Schwarmverhalten und Schwarmintelligenz mit mobilen Robotern

Studienarbeit

(T3_3201 in der 5. und 6. Theoriephase)

des Studienganges Informationstechnik

an der Dualen Hochschule Baden-Württemberg, Karlsruhe

Ersteller:	Katharina-Maria Heer, André Harbrecht
Datum:	25.04.2018
Abgabedatum:	14.05.2018
Abgabeort:	Dualen Hochschule Baden-Württemberg, Karlsruhe
Matrikelnummer:	
Kursbezeichnung DHBW:	TINF15B3
Betreuer:	Prof. Hans-Jörg Haubner

Erklärung

(gemäß §5(3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 29.09.2015)

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Titel

Schwarmverhalten und Schwarmintelligenz mit mobilen Robotern

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum	Unterschrift Studierende/r
Ort, Datum	Unterschrift Studierende/r

Abstract

Diese Studienarbeit beschäftigt sich mit dem Thema Schwarmverhalten und Schwarmintelligenz mit mobilen Robotern. Es wurden zunächst sämtliche Grundlagen ausgearbeitet und dann mithilfe von Lego NXT Robotern ein Szenario für Schwarmverhalten implementiert. Bereits am Anfang war zu erkennen, dass der Schwarm bzw. die einzelnen Agenten jedoch nur so gut sind, wie deren Sensoren. Die Sensoren der Lego NXT Roboter sind leider sehr ungenau, wodurch sämtliche Berechnungen, die als Grundlage gemessene Sensordaten verwenden, ebenfalls enorm ungenau werden. So war es beispielsweise nicht möglich, die Berechnungen der Lokalisierung in der Umgebung durchzuführen. Als Ergebnis lässt sich jedoch feststellen, dass die Realisierung eines einfachen Roboterschwarms auf der Basis des Schwarmverhaltens ohne große Probleme erfolgen kann. Eine Realisierung des Roboterschwarms auf der Basis der Schwarmintelligenz, bedarf allerdings einiger Erweiterungen, die im Kapitel *7.3 Schwarmintelligenz der Roboter* genauer beschrieben werden.

This study deals with the topic of swarm behavior and swarm intelligence with mobile robots. We first worked out all the basics and then implemented a scenario for swarm behavior using Lego NXT robots. Already at the beginning it could be seen that the swarm or the individual agents are only as good as their sensors. Unfortunately, the sensors of the Lego NXT robots are very inaccurate, which means that all calculations, that use sensor data measured as a basis, also become enormously inaccurate. For example, it was not possible to perform the localization calculations in the environment. As a result, however, it can be stated that the realization of a simple robot swarm on the basis of the swarm behavior can be done without great problems. A realization of the swarm of robots based on the swarm intelligence, however, requires some extensions, which are described in more detail in the chapter *7.3 Schwarmintelligenz der Roboter*.

Inhaltsverzeichnis

Darstellungsverzeichnis

Quellcodeverzeichnis

Formelverzeichnis

1. Einleitung

Das Schwarmverhalten können wir alltäglich bei Fischen, Vögeln oder auch Bienen beobachten. Schwärme wirken für uns Menschen sehr faszinierend und wir fragen uns, wie die Tiere eine solch große Menge an Individuen koordiniert und orientiert bewegen können. Das Verhalten der Individuen beruht lediglich auf nur sehr einfachen Verhaltensweisen, welchen den Schwarm koordiniert bewegen und teilweise intelligent erscheinen lassen. Auch gibt es in den meisten Schwärmen keine übergeordnete Instanz, welche den Schwarm lenkt, was Tierschwärme noch komplexer erscheinen lässt. Das Leben im Schwarm bringt den Individuen sehr viele Vorteile. Vor allem die Feindabschreckung bzw. die Feindverwirrung bei Fischschwärmen und die organisierte Futtersuche bei den Bienen bringen den einzelnen Individuen mehr Schutz bzw. mehr Futter. Auch haben es die Tiere nicht schwer, fortpflanzungsfähige Partner im Schwarm zu finden. Ameisen und Bienen sind echte Arbeiter, die sich die Arbeit untereinander aufteilen und verschiedene Rollen vergeben, um das übergeordnete Ziel organisiert zu erreichen. Zurückzuführen ist dies auf eine sehr hohe Form der Selbstorganisation und auf das Verhalten auf ihre Umwelt, welches die Tiere instinktiv anwenden. Das Schwarmverhalten scheint somit intelligent.

Doch das Schwarmverhalten bringt nicht nur den Tieren einige Vorteile, sondern auch uns Menschen. Forscher und Ingenieure versuchen, die herrschenden Regeln in den Tierschwärmen zu nutzen, um künstliche Roboterschwärme abbilden zu können. Diese Roboterschwärme sollen verschiedenste Aufgaben, die man nur in einem Schwarm effizient und vorteilhaft bewältigen kann, für uns Menschen lösen. Vor allem gegenüber den konventionellen Lösungsverfahren sollen Roboterschwärme wesentlich mehr Vorteile mit sich bringen.

Diese Studienarbeit beschäftigt sich mit der Implementierung eines kleinen Roboterschwarms, mithilfe von Lego NXT Robotern. Hierzu sollen Verhaltensweisen der Tierschwärme genutzt und effizient auf verschiedene Problemstellungen adaptiert werden.

2. Schwarmverhalten

Ein Schwarm bezeichnet eine große Menge an Tieren, die der gleichen Tierart angehören. Viele Tiere, wie z.B. Bienen oder Ameisen können nur durch ihre gemeinsame Lebensweise im Schwarm überleben und so Aufgaben und Herausforderungen bewältigen, zu denen die Tiere einzeln nicht in der Lage wären.

Das Schwarmverhalten bedeutet, dass sich Tiere zusammenschließen. Dieses Verhalten beruht allerdings nicht auf soziale Bedürfnisse des Individuums, sondern eher auf dessen Instinkt. Das Bilden eines Schwarms hilft den Individuen, sich vor größeren Fressfeinden zu schützen, da hier der Feind durch stetige Wachsamkeit schneller erkannt und durch die Größe des Schwarms eventuell abgeschreckt werden kann. Des Weiteren kann so, wie bei den Bienen, die Nahrungssuche und deren Beschaffung effizienter gestaltet und ausgeführt werden. Auch eine verbesserte Anpassung an die Umwelt wird durch die Bildung eines Schwarms erreicht.[1]

Für den Außenstehenden wirken manche Schwärme sehr ungeordnet und unübersichtlich, dabei gibt es drei Grundverhaltensregeln der Individuen damit sich der Scharm als Einheit und nicht als Chaos fortbewegen kann:

- Kohäsion:

  Ein Individuum bewegt sich immer in Richtung Zentrum des Schwarms, sprich zur durchschnittlichen Position. Diese Position wird jedoch nur von den in unmittelbarer Nähe befindlichen Nachbarn eines Individuums ermittelt.[2] So schwimmen z.B. Heringe nicht einfach in die Schwarmmitte, sondern in die Mitte ihrer Nachbarn.

[1] Vgl. Spektrum Akademischer Verlag (2001).

[2] Vgl. Sedlacek, Klaus-Dieter (2010), S.110.

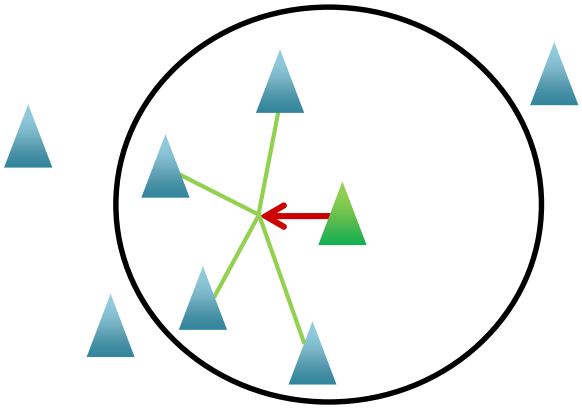

Abbildung 1: Kohäsion

- Separation:

  Damit durch die Kohäsion Zusammenstöße der Individuen vermieden werden kann, besagt die Separation, dass die Individuen untereinander Abstand halten und gegebenenfalls die Bewegungsbahnen verändern, falls sich ein anderes Individuum in die eigene Bahn bewegt.[3]

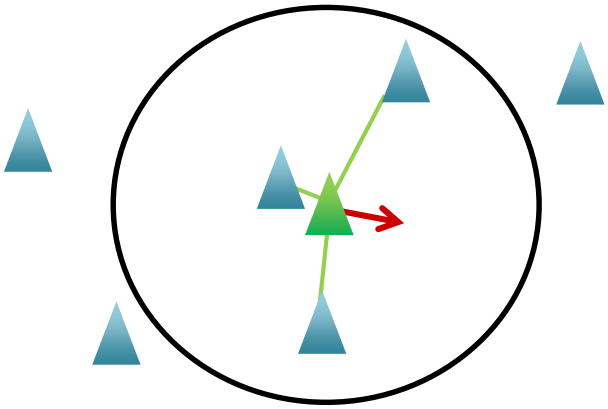

Abbildung 2: Seperation

[3] Vgl. Sedlacek, Klaus-Dieter (2010), S.110.

- Alignment:

  Damit sich der Schwarm nun koordiniert fortbewegen kann, ist das Alignment nötig. Es beschreibt, dass sich das einzelne Individuum in dieselbe Richtung ausrichtet und bewegt wie dessen Nachbarn.[4]

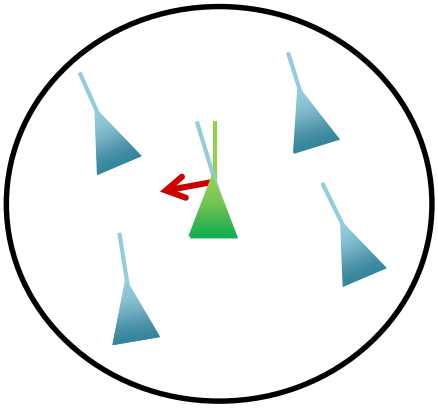

Abbildung 3: Alignment

Durch die Natur lassen sich vielerlei Lösungsmöglichkeiten für heutzutage noch offene Probleme ableiten. Dabei werden nicht nur die körperlichen Gegebenheiten oder Verhaltensweisen einzelner Individuen betrachtet, sondern auch das Zusammenspiel mit ihren Artgenossen. Dieses, nach außen hin intelligent aussehende Verhalten der Tiere, wie beispielsweise bei den Ameisen, wird beobachtet, um Verhaltensmuster oder Regeln ableiten zu können, welche Lösungsansätze für heutige Probleme darstellen könnten.

Das Verhalten der einzelnen Tiere basiert hierbei auf genetisch festgelegten, simplen Reizen. Ein Individuum empfängt einen Reiz und reagiert darauf, ohne weiter darüber nachzudenken. Es handelt demnach nach einfachen Regeln, welche erst in der Gruppe komplex wirken. Wissenschaftler versuchen diese einfachen Regeln der einzelnen Individuen mithilfe von Tests zu ergründen, zu formalisieren und als Modell abzubilden.

[4] Vgl. Sedlacek, Klaus-Dieter (2010), S.111.

Mithilfe der gesammelten Informationen können diese Regeln formuliert und anschließend in Simulationsmodelle übertragen werden, welche überprüfen, ob diese der Realität entsprechen oder nicht. Dies geschieht beispielsweise bei Ameisen, wie im Kapitel *2.1.1 Ameisen* beschrieben. Ein Beispiel dafür, dass diese Methodik funktioniert ist das 1989 von Dorigo entwickelte Ant-Colony-System (siehe Kapitel *2.2.1 Ant Colony System*).

Dieser Theorie nach sollte man meinen, dass mit Kenntnis der simplen Regeln, das Verhalten der einzelnen Individuen (hier Ameisen) vorhersehbar ist. Das ist allerdings nicht der Fall. Bei intensiverer Betrachtung fällt ein des Öfteren undefiniertes Verhalten auf. Zu Beweisen ist dies indem einzelne Individuen bestimmten Reizen ausgesetzt werden und deren Verhalten darauf beobachtet wird. Zum einem kommt es vor, dass das Individuum sich wie erwartet verhält und auf die Reize reagiert, zum anderen kann auch ein Verhalten beobachtet werden, das nicht dem ausgesetzten Reiz entspricht. Dieses zufällige Verhalten wird als randomisiertes Verhalten beschrieben. Eine Ameise, die auf der Suche nach neuen Futterquellen ist, kann somit nicht auf vorhandene Reize reagieren, da entweder keine vorhanden sind, weil die Futterquelle noch unentdeckt ist, oder die Futterquelle ansonsten schon bekannt wäre. Somit sucht die Ameise die Umgebung willkürlich ab, in der Hoffnung etwas zu finden. Laut Kai Bornemann wird dieses Verhalten in der Biologie auch Exploration genannt. Es ist für das Überleben des Staates von wesentlicher Bedeutung, da die gefundene Futterquelle nach einer gewissen Zeit aufgebraucht ist. Somit sichert dieses Verhalten das Überleben des Staates, da es immer Individuen gibt, die trotz der aktuell vorhandenen Reize andere Wege einschlagen, um neue und bessere Futterquellen zu erschließen.[5] Wie die Ameise sich entscheidet ist folglich nicht sicher vorhersehbar, sondern lediglich zu einer bestimmten Wahrscheinlichkeit, die bei auftretenden Reizen steigt. Wenn die einzelne Einheit auf mehrere Reize trifft, muss sie sich entscheiden, welchem sie nachgibt. Somit sind die

[5] Vgl. Bornemann, K. (o.J.).

