken erhoben, inwiefern ein Anführer in der Lage ist einen Schwarm an sein Ziel zu führen, ohne dass diese aktiv gesteuert werden müssen. Dafür wurde der gesamte Schwarm an einer Ecke der Simulation platziert. Die Einheiten waren dabei immer nahe genug beieinander, um als einzelner Schwarm angesehen zu werden, aber ansonsten zufällig verteilt. Einem der Roboter wurde daraufhin zufällig eine New_Mission zugeteilt und damit zum Anführer gewählt. Dieser hat daraufhin versucht seinen Schwarm zum Ziel zu führen.

Auch in dieser Analyse spielt der Zufall eine beachtliche Rolle, da der freie Wille der Einheiten darauf ausgelegt ist. Es wurden daher immer 10 Simulationen durchgeführt und aus den mittleren 6 Ergebnissen ein Mittelwert berechnet, um Aureißer möglichst zu ignorieren.

Nachbesserung der Implementierung

Recht schnell hat sich gezeigt, dass das Einfangen des Schwarms, wie es in Abschnitt 4.3 vorgestellt wurde, mehr negative als positive Auswirkungen hat. Versucht der Roboter seinen Schwarm einzufangen, ändert er, dadurch dass er sich um 180° umdrehen muss, die gesamte Ausrichtung des Schwarms. Dies führte in den Simulationen meist dazu, dass der Schwarm vollkommen abgedriftet ist und der Anführer Mühe und Not hatte, ihn wieder auf das Ziel auszurichten. Meist verlor er dabei den Schwarm wieder, bevor dieser ausgerichtet war, womit das ganze wieder von vorne los ging. Durch dieses Fehlverhalten mussten die meisten Simulationen abgebrochen werden und die Ergebnisse waren letztlich unbrauchbar, da zu viel von Hand gefiltert werden musste.

Aus diesem Grund wurde der Teil des Algorithmus' wieder entfernt. Stattdessen wartet der Anführer nun, wenn der Schwarm 50% der lokalen Reichweite entfernt ist und gibt auf, wenn der Mittelpunkt die lokale Reichweite ganz verlassen hat. Aufgeben bedeutet in diesem Kontext, dass der Anführer ohne seinen Schwarm zum Ziel gefahren ist. Dies führte allgemein zu besseren Ergebnissen und der Schwarm konnte zumindest in Teilen zum Ziel gebracht werden.

Abhängigkeit: Freier Wille

In dieser Statistik wurde geprüft wie sich verschiedene Werte für den freien Willen darauf auswirken, dass der Schwarm zum Ziel geführt werden kann. Über die X-Achse von Abbildung 6.6 hinweg ist die vergangene Wegstrecke in Prozent angegeben, die Y-Achse zeigt die Anzahl der Roboter die im Schwarm des Anführers vorhanden sind.

Eigenschaften des Schwarms:

- Größe des Schwarms: 25 Roboter
- Anzahl der Anführer: 1 Anführer
- Lokale Reichweite: 5% der Größe der Simulation
- Geschwindigkeit: 10% der lokalen Reichweite
- Drang zur Gruppierung: 5%
- Freier Wille: Variabel

In der Abbildung 6.6 zu sehen, ist dass der Schwarm anfangs immer zusammen blieb. Da die Roboter als gemeinsamer Schwarm zusammen gestartet sind, mussten sie sich nicht erst finden. Wenn der Schwarm in eine andere Richtung lenkte als der Anführer, blieb der Anführer ab einer bestimmten Entfernung stehen und wartete. Zog der Schwarm weiterhin in die Richtung löste er sich irgendwann vom Anführer und dieser zog alleine weiter Richtung Ziel. Durch den relativ hohen Wert von 5% im Gruppendrang blieb der Schwarm jederzeit zusammen. Wenn sich der Schwarm vom Roboter löste, tat er dies immer im gesamten.