Muster, die beobachtet werden können ein Zusammenspiel aus einfachen Regeln und zufälligen Entscheidungen, die von den einzelnen Einheiten getroffen werden. Hieraus lässt sich auch die Effizienz des Verhaltens ableiten, die auch in der Wissenschaft eingesetzt wird. Wenn die Ameise nicht auf die Reize des Schwarms reagieren würde, wäre sie auf sich alleine gestellt und müsste jedes Mal aufs Neue für sich eine Futterquelle suchen. Wenn sie andererseits immer auf die Reize reagieren würde, gäbe es keine Möglichkeit neue Futterquellen zu finden, die sogar besser sein könnten. Der Schwarm würde sich also nicht weiterentwickeln. Wenn Individuen also auf Reize reagieren, nutzen sie das geteilte Wissen anderer Individuen. Dieses Verhalten nennt Bornemann im Zusammenhang mit dem Begriff Exploitation.[6] Sowohl die Exploration als auch die Exploitation bestimmen den Erfolg zur Optimierung des Schwarmverhaltens. Die Entstehung der Reize lässt sich auf die Kommunikation zwischen den Einheiten zurückführen. Man kann hierbei zwischen zwei Arten der Kommunikation unterscheiden. Die direkte oder die indirekte Kommunikation. Honigbienen geben ihr Wissen über gefundene Nahrungsquellen über den Schwänzeltanz oder den Rundtanz weiter (siehe *Kapitel 2.1.2 Bienen*), während die von Ihren Artgenossen dabei beobachtet werden. Somit geben sie ihr Wissen direkt weiter. Bei der indirekten Kommunikation hingegen werden Reize hinterlassen, die zu einem späteren Zeitpunkt von anderen Individuen erkannt werden, wie bei Ameisen, die eine Pheromonspur hinterlassen (siehe Kapitel *2.1.1 Ameisen*). Die Individuen reagieren also nicht aufeinander, sondern auf ihre Umwelt. Beide Kommunikationsarten haben hierbei Vor- und Nachteile. Während die Pheromonspur mit der Zeit schwächer wird und verschwindet, wird der Schwänzeltanz nur von unmittelbar Beteiligten wahrgenommen. Zusammengefasst lässt sich das intelligente Verhalten von Schwärmen auf simple Entscheidungen zurückführen, die meist ohne Überlegung durch Reaktionen auf die Umgebung geschehen. Dennoch weist dieses Verhalten eine sehr

[6] Vgl. Bornemann, K. (o.J.).

hohe Effizienz auf und könnte somit, wenn sie richtig genutzt, interpretiert und umgesetzt wird, die Lösung für viele Probleme sein.[7] Einige dieser Lösungsansätze, welche vom Schwarmverhalten in der Tierwelt inspiriert wurden, werden in *Kapitel 2.2. Schwarmverhalten in der Technik* näher beschrieben.

2.1 Schwarmverhalten in der Tierwelt

Schwarmverhalten findet sich in der Natur in zahlreichen Beispielen wie der Nahrungssuche von Ameisen oder dem Fluchtverhalten von Heringen wieder. Durch viele Individuen, die alle einfachen Regeln gehorchen wird ein komplexes Verhalten simuliert, welches in der Lage ist sich selbst zu organisieren und anzupassen.

2.1.1 Ameisen

Das Schwarmverhalten von Ameisen lässt sich mit einem einfachen Test veranschaulichen, in dem es zwei Wege zu einer Futterquelle gibt, wobei der eine deutlich kürzer ist. Es fällt auf, dass zu Beginn beide Wege gewählt werden, da noch nicht bekannt ist, welcher der kürzere ist. Wenn allerdings die ersten Ameisen zurückkommen, haben sie eine Pheromonspur gelegt, welche beim kürzeren Weg folglich intensiver ist als beim längerem. Somit reagieren die Ameisen auf diesen Reiz der intensiveren Pheromonspur, indem sie vermehrt den kürzeren Weg einschlagen.[8] In Abbildung 4 sieht man wie dies beispielsweise aussehen kann. Eine einzelne Ameise findet eine Futterquelle geht einen zufällig ausgewählten Weg zu ihrem Nest zurück, da es keinen Einfluss von Reizen gibt, denen sie folgen könnte. Währenddessen hinterlässt sie eine Pheromonspur. Das setzen der Pheromonspur verläuft dabei automatisch, ohne dass die Ameise es bewusst wahrnimmt. Andere Ameisen folgen dieser einen nun auf vier möglichen Pfaden und hinterlassen dabei selbst eine Pheromonspur. Je mehr Ameisen einen der vier

[7] Vgl. Bogon, Tjorben (2012), S. 35ff.

[8] Vgl. Serban,N. (2017), S.2.

Wege einschlagen, desto intensiver wird die Spur und desto wahrscheinlicher wird es, dass andere Ameisen diesen Weg auch einschlagen. Da der kürzeste Weg nun am schnellsten zu überwinden ist, wird die Pheromonspur hier immer intensiver, sodass immer mehr Ameisen diesen Weg einschlagen, der sich als kürzester herausgestellt hat.

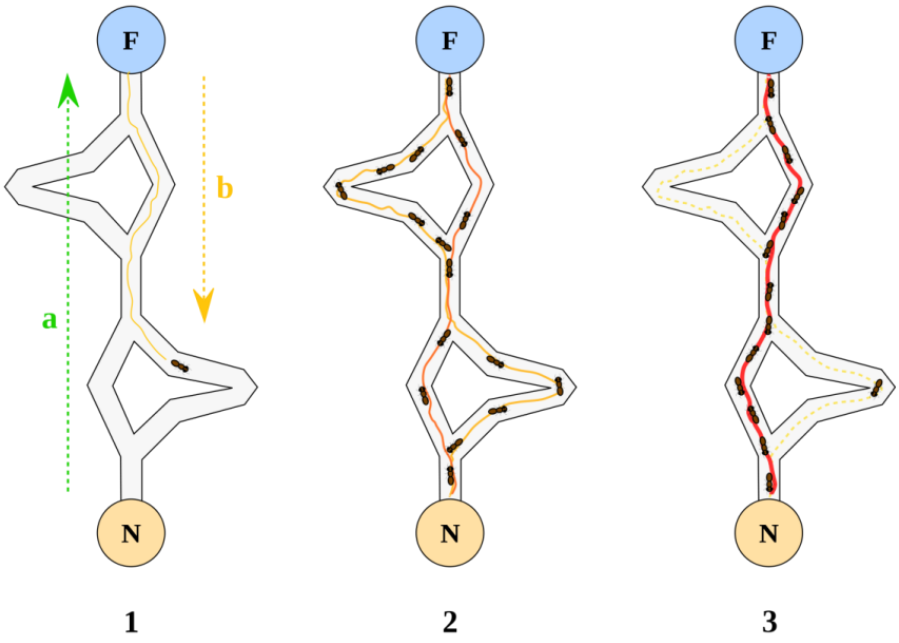

Abbildung 4: Wegfindung bei Ameisen (Vgl. Johann Dréo (2006))

In Abbildung 5 ist ein weiteres Beispiel dargestellt, wie Ameisen auf aufkommende Hindernisse reagieren. Die grüne Linie stellt die intensivste Pheromonspur dar, der die Ameisen wie im ersten Bild oben links folgen. Wenn ein plötzlich aufkommendes Hindernis über der Spur auftritt versuchen die Ameisen dieses zu umgehen, indem sie zufällig bestimmte Wege einschlagen, Stichwort Exploration. Währenddessen hinterlassen diese weiterhin eine Pheromonspur, die wieder nach einer kurzen Zeit beim kürzeren Weg intensiver wird, sodass nach einiger Zeit fast alle Ameisen diesen Weg einschlagen. In dieser Abbildung wird sehr gut verdeutlicht, wie feinfühlig die Ameisen die Pheronspur erkennen und folgen können.

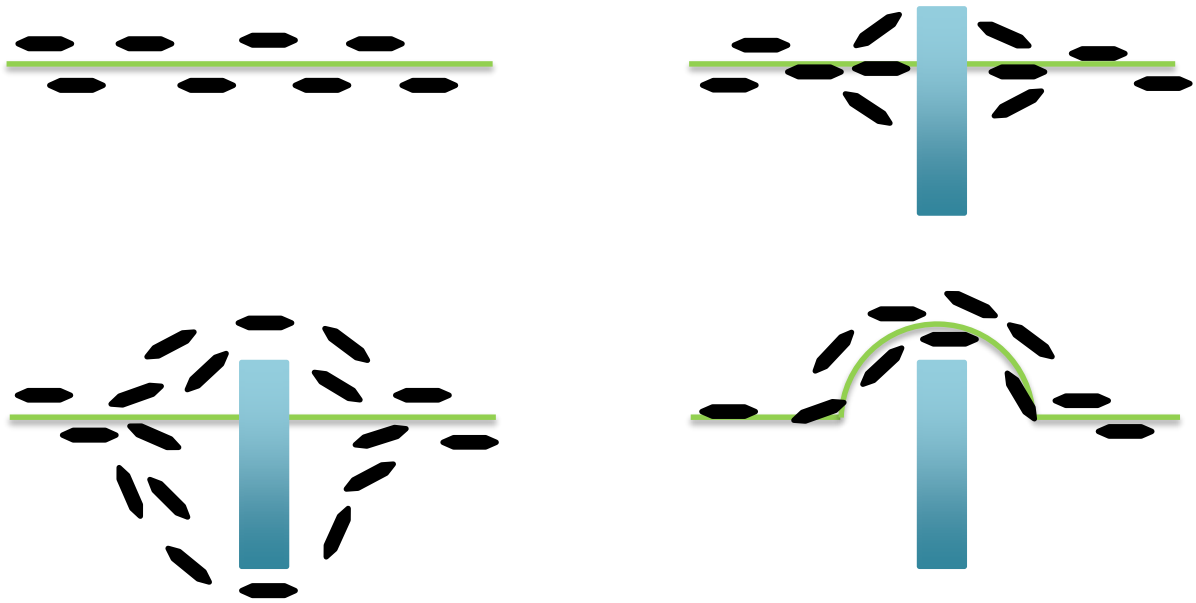

Abbildung 5: Wegfindung bei Ameisen

Ameisen zeigen durch ihr primitives aber nach außen hin intelligent erscheinendes Verhalten, wie einfach es sein kann, in einem so großen Schwarm zu leben, welcher sich selbst organisiert. Der Fachbegriff hierfür lautet Stigmergie. Aus diesem Grund ist die Anpassungsfähigkeit der Ameisen enorm. Es wird angenommen, dass die Ameisen ca. 40% der Biomasse aller Insekten darstellen. Zusammen mit den Termiten sollen diese, wenn man den Menschen nicht mit einberechnet, 25% der gesamten tierischen Biomasse ausmachen. Außerdem können, je nach Art, bis zu 20 Millionen Ameisen in einem Ameisenstaat leben. [9] Dies zeigt, dass sich das Leben im Schwarm als großer Vorteil für den Existenzkampf sowie für die Anpassungsfähigkeit entpuppt.

[9] Vgl. Spektrum Akademischer Verlag (1999).

2.1.2 Bienen

In einem Bienenschwarm leben bis zu 40.000 Individuen.[10] Sie haben, wie die Ameisen, verschiedene Aufgaben im Kollektiv. Es gibt unter anderem Sammlerinnen und Kundschafterinnen. Letzteren legen viele Kilometer auf der Suche nach Nahrung bzw. Pollen zurück. Diese fliegen in zufällige Richtungen, ohne einen Plan zu haben, sprich ohne die Umgebung systematisch abzusuchen. Findet eine Kundschafterin eine vielversprechende Nahrungsquelle, kehrt sie umgehend zum Bienennest zurück. Dort angekommen, vollführen sie einen der beiden, unter *2. Schwarmverhalten* genannten, Schwänzeltänze. Über diese übermitteln sie den anderen Bienen, in erster Linie den Sammlerinnen, einige Informationen über die Lage und Qualität der Futterquelle. Der erste Tanz ist der sogenannte Rundtanz (vgl. Abbildung 6). Laut einem Bienen-Blog wird dieser vollführt, wenn die Futterquelle max. 100 Meter vom Nest entfernt ist. Informationen über Lage dieser werden hier nicht weitergegeben.