Je niedriger der Wert im freien Willen war, desto eher löste sich der Schwarm scheinbar von seinem Anführer. Dies ist darauf zurückzufiühren, dass bei einer Gruppengröße von 25 Robotern der Einfluss eines Anführers nur 4.2% ausmacht und der Schwarm mehr vom Zufall als vom Anführer gelenkt wird. Dass die Roboter mit dem größeren freien Willen letztlich scheinbar länger beim Anführer blieben, ist dem geschuldet, dass die Roboter weniger eng als Schwarm zusammen bleiben und durch die Verteilung der Roboter mehr in der Nähe des Anführers blieben. Die Roboter mit dem geringen freien Willen hingegen blieben mehr zusammen und sind somit

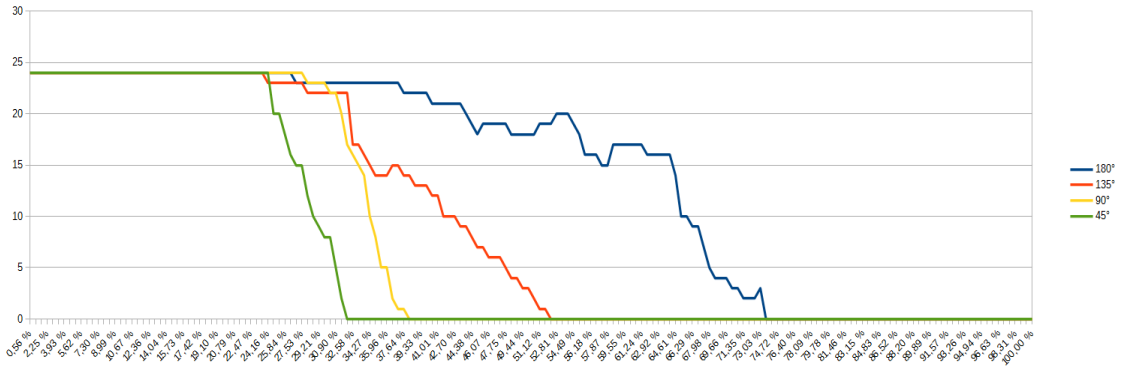


Abbildung 6.6: Erfolg eines Anführer ins Abhängigkeit des freien Willens

Abhängigkeit: Gruppengröße und -drang

In den folgenden Statistiken wurde ein Anführer mit steigender Gruppenzahl und unterschiedlichem Gruppendrang gemessen. Über die X-Achse von Abbildung 6.6 hinweg ist der genutzte Gruppendrang angegeben. Das erste Diagramm zeigt dabei, für wie viel Prozent der Wegstrecke der gesamte Schwarm beisammen gehalten werden konnte. Das zweite Diagramm zeigt die Größe des Schwarms um den Anführer herum beim erreichen des Ziels. Das dritte Diagramm zeigt die Anzahl der Wartezeiten die der Anführer hatte. Eine höhere Anzahl an Wartezeiten ist direkt equivalent zu einer längeren Zeit die die Simulation gebraucht hat.

Dabei werden immer 2 Werte gezeigt. 'Mittelwert', der einen einfachen Mittelwert über alle 15 Messwerte anzeigt und 'Median', bei dem der Mittelwert auf die mittleren 10 Messwerte begrenzt wurde um Ausreißer auszusortieren. Da der Anführer selbst immer am Ziel ankommt, wurde er bei den Messwerten abgezogen. Die Anzahl der dargestellten Roboter entspricht also immer denjenigen, die passiv zum Ziel geführt wurden.