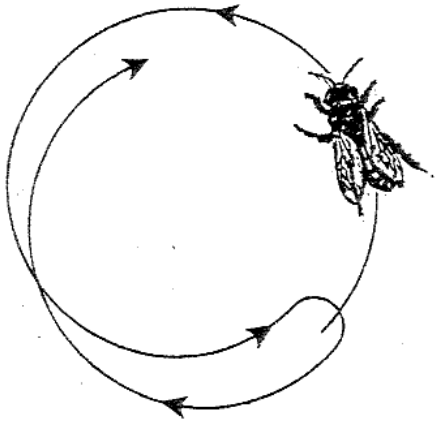

Abbildung 6: Rundtanz einer Kundschafterin (Vgl. Bienengarten Ahrensburg (2018))

Wurden die Sammlerinnen von dem Tanz überzeugt, bewegen diese sich auf die entsprechende Kundschafterin zu und riechen an ihr den Duft der Blume, die die Kundschafterin entdeckt hat. Mithilfe des Geruchs können sich die Sammlerinnen dann ori-

[10] Vgl. Bezirksimkerei Metzingen e.V. (2017).

entieren und die Blume in der Nähe des Nestes ausfindig machen. Der Schwänzeltanz kommt zum Einsatz, wenn die Futterquelle mehr als 100 Meter vom Bienenstock entfernt ist (vgl. Abbildung 7). Hier werden im Vergleich zum Rundtanz, Informationen über die Entfernung und Richtung der Quelle über die direkte Kommunikation weitergegeben.[11]

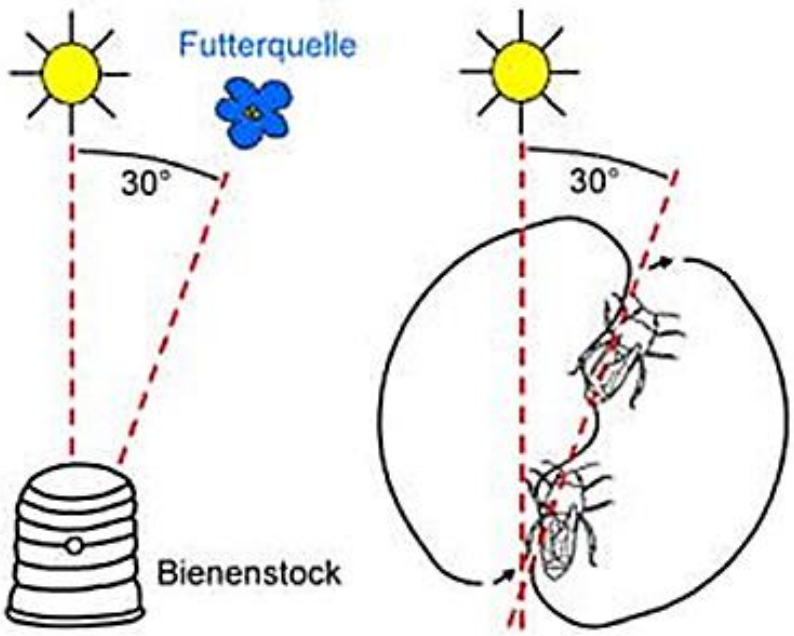

Abbildung 7: Schwänzeltanz einer Kundschafterin (Vgl. Thomas Seilnacht (2013))

In Abbildung 7 ist der Ablauf des Schwänzeltanzes sehr gut zu erkennen. Die Biene läuft auf einer geraden Linie und schwänzelt dabei mit ihrem Hinterteil. Dann macht sie eine Bewegung nach links, läuft wieder schwänzelnd auf einer geraden Linie und macht dann eine Bewegung nach rechts und wiederholt danach den kompletten Tanz beliebig oft. Je nachdem, wie weit die Futterquelle entfernt ist, führt die Kundschafterin diesen Tanz kürzer oder länger aus. Wie die Kundschafterinnen genau die Entfernung bestimmen, galt lange Zeit als Mysterium. Laut einem Artikel der Fachzeitschrift *Science* der

[11] Vgl. honig-und-bienen.de (o.J.).

American Association for the Advancement of Science, wurden einige Versuche durchgeführt, die zeigten, dass die Bienen aufgrund ihres Energieverbrauchs die Entfernung angeben. Je mehr Energie die Bienen verbraucht haben, desto länger die Entfernung. Die Forscher führten jedoch einen neuen Versuch durch, der die bisherigen Forschungsergebnisse widerlegte. Hierfür wurde eine Futterquelle 6 Meter von einem Bienenstock aufgestellt. Die Bienen gelangen nur durch ein 6 Meter langes Rohr zu dieser. Das Rohr wurde auf der Innenseite mit unterschiedlich großen weißen und schwarzen Quadraten bebildert, was einem riesigen QR-Code ähnelte. Zurückkommende Kundschafterinnen tanzten den Schwänzeltanz. Also waren diese der Meinung, dass die Futterquelle über 100 Meter vom Bienenstock entfernt ist. Als nächstes Experiment wurde das Bild im Rohr verändert, sodass dies ein paralleles Muster zur Fluglinie der Bienen aufwies. Die ankommenden Kundschafterinnen tanzten dann den Rundtanz. Sie hielten die Position der Futterquelle also recht nahe am Stock liegend. Dies zeigte den Forschern, dass die Bienen das vorbeiziehende Bild der Landschaft als Kilometerzähler nutzen.[12]

Wirkt der Tanz sehr aufgeregt und hektisch, können die anderen Sammlerinnen daraus schließen, dass es sich um eine sehr ergiebige Futterquelle handelt. Somit ist die Information über die Entfernung der Futterquelle direkt weitergegeben. Die Information über die Richtung, in der die Quelle liegt, wird über die Richtung des Tanzes weitergeben. Die Richtung bezieht sich dabei immer auf den Stand der Sonne. In Abbildung 7 befindet sich die Futterquelle 30° Richtung Osten, in Abhängigkeit zu der Sonne. Für die Bienen ist also nur wichtig, um wie viel Grad versetzt sich die Futterquelle rechts oder links von der Sonne weg befindet. Die Sammlerinnen sehen sich die Tänze der der Kundschafterinnen an und entscheiden dann, zu welcher Futterquelle sie fliegen. Das erstaunliche an Bienen ist, zum einen, wie sich diese Art der Kommunikation über viele Jahre entwickelt hat, zum anderen, dass sie über weitere Kommunikationsarten verfü-

[12] Vgl. Mandyam V. Srinivasan, et al. (2000).

gen. Sie verfügen nicht nur über die direkte Kommunikation mithilfe der Tänze, sondern auch über indirekte Kommunikation. Beispiele hierfür wären Pheromone. Ein Fallbeispiel: Müssen die Wächterinnen Eindringlinge abwehren, setzen die Duftstoffe ab, die die anderen Bienen alarmieren. Die sich in der Umgebung befindlichen Bienen reagieren somit auf ihre veränderte Umwelt und riechen den Alarm. Sie eilen dann den Wächterinnen zur Hilfe. Ein dazugehöriges weiteres Beispiel ist, dass die Königin selbst Duftstoffe produziert, die sich durch die Bienen im ganzen Bienenstock und auch auf den Bienen selbst verteilt. Hierdurch riechen alle Bienen vom selben Stock gleich. Kommt nun eine Kundschafterin zu ihrem Stock zurück, riechen die Wächterinnen, dass diese Biene zum Bienenstock gehört.[13] Auch dieses Beispiel bezieht sich auf die indirekte Kommunikation der Bienen. Man kann erkennen, dass Bienen selbiges Prinzip wie die Ameisen nutzen: Kommunikation über Pheromone. Außerdem nutzen sie zusätzlich die Form der direkten Kommunikation, durch die Tänze, die im Reich der Insekten einzigartig sind. Das Schwarmverhalten, bzw. die Schwarmintelligenz kommt bei den Bienen zum Tragen, indem eine gewisse Streuung der Angaben von den ankommenden Kundschafterinnen existiert. Diese Streuung tritt immer dann auf, wenn mehrere Kundschafterinnen dieselbe Futterquelle gefunden haben und die Entfernung unterschiedlich angeben. Dies hat den Grund, dass die Entfernung nicht immer die gleiche ist, da evtl. nicht der direkte Weg genommen wurde. Die Sammlerinnen können diese Streuung Mitteln und finden so meist immer den exakten Ort der Futterquelle.[14] Eine genaue Erklärung einer solchen Mittelung anhand einiger Versuche bzw. der Schwarmintelligenz am Beispiel des Menschen, finden Sie im Kapitel *3. Schwarmintelligenz.*

[13] Vgl. honig-und-bienen.de (o.J.).

[14] Vgl. Imkerpate.de (2014).

2.1.3 Fische

Bei den Fischen findet man sehr häufig riesige Schwärme. Egal ob im Salz- oder Süßwasser, das Leben im Schwarm bietet den Fischen meist nur Vorteile. Auch sie stehen seit vielen Jahrzehnten im Mittelpunkt der Schwarmforschung. Es war sehr lange unklar, wie sich so ein Schwarm überhaupt fortbewegt, geschweige denn wer diesen leitet. Damit sich ein solch kompliziert wirkender Schwarm fortbewegen kann, brauch man nur die unter *2. Schwarmverhalten* erklärten Verhaltensregeln: Kohäsion, Separation und Alignment. Hiermit sind die Grundbausteine für das fortbewegen im Schwarm zwar sichergestellt, aber es stellt sich weiterhin die Frage, wer dem Schwarm die Richtung vorgibt. Hierzu wurde, laut einem Artikel in der Zeitschrift *Der Spiegel,* von Jens Krause, einem Verhaltenswissenschaftler, ein Großexperiment mit 200 Menschen in einer Kölner Halle gestartet. Die Regeln hierfür waren identisch mit den drei oben genannten. Sie durften nicht miteinander kommunizieren, mussten ständig in Bewegung bleiben und einen bestimmten Abstand zu den benachbarten Teilnehmern einhalten. Auch durften sie sich in der großen Halle frei bewegen. Zu sehen war hier, dass nach einiger Zeit die Teilnehmer in einem großen Kreis durch die Halle liefen. Es bildete sich auch ein Kreis im Kreis, der in entgegengesetzter Richtung verlief.[15] Dieses Verhalten weisen ebenfalls Barrakudas auf. So konnte der Forscher viele Ähnlichkeiten zwischen Mensch und Fisch erkennen. Ein Schwarm benötigt also zuallererst einige Individuen, um als Schwarm zu gelten. Wie diese sich fortbewegen ist durch die 3 Grundregeln festgelegt, die durch das eben genannte Experiment bestärkt worden sind. Der Schwarm bewegt sich also nicht als komplexes Konstrukt, sondern nach einfachen Regeln, die im Instinkt der Fische enthalten sind. Im nächsten Abschnitt des Experiments wurden 5 der 200 Menschen angewiesen in eine bestimmte Richtung zu gehen, um den Schwarm so zu beeinflussen. Dies funktionierte nicht, weswegen hierfür 10 Menschen auserwählt wur-

[15] Vgl. Dambeck, H. (2007).

den. Erst dann gelang es, den Schwarm zu lenken[16]. Somit sind dies mindestens 5% des menschlichen Schwarms, die den ganzen Schwarm lenken können. Doch warum 5% und funktioniert das auch bei Fischen? Ein weiterführender Versuch von Jens Krause zeigt, dass dieser Versuch auch auf einen Fischschwarm anwendbar ist. Da er den Fischen keine Anweisung geben kann, führte er das Experiment mithilfe von Roboterfischen durch, die vom Schwarm akzeptiert wurden und mithilfe von Magneten im Becken bewegt wurden. Es zeigt sich, dass dieser Wert einen gewissen Schwellenwert darstellt, ab diesem der Schwarm den Entscheidungen der 5% Folge leistet, ohne zu denken, es sei ein Irrtum eines einzelnen Individuums. Liegt die Anzahl der Individuen unter 5%, so folgt der Schwarm nicht.[17] Dies bestätigt, dass tatsächlich mindestens 5% des Schwarms nötig sind, um diesen zu lenken. Diese 5% setzen sich instinktiverweise aus den Individuen im Schwarm zusammen, die zuerst auf die Reize ihrer Umwelt reagieren. Es müssen also nicht zwangsläufig die leistungsfähigsten und ausdauerndsten Individuen sein, die, laut einem Artikel des Wissenschaftsmagazins *Proceedings oft he Royal Society,* an der Spitze des Schwarms zu finden sind. Dies hat den Vorteil, dass diese mehr fressen bekommen als der restliche Schwarm. Die leistungsschwächeren Individuen befinden eher weiter hinten, um sich hier im Strömungsschatten der davor schwimmenden Fische zu befinden und um somit weniger Energie verbrauchen zu müssen[18]. Betrachtet man den Schwarm als Gesamtbild, sieht so ein Richtungswechsel sehr koordiniert und synchronisiert aus. Dieser gleichzeitige Richtungswechsel basiert überraschenderweise ebenfalls auf den oben genannten 3 Grundregeln des Schwarmverhaltens. Sobald mindestens 5% der Tiere, z.B. aufgrund von einem Angriff eines Raubfisches, eine andere Richtung einschlagen als der Schwarm, folgen leicht verzögert deren Nachbarn und daraufhin wiederum deren Nachbarn usw. (vgl. Abbildung 8).