Eigenschaften des Schwarms:

- Größe des Schwarms: Variabel Roboter
- Anzahl der Anführer: 1 Anführer
- Lokale Reichweite: 5% der Größe der Simulation
- Geschwindigkeit: 10% der lokalen Reichweite
- Drang zur Gruppierung: Variabel
- Freier Wille: 90°

In Abbildung 6.7 zu sehen ist zunächst einmal, dass der Anführer in einer Gruppe mit nur 5 Robotern einen sehr starken Einfluss auf die Gruppe hat. Bereits ohne Gruppendrang gibt es eine Erfolgsquote von über 20%. Ab 5% Gruppendrang gibt es schon annähernd 80% Erfolgsquote und ab 10% bekommt er seinen Schwarm immer ans Ziel gebracht.

Die Größe des Schwarms zeigt ein sehr ähnliches Bild. Kommen bei 0% Gruppendrang im Schnitt nur sehr wenige Roboter an, steigert es sich sprunghaft auf 4 Einheiten und ab einem Gruppendrang von 10% kommen immer alle Roboter an.

Die Wartezeiten sind nicht ganz so konstant, zeigen aber, dass es noch deutliche Unterschiede zwischen den Prozentwerten im Bereich von 10-100% gibt, auch wenn die beiden

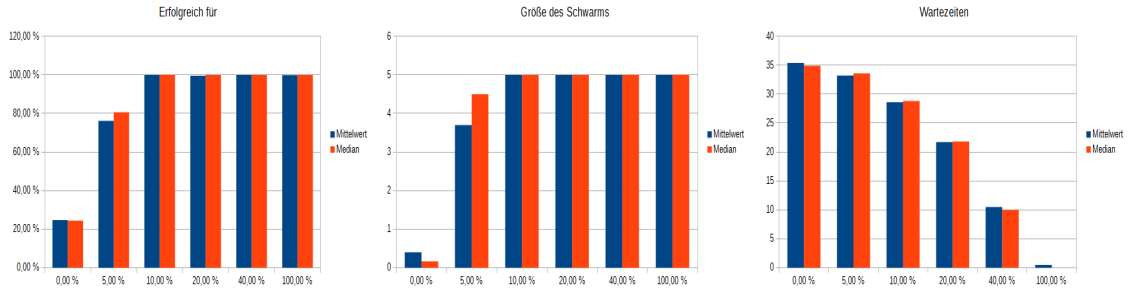


Abbildung 6.7: Erfolg eines Anführer ins Abhängigkeit der Gruppengröße und des Gruppendrangs mit 5 passiven Robotern

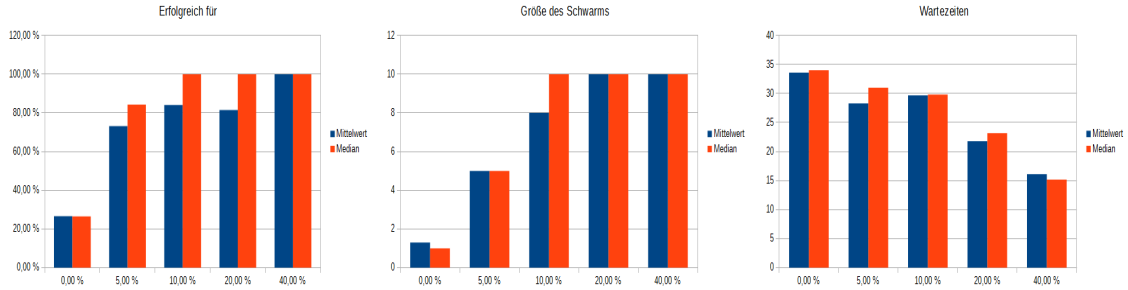


Abbildung 6.8: Erfolg eines Anführer ins Abhängigkeit der Gruppengröße und des Gruppendrangs mit 10 passiven Robotern

vorigen Diagramme dort keine mehr gezeigt haben. Mit einem höheren Gruppendrang sinkt die Anzahl der Wartezeiten immer mehr und bei 100% gibt es sogar annähernd gar keine mehr.