[16] Vgl. Ebenda S.15.

[17] Vgl. Stober, A. (2016).

[18] Vgl. S. Killen, S., et al. (2011).

Laut Krause ändert sich die Bewegung der Fische Impulsartig binnen weniger Millisekunden. Dies konnte er nur mithilfe einer Hochgeschwindigkeitskamera filmen und danach auswerten.[19]

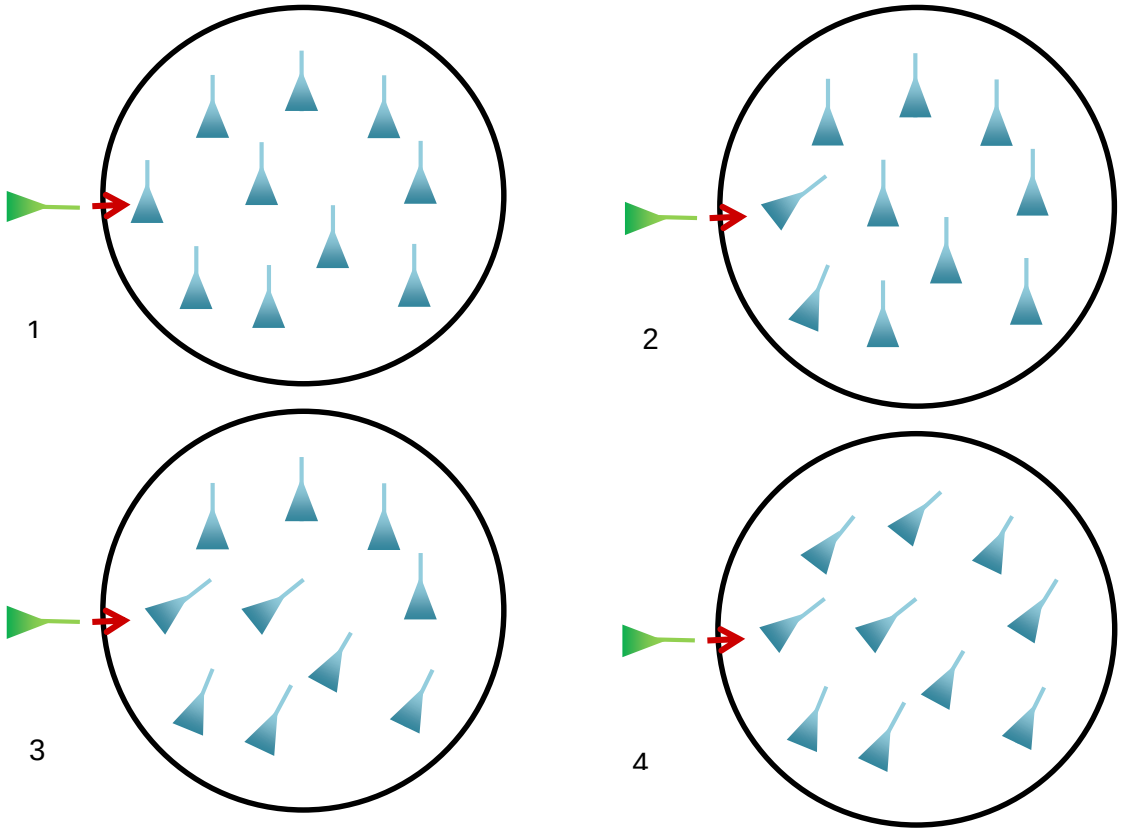

Abbildung 8: Richtungswechsel des gesamten Schwarms

In Abbildung 8 ist der Räuber (grüner Pfeil) der Grund, warum die Fische (blaue Pfeile) schlagartig ihre Richtung wechseln. Zu sehen ist in 4 Schritten, wie die Individuen Impulsartig die Richtung ändern. Dieser Vorgang ist ohne Hilfsmittel mit dem bloßen Auge nicht zu erkennen, da die Tiere mithilfe ihres Instinkts und ihrem sehr empfindlichen Seitenlinienorgan sehr schnell reagieren können. Aus dieser Reihe von Experimenten kann man Schlussfolgern, dass hier, wie auch beim großen Ameisenstaat, eine hohe instinktive Art der Selbstorganisation vorliegt, der lediglich 3 Verhaltensregeln als

[19] Vgl. Axel Springer SE (2009).

Grundlage dienen. Es wurden ebenfalls viele Parallelen zu den Menschen aufgezeigt, wodurch man sagen kann, dass Menschen auch ein gewisses Maß an Schwarmverhalten besitzen. Als kleines Beispiel dient hier eine Szene in einem Flughafen. Viele steigen aus dem Flugzeug aus und folgen denjenigen, die zielgerichtet Richtung Ausgang laufen, ohne selbst aktiv nach einem Ausgang zu suchen. Ein weiteres Beispiel wäre auch, dass sich Menschen in einem Brandfall an einen Ausgang drängeln, obwohl es weitere Ausgänge gibt, die so gut wie frei sind. Es gibt noch viele weitere solcher Fälle, die bestätigen, dass Menschen ein ähnliches Schwarmverhalten wie Fische aufweisen.

Durch den Schwarm können Fische schneller Fressfeinde ausfindig machen und auf diese reagieren, da bekanntlich viele Individuen mehr sehen als ein einzelnes. Fischschwärme repräsentieren, genau wie die Ameisen und Bienen, dass mithilfe weniger Regeln ein komplex wirkender und agierender Superorganismus entsteht, der Probleme löse kann, wozu ein einzelnes Individuum definitiv nicht in der Lage wäre. Laut der Definition für Schwarmverhalten und dessen drei Grundregeln, stellt, im Vergleich zu den Ameisen und den Bienen, der Fischschwarm somit das passendste Paradebeispiel eines Tierschwarms dar.

2.2 Schwarmverhalten in der Technik

In Kapitel *2.1 Schwärme in der Tierwelt* wurden verschiedene Verhaltensmuster aus Beispielen in der Natur betrachtet und analysiert. Die Tierwelt hat es geschafft mithilfe einfacher Regeln ein scheinbar komplexes Verhalten zu simulieren, in welchen das einzelne Individuum erst im Zusammenspiel mit seinen Artgenossen schlau erscheint. Einige Wissenschaftler haben es bereits geschafft dieses Phänomen aufzugreifen und in der Technik zu nutzen.

Das Kapitel umfasst somit nicht nur Beispiele von wirklichen Schwärmen in der Technik, wie in Kapitel *2.2.4 S-Bots*, sondern auch solche, wie ein Schwarmverhalten aus der Natur in der Technik beispielsweise durch einen davon inspirierten, erstellen Algorithmus nutzbar gemacht werden konnte.

2.2.1 Ant Colony Optimization Algorithm (ACO)

Der durch das natürliche Beispiel der Ameisen basierende (siehe *2.1.1 Ameisen*) Ameisenalgorithmus überführt ein beliebiges Problem in einen Graphen. Hierbei wird das Problem der Wegfindung behandelt, in dem der kürzeste Weg von A nach B herausgefiltert wird. So wie die unterschiedlichen Wege, die die Ameise einschlagen kann, werden auch die verschiedenen Lösungswege im Graph dargestellt und gewichtet.[20] Ziel ist es, den Weg zum Ziel, unter der Berücksichtigung der minimalen Kosten, zu finden. Hierzu werden zyklisch virtuelle Ameisen über den Graphen laufen gelassen. Am Anfang haben sämtliche Wege keine Pheromonspur. Nach einem Zyklus werden die Pheromonwerte entsprechend angepasst, wenn Ameisen über die Wege gelaufen sind. Auch werden die Pheromonkonzentrationen der Wege reduziert, wenn diese nicht zyklisch belaufen werden.

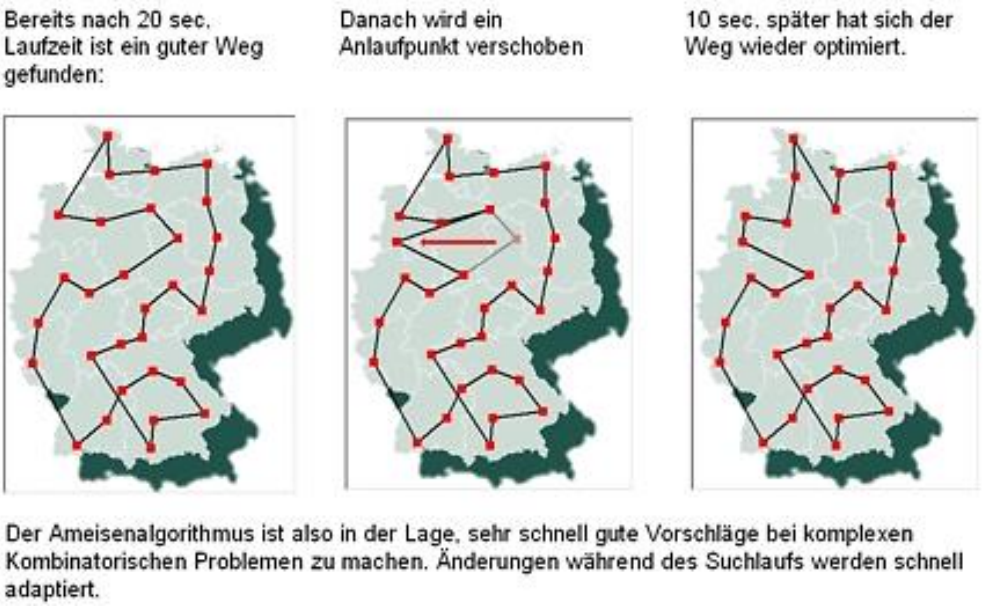

Abbildung 9: Problem des Handlungsreisenden mit ACO (Vgl. Hans-Ulrich Mährlen (2004))

[20] Vgl. Dorigo, M. / Stützle, T. (2004).

Der Ameisenalgorithmus kann schon jetzt für einige Problemstellungen genutzt werden, wie z.B. für das Problem des Handlungsreisenden (TSP). Dies ist ein Optimierungsproblem, dessen Aufgabe darin besteht, den kürzesten Weg des Handlungsreisenden für den Besuch mehrerer Städte zu finden (vgl. Abbildung 9). Weitere Anwendungsmöglichkeiten für diesen Algorithmus wären unter anderem Routenplanung und Routenoptimierung für Transportdienste.

2.2.2 Particle Swarm Optimization Algorithm (PSO)

Die Partikelschwarmoptimierung orientiert sich, wie der oben genannte Ameisenalgorithmus an Tierschwärmen. Vor allem aber an Fisch- und Vogelschwärmen. Die Grundregeln eines Schwarms (Kohäsion, Alignment und Separation) werden hier auf ein Problem angewandt, in dem sich die einzelnen Schwarmpartikel (auch Boids genannt), in einem, an das Problem angepassten, Suchraum bewegen, um es dadurch zu lösen. [21] So wie die Schwarmtiere selbst, unterliegen die Partikel auch einigen Regeln bzw. besitzen vorausgesetzte Eigenschaften.

Jedes Partikel:[22]

- besitzt einen zufälligen Startpunkt im Suchraum
- bewegt sich frei im Raum und besitzt daher eine Geschwindigkeit
- sucht das Optimum, sprich die beste Position
- erinnert sich an seine eigene beste Position und den Funktionswert an dieser
- kennt seine Nachbarn und deren Funktionswerte
- kennt die beste globale Position und den dazugehörigen Funktionswert

[21] Vgl. Bäcker, B., et al. (2016), S.9

[22] Vgl. Liu, Yushan (2014), S.2

Da es sich hier um ein matheheuristisches Verfahren handelt, kann dieses Verfahren anwenden, um z.B. ein Minimum einer numerischen Funktion zu finden (vgl. Abbildung 10).