Die Diagramme in Abbildung 6.8 zeigen, dass der Einfluss des Anführers mit 5 Robotern mehr schon deutlich nachlässt. Konnten vorher scon ab 10% Gruppendrang beste Ergebnisse erziehlt werden, zeigt der Durchschnitt nun deutlich niedrigere Werte. Dies bedeutet vor allem dass die Fehlerquote zunahm und die 10 mittleren Ergebnisse nun ebenfalls höhere Schwankungen aufweisen. Nur bei 40% sieht man noch einen vollen Erfolg über alle Messwerte hinweg.

Die Größe des Schwarms ist von ähnlichen Rückschlägen betroffen. Bei 0% Gruppendrang schafft es nur jeder zehnte Roboter ins Ziel, fast der gleiche Wert wie bei 5 passiven Robotern. Bei 10% Gruppendrang schafft es der Median auf einen vollen Erfolg, der Mittelwert hingegen erst ab 20%.

Die Wartezeiten zeigen hingegen nicht unbedingt den gleichen Trend. Sie sind zwar allgemein stiegen, der Trend ist aber nicht mehr so steil wie in der Statistik mit 5 passiven Robotern. Steigen beide Diagramme bei ca. 30 Wartezeiten ein, ist nun auch bei 40% Gruppendrang eine deutlich höhere Wartezeit zu sehen.

In Abbildung 6.9 nimmt der allgemeine Trend weiter seinen Lauf. Die Erfolgsquote für die Werte unter 40% nehmen immer weiter ab, wenn auch die 40% selbst noch einen vollen Erfolg vorweisen kann.

Es kommen auch weniger Roboter insgesamt im Ziel an, wenn die Gesamtzahl der ankommenden Roboter auch bei 20% Gruppendrang noch recht hoch ist. Bei 40% Gruppendrang ist nach wie vor ein voller Ausschlag zu sehen.

Auch die Wartezeiten zeigen den selben Trend wie zuvor. Sie starten recht hoch, der Trend bleibt aber flacher als bei den Messungen mit 5 passiven Robotern weniger.

Ab einer passiven Anzahl von 20 Robotern zeigt sich in Abbildung 6.10 erstmals der

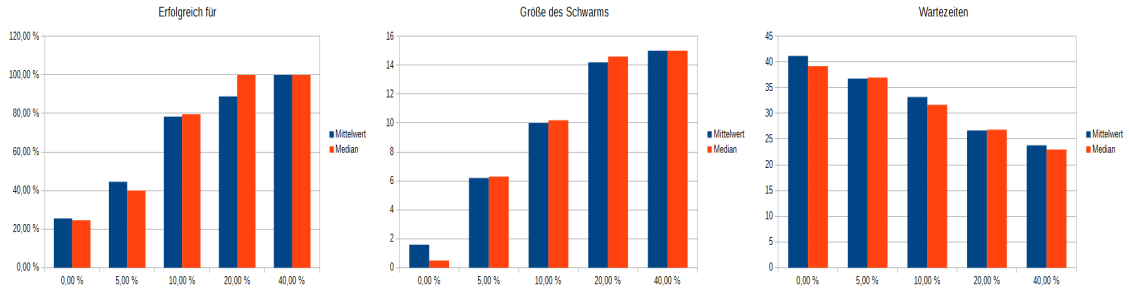


Abbildung 6.9: Erfolg eines Anführer ins Abhängigkeit der Gruppengröße und des Gruppendrangs mit 15 passiven Robotern