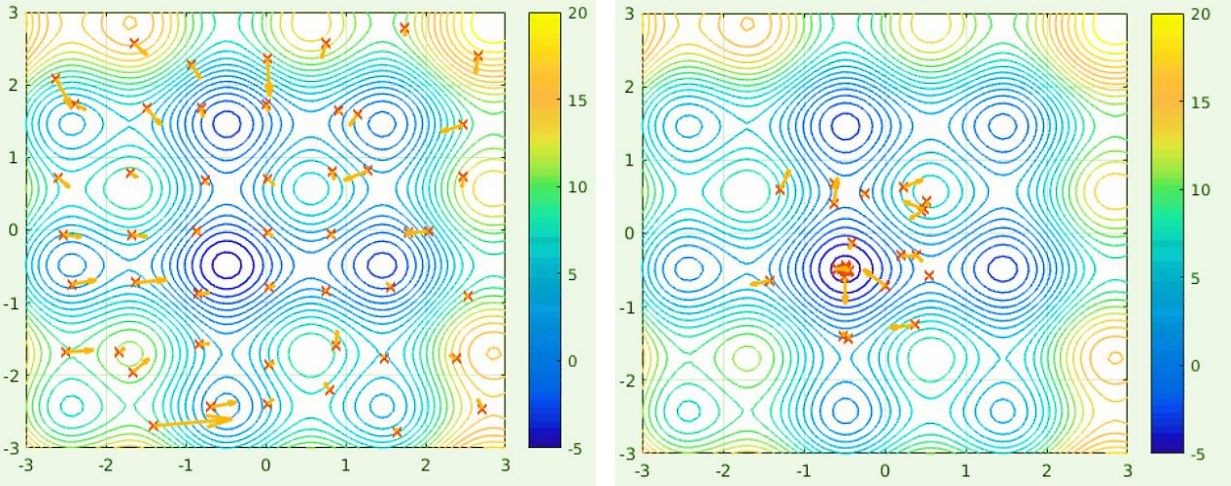

Abbildung 10: Partikelschwarm sucht ein globales Minimum einer Funktion (Vgl. Ephramac (2017))

In Abbildung 10 kann man den Vorteil eines Schwarms bei der Suche nach dem globalen Minimum sehr gut erkennen. Die Schwarmpartikel werden zufällig im Suchraum angeordnet und eine zufällige Geschwindigkeit zugeordnet. Während der Optimierung berechnen die Partikel ständig ihre neue Geschwindigkeit und ihre Position im Suchraum. Die Bewegungsrichtungen werden durch einen Vektor visualisiert. Nachdem die Funktion analysiert worden ist, erfolgt die Bestimmung bzw. Auswertung der globalen und der eigenen besten Position im Raum. Ist die persönliche Position besser als die globale, wird diese zur neuen globalen Position. Nach dieser werden die Partikel ausgerichtet, bzw. neue Geschwindigkeiten zugeordnet und deren Position abermals aktualisiert. [23] Ist dies geschehen, erfolgt eine erneute Auswertung der besten Positionen und Ausrichtung der gesamten Partikel. Dies geschieht solange, bis die Abbruchbedingung erfüllt ist, z.B. wenn sich die globale Position eine gewisse Zeit lang nicht mehr ändert und

[23] Vgl. Bäcker, B., et al. (2016), S.13

somit das Minimum der Funktion gefunden wurde. Zu erkennen ist, dass sich die meisten Partikel nach einer gewissen Zeit an dem globalen Minimum häufen. Darüber hinaus kann man das Optimierungsverfahren auch auf das Problem des Handlungsreisenden[24] oder zur Objektlageerkennung zur automatisierten Entnahme von bestimmten Objekten[25] anwenden.

2.2.3 Artificial Bee Colony Algorithm (ABC)

Der Artificial Bee Colony Algorithm, zu Deutsch: künstlicher Bienenkolonie Algorithmus, ist genau wie die Partikelschwarmoptimierung ein Optimierungsverfahren. Dieser orientiert sich an der effizienten Nahrungssuche der Bienen. Auch hier existiert ein Suchraum, in dem sich die Bienen frei bewegen. Jedes mögliche Ergebnis stellt eine Futterquelle dar. Grundlage bildet hier die Bienenkolonie. Die eine Hälfte der Kolonie besteht aus Arbeiterinnen und Scouts, während die andere Hälfte aus beobachtenden Bienen besteht, welche sich den Tanz der Scouts anschauen und sich für die wahrscheinlich beste Futterquelle entscheiden. [26] Eine Arbeiterin ist am Anfang einer festen Futterquelle zugeordnet. Ziel ist es, in jeder Iteration eine benachbarte Futterquelle, sprich eine benachbarte Lösung für eine Arbeiterin zu identifizieren. Jeder beobachtenden Biene wird zufällig einer Arbeiterin zugeordnet. Für jede neu zugeordnete beobachtende Biene wird eine neue benachbarte Futterquelle für die Arbeiterin definiert. Wurde eine benachbarte Futterquelle identifiziert, wird verglichen, ob diese qualitativ besser ist (vgl. Abbildung 11). Ist dies der Fall, wird die benachbarte Futterquelle als neue Lösung für diese Arbeiterin gesetzt. Wird die Abbruchbedingung erreicht und die Arbeiterin hat

[24] Vgl. Bäcker, B., et al. (2016).

[25] Vgl. Ledermann, Thomas (2012).

[26] Vgl. Morillo, Oscar (2014).

keine benachbarte und zugleich qualitativ bessere Futterquelle identifiziert, wird diese zu einem Scout und sucht nach einer neuen zufälligen Futterquelle.[27]

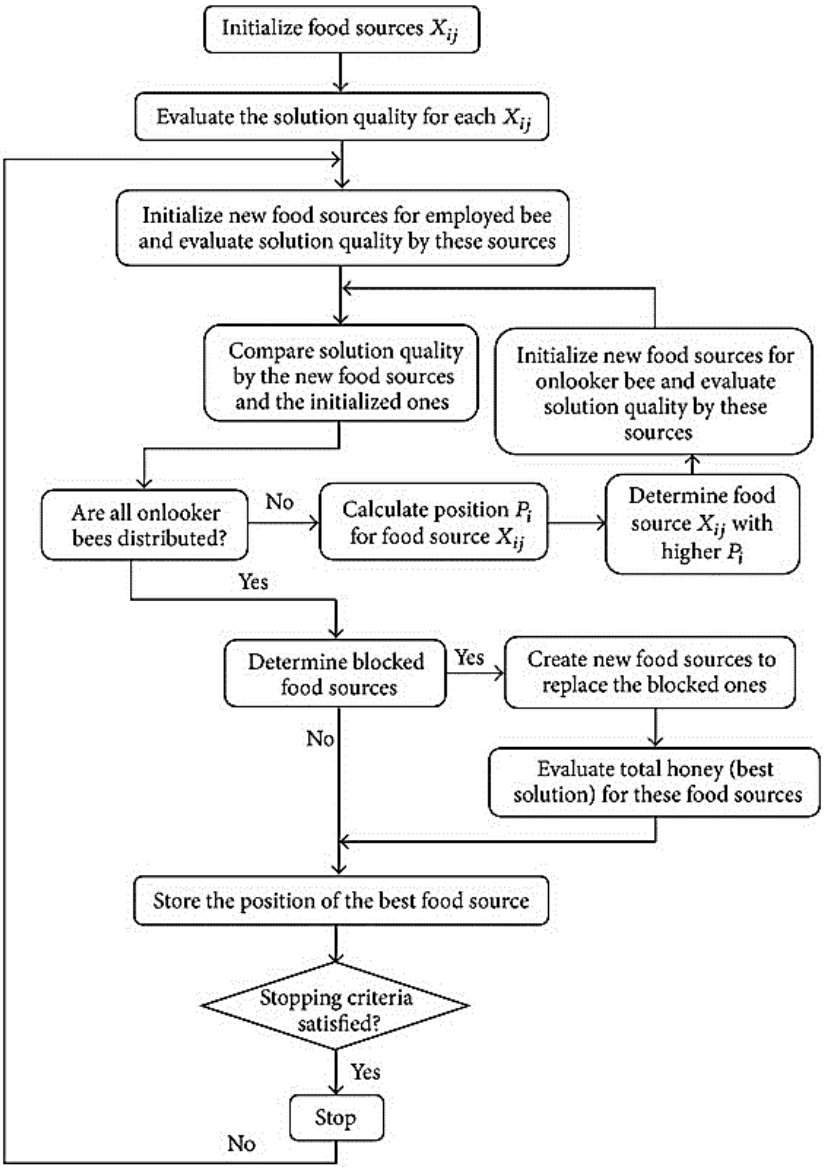

Abbildung 11: Flussdiagramm des ABC Algorithmus (Vgl. Loung Dinh Le (2013))

[27] Vgl. Bosse, Sascha (2016).

Natürlich gibt es noch wesentlich mehr Algorithmen und Ansätze, die von Schwärmen aus der Tierwelt inspiriert wurden, auf die im nachfolgenden nicht genauer eingegangen wird. Einige Beispiele für weitere Algorithmen sind:

- Firefly Algorithm
- Monkey Search Algorithm
- Roach Infestation Optimization
- Jumping Frogs Optimization
- Bacterial Foraging Algorithm
- Artificial Immune System Algorithm
- Glowworm Swarm Optimization

2.2.4 Swarm-Bots

Mithilfe des Schwarmverhaltens wurden neue Ideen für die heutige Robotik entwickelt. Kernelement für Roboterschwärme sind die einzelnen Schwarmroboter, auch S-Bots oder Agent genannt, die so einfach wie möglich aufgebaut und so robust wie möglich konstruiert werden. Durch die einfache Konstruktion soll eine gewisse Robustheit und Ausfallsicherheit erreicht werden.

Mit dem MARS Projekt wird versucht, Roboterschwärme für die Landwirtschaft zu nutzen. MARS steht für *Mobile Agricultural Robot Swarms* und besteht aus 6 bis 12 Robotern, die zusammen eine Flächenleistung von 10.000 Quadratmeter pro Stunde besitzen. Laut Fendt werden die Roboter mithilfe von Satellitennavigation gesteuert und stellen ihre Informationen über eine Cloud zur Verfügung. Zusätzlich existiert eine Logistikeinheit (vgl. Abbildung 12), die den Roboterschwarm transportiert, diesen mit Strom versorgt (die einzelnen Roboter müssen zum Aufladen in die Logistik Einheit fahren), das Saatgut enthält und die Kommunikationszentrale des Schwarms darstellt. Hierdurch ist es möglich, egal von welchem Ort, die Saat zu konfigurieren, den aktuellen Status des Schwarms anzeigen zu lassen sowie Updates „Over-the-Air“ durchzuführen. Robo-

terschwärme in der Landwirtschaft einzusetzen, bringt enorm viele Vorteile mit sich. Sollte ein Schwarmroboter ausfallen, werden sämtliche Pfade neu berechnet und die verbleibenden Roboter übernehmen die Arbeit des Roboters. Darüber hinaus können die Roboter mit verschiedenem Saatgut bestückt werden, wodurch man strukturiert verschiedene Pflanzensorten auf einem Feld anbauen kann. Durch die hohe Flexibilität des Schwarms und der autonomen Arbeitsweise des ganzen Systems, ist es möglich, 24 Stunden, 7 Tage in der Woche das Land zu bewirtschaften. Ein weiterer Vorteil der Roboter ist, dass diese elektrisch angetrieben werden und somit wesentlich umweltschonender sind, als große Traktoren. Aufgrund des geringen Gewichtes der Roboter (ca. 50kg), ist dementsprechend auch die Bodenbelastung wesentlich geringer. [28] Ein wesentlicher Nachteil des Systems sind die wahrscheinlich hohen Anschaffungskosten, zu denen bisher noch keine Angaben vom Hersteller gemacht wurden.

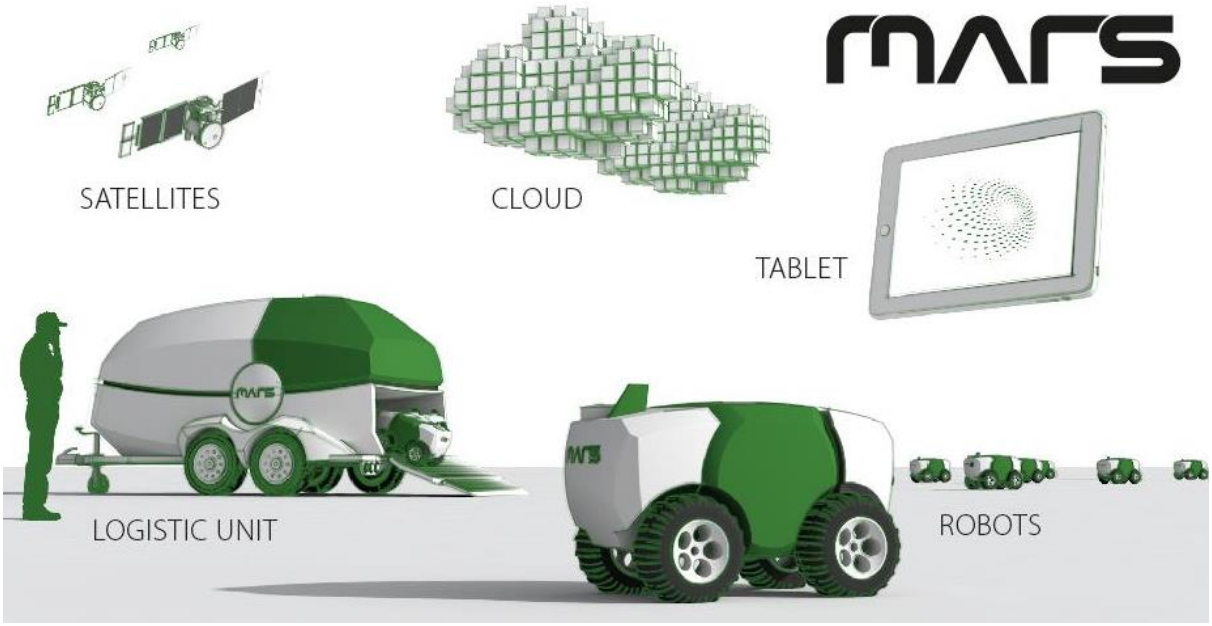

Abbildung 12: Das MARS System (Vgl. Ola Tande (2016))

[28] Vgl. Fendt (2018).

Darüber hinaus gibt es noch wesentlich mehr Projekte, in denen S-Bots Anwendung finden. Beispiele Hierfür sind unter anderem:

- Bauen von Häusern oder beliebigen Strukturen mithilfe von S-Bots, inspiriert durch Termiten[29]
- Drohnenschwärme aus sogenannten Logistikdrohnen zum Ausliefern von Lieferungen jeglicher Art, wodurch diese schneller zugestellt werden können[30]
- Drohnenschwärme auf dem Mars zum kartographieren der Marsoberfläche[31]
- Drohnenschwärme um Feuer zu löschen, da diese schneller vor Ort sind und auch an schlecht zugänglichen Stellen eigesetzt werden können[32]
- Militärische Anwendungen, wie bewaffnete Drohnenschwärme, um den Bodentruppen in gefährlichen Situationen Feuerschutz zu gewährleisten[33]
- Drohnenschwärme zum Aufbauen von Kommunikationsnetzwerken in Krisengebieten, um die Kommunikation und Organisation der Helfer zu gewährleisten[34]

Zu sehen ist hier eine rasante Zunahme in der Nutzung von Roboterschwärmen in der Technik, um Kosten, Gefahren und Aufwand zu minimieren. Klar wird auch, dass der Schwarm einen deutlichen Mehrwert sowie eine wesentlich gesteigerte Ausfallsicherheit mit sich bringt, im Gegensatz zu einem einzelnen Roboter.

---

[29] Vgl. Werfel, J. (2016).

[30] Vgl. Deutsche Post AG (2016).

[31] Vgl. Bayrischer Rundfunk (2017).

[32] Vgl. ITG Fotoflug (2017).

[33] Vgl. Barrie, A (2018).

[34] Vgl. FOCUS Online (2010).

3. Schwarmintelligenz

Die Begriffe Schwarmverhalten und Schwarmintelligenz werden in der Literatur häufig als Synonym benutzt, da es sehr umstritten ist, ab wann genau man von Schwarmintelligenz spricht. Die Begriffe Schwarmintelligenz und Schwarmverhalten kann man jedoch sehr gut voneinander unterschieden, da die Definitionen von Verhalten und Intelligenz ebenfalls eindeutig unterschiedlich sind.

Laut dem Duden bedeutet das Wort Verhalten, dass die Individuen des Schwarms „in bestimmter Weise auf jemanden, etwas in einer Situation o. Ä. reagieren“[35]. Dies bestätigt, dass, wie in Kapitel *2. Schwarmverhalten* schon aufgeführt, das Schwarmverhalten das simple Verhalten der Individuen innerhalb eines Schwarms und deren Reaktion auf die Reize ihrer Umwelt bedeutet.

Die Schwarmintelligenz, oder auch kollektive Intelligenz genannt, entsteht erst, indem einzelne Entscheidungen bzw. Reaktionen eines Individuums das scheinbar intelligente Verhalten des ganzen Schwarms beeinflussen. Ein Phänomen, welches lange Zeit unerklärlich schien. Man spricht hier nicht etwa vom im Duden definierten Intelligenzbegriff, nämlich die „Fähigkeit […], abstrakt und vernünftig zu denken und daraus zweckvolles Handeln abzuleiten“[36], sondern vielmehr von einer sehr hohen Form der Selbstorganisation und der Kommunikation. Am Beispiel eines, in Kapitel *2.1.1 Ameisen* genannten, Ameisenstaats lässt sich der Begriff Schwarmintelligenz sehr deutlich veranschaulichen. Betrachtet man eine einzelne Ameise, wird klar, dass das Individuum über keine höhere Intelligenz verfügen muss. Durch die hohe Selbstorganisation des Ameisenstaats und die Reaktion auf Umwelteinflüsse sowie die Kommunikation der Individuen untereinander, entsteht erst die vermeintliche und augenscheinliche Schwarmintelligenz, da das Verhalten des Schwarms sinnvoll und intelligent erscheint. Der Begriff In-

[35] Bibliographisches Institut GmbH (2018a).

[36] Bibliographisches Institut GmbH (2018b).

telligenz fällt hierbei im Zusammenhang mit einem Gruppengedächtnis, welches dafür sorgt, dass verschiedene Probleme nicht individuell, sondern im Schwarmverbund gelöst werden.[37] Schwarmintelligenz setzt also ein vernünftiges und eigenverantwortliches Denken und Handeln eines Individuums voraus. Dies ist bei einem Tierschwarm nicht gegeben, außer dieser hätte eine zentrale Einheit, welche Informationen auswerten und den Schwarm dementsprechend koordinieren und somit steuern würde.

Ein passendes Beispiel für den Begriff Schwarmintelligenz lieferte Francis Galton schon im Jahr 1907. Auf einer westenglischen Nutztiermesse ließ er 787 Personen das Gewicht eines Ochsen schätzen. Hierzu mussten die Befragten, darunter auch einige Experten wie Metzger und Farmer, unabhängig voneinander das Gewicht schätzen. Der Mensch an sich ist in der Lage, eigenverantwortlich und logisch zu denken, danach zu handeln und dadurch Probleme zu lösen. Hierdurch ist für dieses Beispiel die Definition der Intelligenz zutreffend. Das Gewicht des Ochsen lag bei 1.207 Pfund (ca. 547kg). Der Median aller Schätzungen der Teilnehmer lag erstaunlicherweise bei 1.198 Pfund (ca. 543kg). Dies bedeutet eine Abweichung von lediglich 0,8 Prozent. Hiermit war der Median exakter als jede Einzelschätzung.[38] Francis Galton bewies somit, dass eine Gruppe von Menschen unter bestimmten Bedingungen, wie z.B. ein solcher Schätzwettbewerb und ohne Kommunikation untereinander, intelligenter sein kann, als die darin befindliche intelligenteste Person. Selbiges Experiment wurde im Jahr 2008 von Mathematiker der Bremer Universität durchgeführt. Allerdings mussten die 150.000 Befragten hier die Anzahl verkaufter Lose einer Tombola schätzen. Es wurden insgesamt 10.788 Lose verkauft. Der Median der Schätzungen lag hier bei 9.834 Losen. Auch hier war der Median genauer als jede Expertenschätzung, was den Versuch von Francis Galton bestärkt.[39] Verblüffend ist ein Artikel der wissenschaftlichen Fachzeit-

[37] Vgl. Pitscher, L. (2008) S. 3 ff.

[38] Vgl. Galton, F. (1907) S.450 f.

[39] Vgl. Riccó, J. (2008).

schrift *Proceedings of the National Academy of Sciences* aus dem Jahr 2011, in dem über die Durchführung eines Versuchs von Dirk Helbing berichtet wird. Dieser Versuch war ein ähnlicher Versuch, wie der von Francis Galton. Hier wurden 144 Studenten befragt. Diese mussten ebenfalls Werte Schätzen, wie z.B. die Anzahl der Morde in der Schweiz im Jahr 2006. Bei diesem Versuch kam auch selbiges Ergebnis wie bei den Versuchen von Galton und den Mathematikern der Bremer Universität zum Vorschein. Wurden die Schätzungen unabhängig voneinander abgegeben, war der Median sehr nah an dem richtigen Wert. Erfuhren die Befragten die Schätzwerte vorheriger Studienteilnehmer, verschwanden zwar die einzelnen Extremwerte, der Median aber entfernte sich deutlich vom richtigen Ergebnis.[40] Für detaillierte Grafiken zu diesem Versuch vgl. Anhang A1.

Durch diesen Versuch lässt sich sehr gut die Wichtigkeit der Rahmenbedingungen eines solchen Versuchs sowie den sozialen Einfluss der Individuen im Schwarm feststellen. Während in den bisher, von Galton und den Mathematikern der Bremer Universität, durchgeführten Versuchen unabhängig voneinander geschätzt wurde, durften im zweiten Teil des Versuchs von Helbing die Teilnehmer vorherige Schätzwerte für die eigene Schätzung nutzen. Zu sehen ist ein sozialer Einfluss der Teilnehmer, da diese sich eher an den Ergebnissen anderer orientierten, als an der eigenen Vermutung. In diesem Zusammenhang wird oft der Begriff „Schwarmdummheit“ genannt. Dieser wird von vielen als Gegenkonzept für die Schwarmintelligenz benutzt. Wie der Versuch von Helbing zeigt, herrscht ein schmaler Grat zwischen Schwarmintelligenz und Schwarmdummheit. Dieser schmale Grat stellt den sozialen Einfluss, den die einzelnen Individuen aufeinander haben, dar. Vergleicht man den sozialen Einfluss von Menschen und Tieren im Schwarm, kann man sehr leicht zu dem Ergebnis kommen, dass die Tiere auch bei gewisser Unsicherheit ihren Instinkten weiterhin folgen bzw. ihr Verhaltensmuster nicht verlassen, während die Mehrheit der Menschen sich hier sozial stärker beeinflussen

[40] Vgl. Lorenz, J. / Rauhut, H. (2011).

lässt und nicht mehr nach dem eigenen Interesse handelt. Und zwar von den Individuen die ihre Meinung durchsetzen und von dieser überzeugt sind oder gut manipulieren können. Der Mensch kann zwar seinen eigenen Verstand benutzen, neigt aber in einer großen Gruppe oder bei Unsicherheit dazu, sich an anderen zu orientieren. Hier steckt das Schwarmverhalten im Menschen und gleichzeitig auch die Schwarmdummheit.

Schaltet man die soziale Beeinflussung durch Rahmenbedingungen, wie z.B. unabhängige Schätzungen oder zu frühe Kommunikation aus, kann der Mensch die Schwarmintelligenz effizient für sich nutzen, ohne dadurch negativ beeinflusst zu werden. Da beim Mensch der soziale Einfluss am größten ist, besteht besonders für ihn die Gefahr der Schwarmdummheit, da er sich schnell an der Mehrheit orientiert.

Zusammenfassend lässt sich also sagen, dass man die Begriffe Intelligenz und Verhalten an sich klar abgrenzen kann, diese aber in Bezug auf einen Tierschwarm ineinander überfließen bzw. aufeinander aufbauen. Denn ohne ein instinktives Verhalten der einzelnen Schwarmtiere, können diese nicht auf die Reize ihrer Umwelt reagieren und dadurch auch keine Probleme im Verbund lösen, was den Schwarm für uns Menschen erst intelligent wirken lässt. Der Mensch ist in der Lage Schwarmintelligenz effizient nutzen zu können, er ist dadurch aber auch anfälliger, Schwarmdumm zu werden, da er durch soziale Beeinflussung in sein Schwarmverhalten bzw. Herdenverhalten fällt und sich an anderen orientiert anstatt an seinen Ideen und Interessen festzuhalten.

4. Lego Mindstorms

Lego Mindstorms ist eine Produktreihe des Herstellers Lego. Sie stellt einen programmierbaren Microcontroller zur Verfügung. Dieser bildet das Kernstück des Roboters dar. An diesen werden die beiden Elektromotoren und verschiedene Sensoren angeschlossen (vgl. Abbildung 13). Deswegen lässt sich diese Produktreihe sehr gut für den Bereich der Robotik verwenden, da hiermit einfache Roboter und Systeme aufgebaut und programmiert werden können. In dieser Studienarbeit werden Roboter der Generation

Lego Mindstorms NXT, die Vorgängergeneration der aktuellen EV3-Generation, verwendet. Die erstellten Programme werden entweder über Bluetooth oder über die USB 2- Schnittstelle in den Speicher des Microcontrollers geladen. Ist dieser mit dem PC verbunden, kann man den Ablauf vom PC aus starten oder, falls er nicht mit einem PC verbunden ist, auch über die Kontrolltasten auf dem Microcontroller selbst.