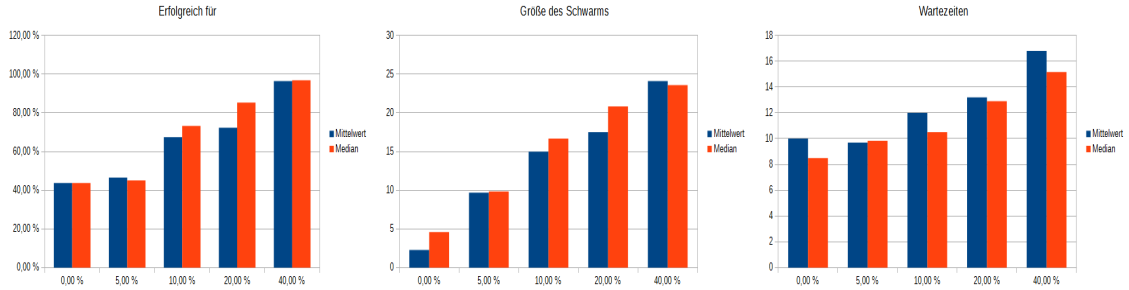


Abbildung 6.10: Erfolg eines Anführer ins Abhängigkeit der Gruppengröße und des Gruppendrangs mit 20 passiven Robotern

Trend, dass die Wartezeiten zunehmen, wenn der Gruppendrang größer wird. Ein genauerer Blick auf das Diagram zeigt jedoch auch, dass die Wartezeiten allgemein sehr viel niedriger sind als in den Messungen mit weniger Robotern. Dies ist vor allem darauf zurückzuführen, dass der Anführer ab einer Zahl von 20 passiven Robotern einen Schwarm viel schneller vollständig verliert und es keine Wartezeiten mehr gibt, weil er schneller dazu übergeht das Ziel alleine aufzusuchen. Auch die anderen beiden Diagramme zeigen nun bei 40% Gruppendrang keinen vollen Erfolg mehr.

In den letzten Diagrammen (Abbildung 6.11) ist letztlich zu sehen dass sich der allgemeine Trend weiter fort führt. Die Wartezeiten sinken noch deutlicher, weil der Anführer seinen Schwarm immer früher verliert. Die erfolgreiche Strecke nimmt genau wie die Anzahl der Roboter die im Ziel ankommen immer mehr ab. Nur bei 40% Gruppendrang bleiben die Werte noch immer recht hoch.

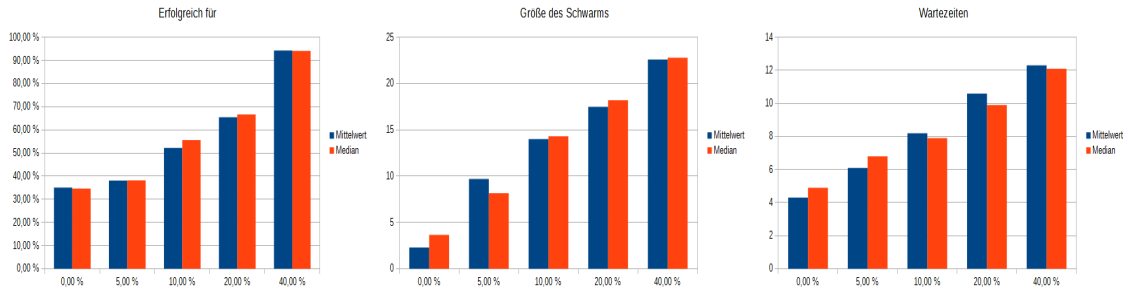


Abbildung 6.11: Erfolg eines Anführer ins Abhängigkeit der Gruppengröße und des Gruppendrangs mit 25 passiven Robotern

6.4 Transport von Waren mit Hilfe eines Schwarms

Das letztliche Ziel dieser Arbeit ist es zu prüfen, ob es möglich, und sinnvoll, ist Gegenstände mit Hilfe eines autonomen Schwarms zu transportieren. Dazu musste der Schwarm nun dazu gebracht werden Waren zu bewegen ohne die einzelnen Roboter zu sehr zu beeinflussen. Da generell die Roboter nicht zentral gesteuert werden sollen und auch die Logik möglichst simple bleiben soll, sollte das Programm der einzelnen Einheiten möglichst wenig verändert werden und der 'natürliche' Trieb der Einheiten genutzt werden.

Erteilen von Aufträgen

Eine der ersten Dinge im Ablauf eines Transports ist die Erteilung des Auftrags. Dazu wurde ein neuer Nachrichten-Typ Namens 'MissionNew' im ROS-Netzwerk eingeführt der die folgenden Felder hat:

```
uint8 robot_index_from
uint8 robot_index_to

uint8 leader_number
uint8 mission_id

float32 object_position_x
float32 object_position_y

float32 object_size_x
float32 object_size_y

float32 target_x
float32 target_y
```

Der Nachrichten-Typ definiert eine Spanne von Robotern die den Auftrag ausführen sollen. Diese werden mit ihren IDs angesprochen. Das Intervall der IDs ist, gemäß Programmierstandards, als halboffenes Intervall definiert: [robot_index_from, robot_index_to[. leader_number gibt die Anzahl der Leader an die verwendet werden sollen und mission_id gibt dem derzeitigen Auftrag eine fixe ID um diesen und zugehörige Dinge genau identifizieren und verbinden zu können. Mit object_position_x/-_y ist die Startposition des zu transportierenden Objektes angegeben. Zusammen mit object_size_x/-_y, welche die Ausmaße des Objektes angeben. Die Position des Objekts ist als Mitte des Objekts definiert. target_x/-_y gibt die letztliche Position an die das Objekt einnehmen soll.

Der Auftrag wird von außen an das Topic 'flock/mission/new' gesendet. Der Ersteller des Auftrags muss keine ROS-Node sein, auch wenn dies verschiedene Vorteile hätte. Dadurch das ROS-Topics auch über das Terminal engesprochen werden können, ist das Senden der Nachrichten auch über jedes andere Programm möglich. Das Topic für die neuen Missionen wird von jedem aktiven Roboter abonniert. Entsprechend nimmt jeder Roboter Notiz von diesem Auftrag, auch wenn er nicht direkt mit seiner ID angesprochen wird.

Von Gefahrenzonen zu Sicherheitszonen

Um die betreffenden Roboter in die Zone zu senden, auf der ihnen später das Transportobjekt aufgesetzt wird, wird ein Mechanismus verwendet der bereits vorher Einzug in das ROS-System hielt: Gefahrenzonen. Diese wurden leicht angepasst um sie invertieren zu können und somit aus einer Gefahrenzone eine Sicherheitszone zu machen. Ist ein Bereich

als Sicherheitszone definiert, ist jeder andere Bereich automatisch eine Gefahrenzone. Dieser Mechanismus birgt nur eine kleine Änderung im System, schafft es aber Roboter in einen bestimmten Bereich zu locken ohne sie aktiv steuern zu mpsse, indem einfach ihr 'natürlicher' Trieb verwendet wird, von Gefahren zu flüchten.

Wird ein Auftrag erteilt, so nehmen die Roboter die dem Auftrag zugeteilt sind, eine neue Sicherheitszone auf, die den Ausmaßen und der Position des Transportobjekts am Aufnahmeort entsprechen. Roboter die nicht dem Transport zugeteilt wurden, werden diese neue Zone als Gefahrenzone auffassen und versuchen sich diesem Gebiet fernzuhalten.

Ein Objekt kann grundsätzlich aus verschiedenen Geometrien zusammengesetzt werden. Dadurch ist es möglich nicht nur Objekte zu transportieren die eine einfache Form wie Rechtecke oder Kreise haben, sondern verschiedene Rechtecke können dann zu einem 'L' oder 'U' zusammengesetzt werden.

Implementierung Die Klassenhierarchie ist wie in Abbildung 6.12 dargestellt. Die Implementierung eines Objekts ist als Sammelklasse definiert, die verschiedene Geometrien unter sich vereint. Die Geometrien haben eine Hauptklasse von der sie sich ableiten. Die abgeleiteten Klassen müssen letztlich nur noch definieren, wann etwas innerhalb ihrer Fläche ist und was ihre Ausmaße im zweidimensionalen Raum sind. Wird ein Objekt auf eventuelle Kollisionen abgefragt, muss dieses letztlich nur noch über die inneliegenden Geometrien iterieren und sobald eine Kollision statt fand ist das Gesamtergebnis ebenfalls eine Kollision.