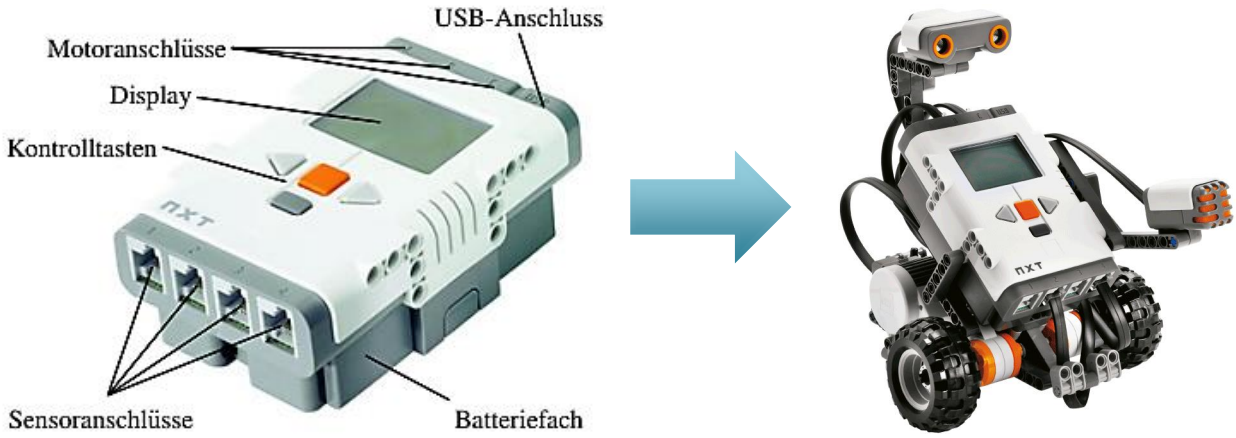

Abbildung 13: Lego NXT (Vgl. Schmidt, D./Berns, K. (2010), S.42)

Im nachfolgenden wird auf die Sensorik und Aktorik des Lego Mindstorms NXT eingegangen und erklärt.

Die Aktorik ist eines der wichtigsten Bauelemente, da ohne die Elektromotoren das Individuum bewegungsunfähig wäre und man somit kein Schwarmverhalten nachbilden könnte. Der NXT verfügt über zwei Servomotoren, die auf der oberen Seite des Microcontrollers an diesen angeschlossen werden. Servomotoren sind Elektromotoren, bei denen man die Geschwindigkeit und die Winkel kontrollieren kann (vgl. Abbildung 14). Sobald diese verbunden sind, kann man sofort mit dem Programm auf diese zugreifen.

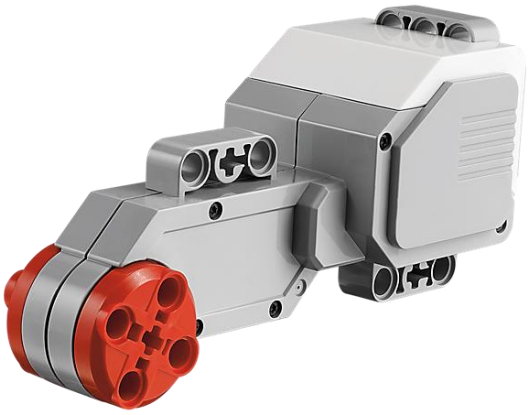

Abbildung 14: Servomotor des NXT (Vgl. The LEGO Group (2016a))

Die Sensorik besteht aus mehreren Komponenten, wie einen Tastsensor, einem Ultraschallsensor, einem Farbsensor und einem Geräuschsensor.

Der Tastsensor ist ein einfacher Sensor, der auf Berührung reagiert, wenn z.B. der Roboter gegen ein Hindernis fährt. Aus Abbildung 15 kann man entnehmen, dass dieser nach dem Federprinzip funktioniert. Sobald die Feder durch das fahren an das Hindernis zusammengedrückt wird, wird der Stromkreis unterbrochen. Dieser Stromkreis wird vom Microcontroller überwacht und ermöglicht es, im Programm darauf zu reagieren. Für diese Studienarbeit ist dieser Sensor eher nicht geeignet, da Schwarmtiere sich nicht mithilfe von Berührung, sondern mithilfe ihrer Sicht orientieren.

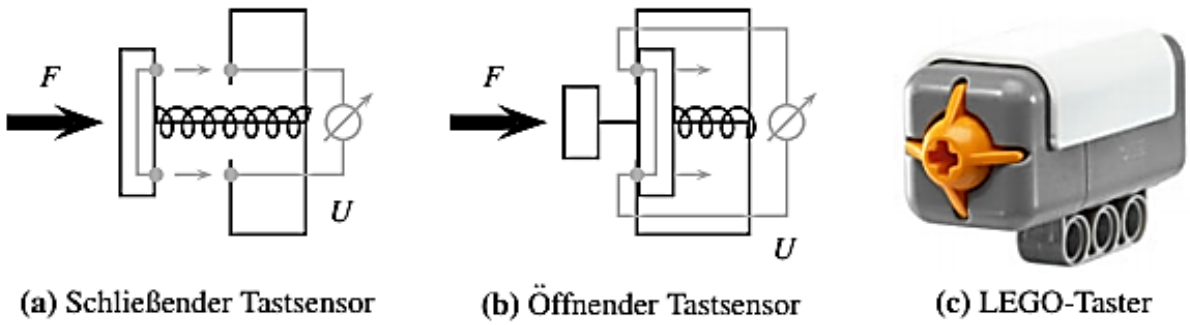

Abbildung 15: Tastsensor des NXT (Vgl. Schmidt, D. /Berns, K. (2010) S.50)

Hier kommt der Ultraschallsensor zum Tragen (vgl. Abbildung 16 (b)). Mit diesem können Entfernungen mithilfe von Ultraschall berechnet werden. Hier werden Töne im hochfrequenten Bereich ausgesendet und die Zeit gemessen, wie lange es braucht, bis das Echo am Sensor antrifft (vgl. Abbildung 16 (a)). Dieser Sensor stellt in gewissen Maßen das Blickfeld eines Individuums dar. Im Programm kann man z.B. den Abstand bestimmen, den das Individuum zu Hindernissen halten soll. Ist dieser Abstand erreicht kann man entweder den Roboter stoppen oder darauf reagieren und eine andere Richtung einschlagen.

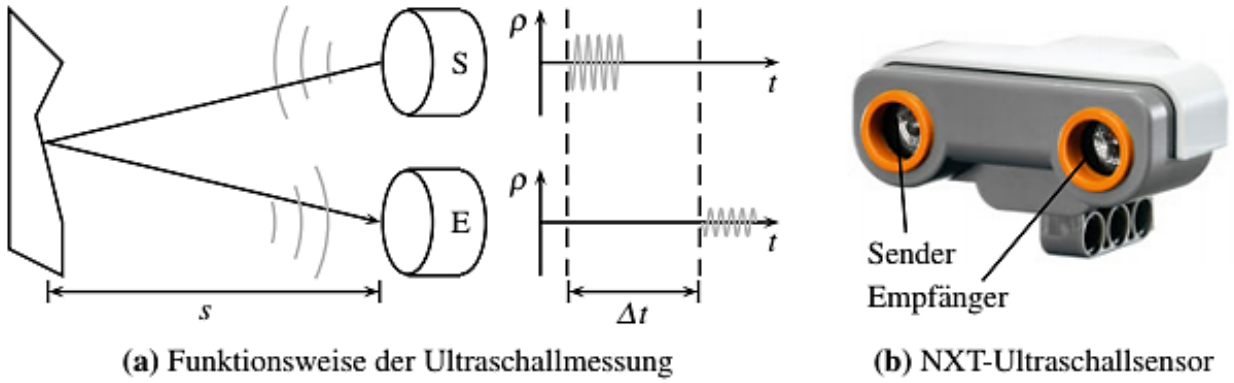

Abbildung 16: Ultraschallsensor des NXT (Vgl. Schmidt, D. /Berns, K. (2010) S.51)

Der Farbsensor kann für verschiedene Versuche sehr hilfreich sein. Zum einen kann man den Räuber, der sich auf einen Schwarm zubewegt, mit einer anderen Farbe kennzeichnen, als die Schwarmtiere. Hierdurch kann der Roboter dann zwischen Schwarmmitglied und Fressfeind unterscheiden und entsprechend das Schwarmverhalten fortsetzen oder ein Ausweichprogramm durchführen. Zum anderen kann hier Futter gekennzeichnet werden, wonach der Schwarm aktiv auf der Suche ist. Der unter Abbildung 17 erkennbaren Farbsensor bestrahlt hierfür die entsprechende Oberfläche mit 3 LEDs, die alle drei eine unterschiedliche Farbe haben. Das zurückkommende Licht wird mithilfe einer Fotodiode gemessen und ausgewertet.[41]

[41] Vgl. Schmidt, D. / Berns, K. (2010).

Abbildung 17: Farbsensor des NXT (The LEGO Group (2016b))

Der letzte Sensor des NXT ist der Geräuschsensor (vgl. Abbildung 18). Ein Fallbeispiel für die Verwendung des Sensors könnte der oben genannte Schwarm sein, der Futter sucht und dieses mithilfe des Farbsensors erkennt. Damit der restliche Schwarm folgt, kann der Roboter, der das Futter gefunden hat, ein bestimmtes Geräusch machen, worauf die anderen Roboter reagieren und ebenfalls zur Futterquelle stoßen. Der Geräuschsensor empfängt die Schallwellen der Umgebungsgeräusche und wandelt diese je nachdem wie laut der Ton ist und wie stark die Schwallwellen auftreffen, in starke oder schwache Signal um.

Abbildung 18: Geräuschsensor des NXT (The LEGO Group (2016c))

Programmiert werden kann der NXT über zwei verschiedene Arten. Die erste Variante ist die mitgelieferte NXT-G Entwicklungsumgebung von Lego Mindstorms. Diese Entwicklungsumgebung ist eine grafische Entwicklungsumgebung, sprich, es werden sogenannte Blöcke per Drag-and-Drop an den gekennzeichneten Startpunkt gezogen (vgl.

Abbildung 19). Es lassen sich hier sehr einfach parallele und sequentielle Programme erstellen. Die Blöcke besitzen unterschiedliche Funktionen. Während einer die Motoren steuern kann, ist wiederum ein anderer für den Farbsensor zuständig. Klickt man die Blöcke an, kann man diese mit individuellen Werten konfigurieren, wie z.B. die Dauer, die der Motor drehen soll.

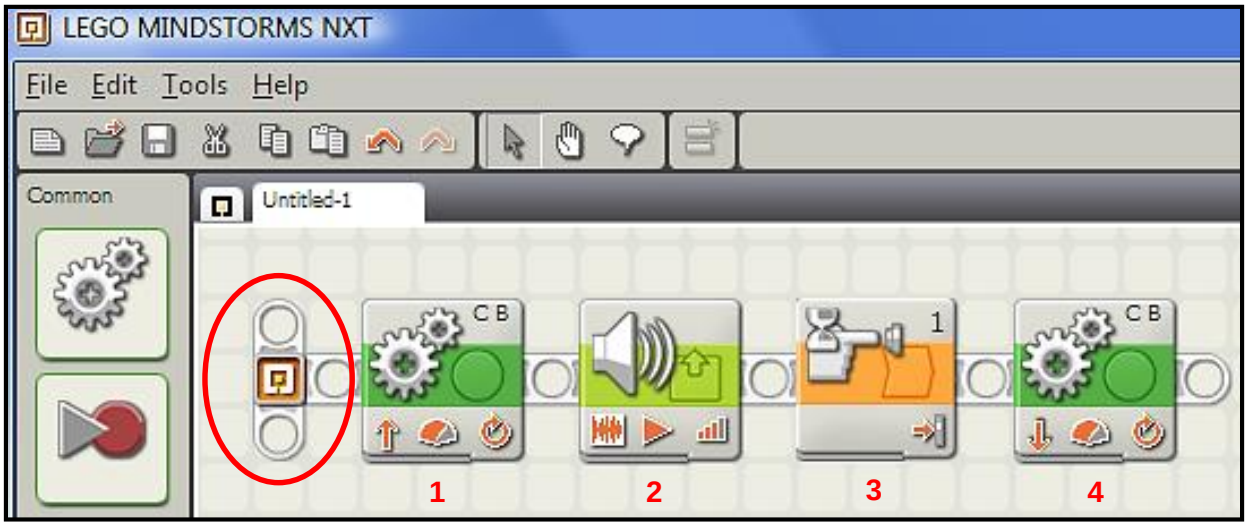

Abbildung 19: NXT- G: grafische Entwicklungsumgebung

Der rote Kreis in Abbildung 19 stellt den Startpunkt der Programmabarbeitung dar. Baustein Nr.1 lässt die Motoren eine eingestellte Dauer laufen. C und B stehen jeweils für den Port, an denen die Motoren mit dem Microcontroller verbunden sind. Block Nr.2 ermöglicht es, einen Ton oder eine individuell hochgeladene Sounddatei abspielen zu lassen. Block Nr.3 hält das Programm an und prüft, ob der Drucksensor gedrückt wird. Erst wenn dieser aktiviert wird, läuft das Programm weiter und der Roboter fährt anhand von Block Nr.4 weiter.