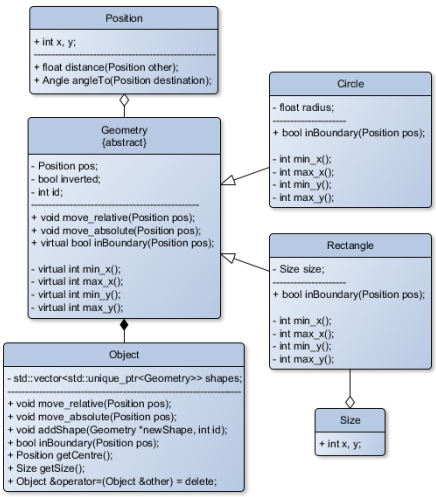


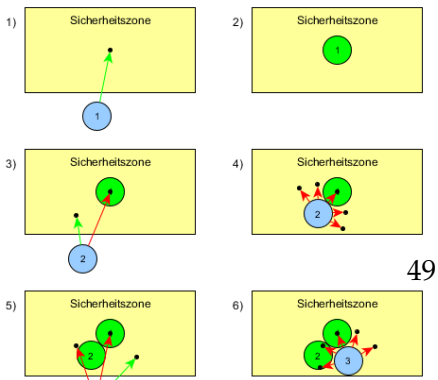
Abbildung 6.12: Die Klassenhierarchie des Objekt-Systems

Füllen eines Raums

Wird eine Sicherheitszone definiert, versuchen die Roboter, die sich in der Gefahrenzone befinden, auf möglichst schnellstem Wege die Sicherheitszone zu betreten. Dazu wird das Zentrum der Sicherheitszone ins Ziel genommen und (ohne mit anderen Robotern zu kollidieren) der direkte Weg darauf zu genommen. Dabei kann es dann allerdings vorkommen, dass eine komplexere Form nicht vernünftig gefüllt werden kann, oder dies sehr lange dauert. Da ein eintreffender Roboter so lange auf den Mittelpunkt zufahren würde, bis die Roboter darin sich genug verteilt haben um genug Platz für den neuen Roboter zu schaffen.

Ein Algorithmus wie man es schafft mit Roboter einen definierten Raum zu füllen lässt sich in (**HIER; QUELLE**) finden. Dieser Algorithmus lässt sich aufgrund anderer Fähigkeiten bei den Robotern nicht und dem gewünschten Schwarmverhalten nicht vollständig nachbilden. Eine andere, bereits implementierte Fähigkeit, lässt sich dagegen gut nutzen um den Algorithmus annähernd nachzubauen: Ausweichen.

Betritt ein Roboter die Sicherheitszone des Transport-Objekts, reduziert dieser seine Geschwindigkeit und fährt langsam weiter auf den Mittelpunkt zu. Neu ankommende Roboter werden durch die vorderen, viel langsameren Roboter zum Aus-



weichen gezwungen, was letztlich dazu führt, dass die Roboter sich seitlich verteilen, aber langfristig den Nachfolgern Platz machen. Ist ein Roboter nahe genug am Mittelpunkt angekommen oder findet in einem Umkreis von [-90°, 90°] keinen Platz mehr der näher am Mittelpunkt ist als der aktuelle, bleibt er stehen. Roboter die stehen bleiben senden den anderen Robotern ihres Schwarms ein entsprechendes Signal dass sie bereit sind. Haben alle Roboter erkannt dass die anderen bereit sind, ist die Phase des Eintreffens am Abnahmepunkt abgeschlossen.