Die zweite Variante, den NXT zu programmieren, erfolgt mithilfe der Programmiersprache Java. Java bietet sich hier sehr gut an, da diese Programmiersprache objektorientiert ist. Es lassen sich also Objekte erzeugen. Doch nicht nur die Objektorientiertheit von Java ist hier ein Vorteil. Ein weiterer Vorteil bietet Java, da man hier viel mehr Möglichkeiten hat, als mit NXT-G. Zum einen kann man die Vererbung sehr leicht auf Objek-

te anwenden, zum anderen bietet Java eine Vielzahl an mathematischen Funktionen und Datentypen, die bei komplexeren Projekten, wie diesem hier, gefordert sind. Aus diesem Grund fiel die Wahl auf eine textuelle Programmiersprache. Doch um Java auf einem NXT nutzen zu können benötigt man LeJOS.

Beim Aufsetzen der Programmierumgebung, in diesem Fall Java Neon mit LeJOS, wird zuerst der Microcontroller, mit einer von LeJOS bereitgestellten Firmware, neu aufgesetzt. Der Controller wird also mit einer neuen Firmware ausgestattet die eine JVM (Java Virtual Machine) enthält. Mithilfe dieser werden Java Programme in einer eigenen virtuellen Maschine ausgeführt, wodurch man Java Programme auf den NXT laden und ausführen kann. Mit der Firmware kommen auch neue Java Klassen, die dazu gehörenden Linker und eine API (Application Programming Interface), damit der Benutzer Befehle für den NXT abrufen, seine Programme samt Klassen auf den NXT laden und dort benutzen kann. Nun muss nur noch die Programmierumgebung konfiguriert werden. Es müssen hier die Pfade zur API und der SDK (Software Development Kit) angegeben werden, damit die Umgebung auf den NXT zugreifen kann.[42] Eine genaue Anleitung, wie das Aufsetzen des NXT und der dazugehörige Programmierumgebung funktioniert, können Sie aus dem Anhang *A2.Aufsetzen von LeJOS* entnehmen.

5. Agentensysteme

Michael Wooldridge ist der Auffassung, dass es keine einheitliche Definition für den Begriff „Agent“ gibt. Ebenso hebt er hervor, dass sich jedoch viele einig sind, dass ein Agent zu einer gewissen Selbstständigkeit fähig ist, bzw. sein muss. Sprich der Agent soll unabhängig von sämtlichen Eingriffen arbeiten. Diese Eigenschaft ist in sehr vielen Definitionsversuchen zu finden, wo andere Eigenschaften jedoch sehr unstimmig verwendet werden. Aus diesem Grund versuchte Wooldridge eine sehr vereinfachte und

[42] Vgl. o.V. (o.J.a)

übergeordnete Definition zu formulieren. So ist laut ihm ein Agent ein Computersystem, das sich in einer bestimmten Umgebung befindet und dort durch sein autonomes Handeln, seine gestalteten Ziele zu erreichen versucht.[43] (Vgl. Abbildung 20)

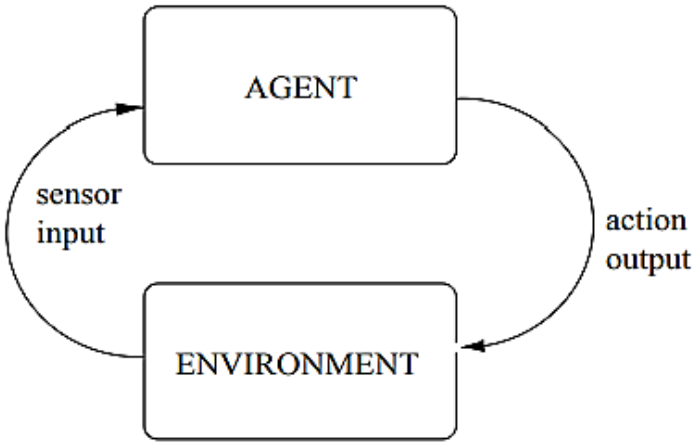

Abbildung 20: Interaktion eines Agenten mit seiner Umgebung (Vgl. Wooldridge, M. (2002), S.6)

Wie aus Abbildung 20 hervor geht, kann ein Agent seine Umwelt nur mit Sensoren wahrnehmen und je nachdem, wie sich diese verändert, darauf reagieren. Diese Reaktion erfolgt mithilfe von einer bestimmten Aktion des Agenten. Wooldridge nennt neben der Selbstständigkeit, auch weitere, für ihn sehr gut passende, Eigenschaften eines Agenten: [44]

- **Soziale Fähigkeit**: Ein Agent interagiert mit anderen Agenten über eine bestimmte Sprache
- **Reaktivität:** Ein Agent nimmt seine Umgebung stets wahr und reagiert auf Verlandungen in dieser
- **Proaktivität:** Ein Agent führt Aktionen aufgrund von Eigeninitiative aus, sprich er weist ein zielgerichtetes Verhalten auf

[43] Vgl. Wooldridge, M. (2002).

[44] Vgl. Wooldridge, M. / Jennings, R. N. (o.J.).

Da es aber je nach Umwelt bzw. je nach Verhalten des Agenten auf seine Umwelt Unterschiede gibt, wurden diese nochmals untergliedert. Eine grobe Gliederung liefert hier Gordon Bernedo Schneider (vgl. Abbildung 21).

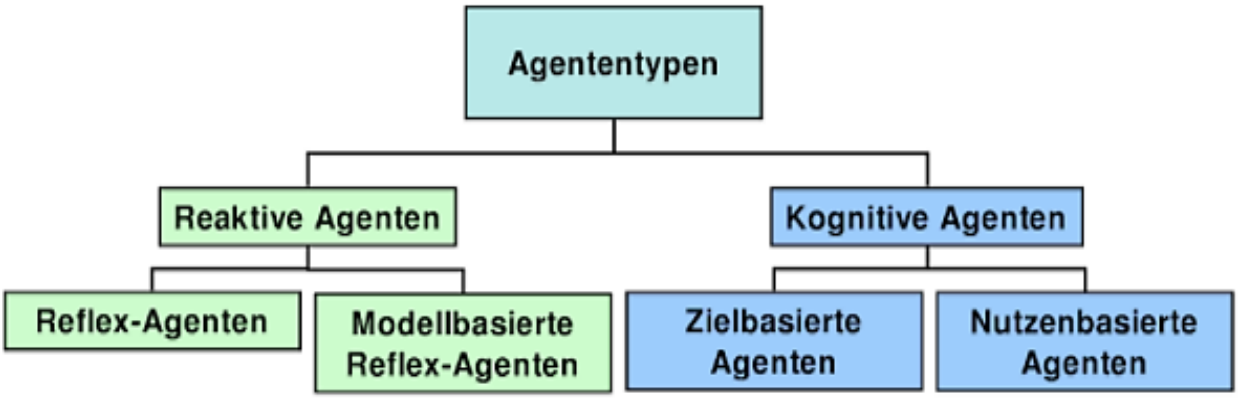

Abbildung 21: Agententypen (Vgl. Schneider, G. B. (2009), S. 79)

Wie in Abbildung 22 zu erkennen ist, werden zwei Agententypen unterschieden. Reaktive und kognitive Agenten. Wie das Wort schon sagt, agieren reaktive Agenten direkt auf ihre Umwelt und deren Veränderung. Ein festes Regelwerk schreibt den Agenten hier vor, mit welcher Aktion sie auf die entstehenden Zustände der Umwelt reagieren sollen, um irgendwann ein bestimmtes Ziel zu erreichen. Einfache Reflex-Agenten handeln reflexartig. Als Beispiel hierfür wäre ein Autofahrer der ebenfalls bremst, sobald das vor ihm befindliche Fahrzeug bremst (vgl. Abbildung 22).

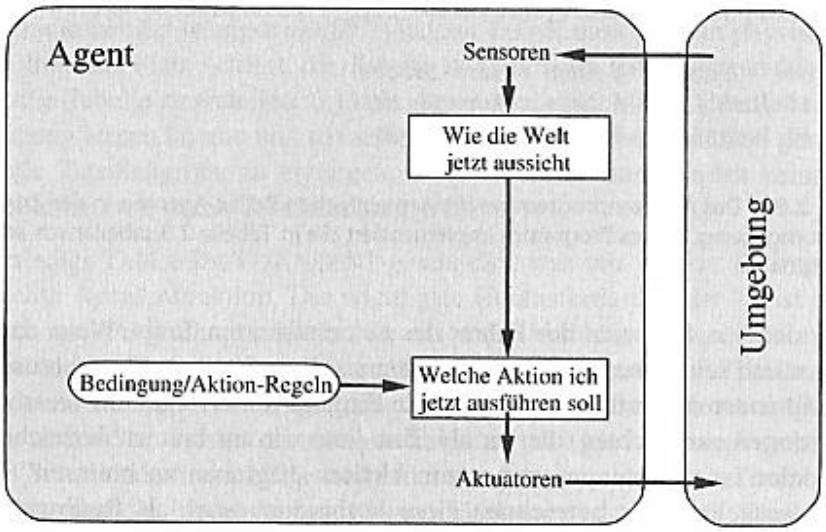

Abbildung 22: Einfacher Reflex-Agent (Vgl. Anders, C. (2010), S.8)

Modellbasierte Reflex-Agenten (vgl. Abbildung 23) sammeln im Gegensatz zu einfachen Reflex-Agenten Informationen ihrer Umwelt, speichern also ihre Sensordaten ab und können so die Umweltveränderung im Modell abbilden. So erstellen Reinigungsroboter ein Modell ihrer Umwelt, um ein effizientes Reinigen der Umgebung zu ermöglichen. Die gesammelten Informationen von deren Sensoren und Aktuatoren werden benutzt, um das Modell zu fertigen. Die Abarbeitung der Reinigung erfolgt dann immer in der Relation zum internen Modell. Auf die tatsächliche Umwelt wird dann mithilfe von Sensoren reagiert, falls sich diese schlagartig ändert und somit nicht mehr dem Modell entspricht.[45]

Als Agenten werden in dieser Studienarbeit die einzelnen mobilen Roboter bezeichnet.

[45] Vgl. Kramann, G. (2014).

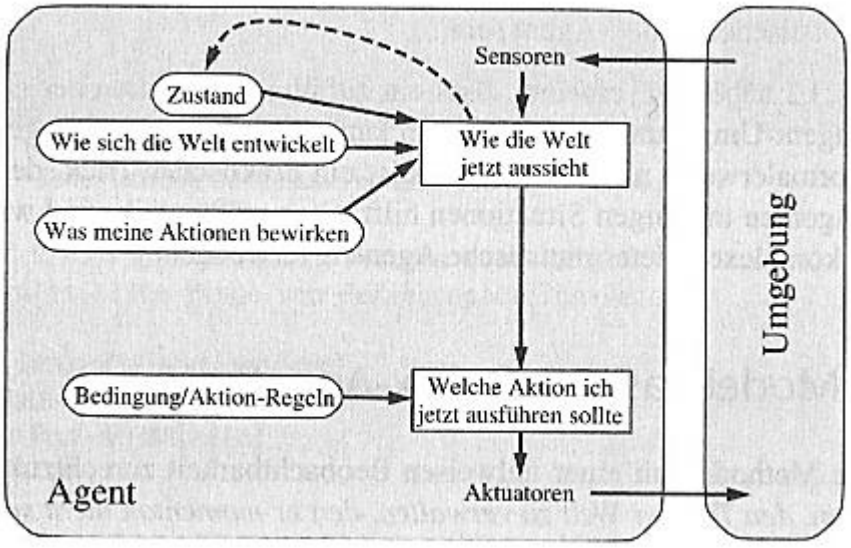

Abbildung 23: Modellbasierter Reflex-Agent (Vgl. Anders, C. (2010), S.9)

Kognitive Agenten reagieren nicht nur wie die reaktiven Agenten direkt auf ihre Umwelt, sondern sind auch von vornherein auf das Erreichen von Zielen ausgerichtet.

Der zielbasierte Agent, hat den Vorteil, dass man bei diesem Agententyp nicht alle möglich vorkommende Handlungsabfolgen definieren muss. Er entscheidet selbst, welche weiteren Schritte möglich wären, bewertet diese, vergleicht sie mit Informationen