Von hier an braucht es ein Event dass den Robotern zeigt dass sie mit dem Transportobjekt Richtung Ziel losfahren können. Möglich wäre eine Zeitsteuerung für streng automatisierte Prozesse. Aber auch interne Signale über ROS sind möglich. Durch die ID die jeder Auftrag hat, wäre eine einfache Nachricht '[TransportID#][Losfahren]' bereits zielführend.

Der Algorithmus am Beispiel Abbildung 6.13 zeigt den Algorithmus anhand eines Beispiels mit 4 Robotern.

Der erste Roboter versucht zur Mitte des Objekts zu gelangen und findet sofort Zugang zum Objekt. Er reduziert seine Geschwindkeit und platziert sich langsam im Mittelpunkt des Objektes. Da seine Nähe zum Mittelpunkt klein genug ist, bleibt er letztlich stehen.

Der nächste Roboter kommt, wird allerdings von seinem Vorgänger leicht blockiert. Das führt dazu dass dieser versucht nach links auszuweichen und damit Erfolg hat. Er betritt die Zone, findet aber keinen besseren Ort und bleibt daraufhin stehen.

Der dritte Roboter wird ebenfalls von seinen Vorgängern blockiert, findet aber rechts eine Stelle. Nachdem er diese eingenommen hat findet auch er keine bessere Stelle und bleibt ebenfalls stehen.

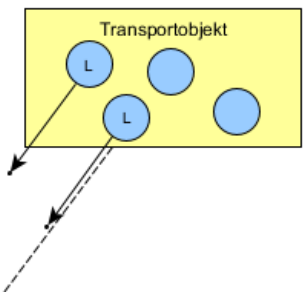
Der vierte Roboter findet zunächst links eine Stelle und betritt daraufhin die Zone. Er verlangsamt seine Bewegung und sucht nun nach Stellen die ihn ohne Kollisionen näher zu seinem Ziel führen, wobei er einmal Roboter#2 umkreist. In der Lücke zwischen Roboter#2 und Roboter#1 findet er die beste Stelle und bleibt dort anschließend stehen. Der Algorithmus für diese 4 Roboter ist damit abgeschlossen. Sie stehen alle in der Fläche des Transportobjektes und stehen still, bereit die Lieferung entgegen zu nehmen.

Der Transport

Um den Transport selbst zu realisieren bedient man sich der Hilfe der Anführer. Diese richten sich dynamisch nach dem Winkel aus, den das Transportobjekt zum Ziel hat, wie Abbildung 6.14 zeigt. Danach fangen sie an sich langsam in die Richtung zu bewegen in die sie sich ausgerichtet haben. Die anderen Roboter werden sich aufgrund des Gruppenverhaltens ebenfalls mehr oder weniger nach ihren Anführern ausrichten und das Objekt so langsam Richtung Ziel bewegen.

Einhalten der Richtung hat Priorität

Kann ein Anführer keine normale Bewegung ausführen, weil er sonst mit einem anderen Roboter



kollidieren würde, darf er nicht versuchen auszuweichen, da dies sonst die Ausrichtung der passiven Roboter negativ beeinflussen könnte. Stattdessen drosselt er zunächst sein Bewegungstempo oder bleibt, falls notwendig, ganz stehen. Die richtige Richtung beizuhalten ist wichtig, da sich passive Roboter nicht unmittelbar nach ihren Anführern ausrichten. Gerade wenn es zahlenmäßig wenige Anführer im Vergleich zu passiven Robotern sind, kann es einige Zeit dauern, bis die Einheiten so weit beeinflusst wurden, dass sie in die gewünschte Richtung zeigen. Dreht sich ein Anführer in die verkehrte Richtung, vielleicht sogar in die entgegengesetzte, kann dies zu einer Kettenreaktion führen die alle Roboter betrifft und zusätzlich das Transportobjekt stark vom Weg abbringen.

Abschluss des Transports

Abkürzungsverzeichnis

Anhang

Kolophon

Dieses Dokument wurde mit der LaTeX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 1.0). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt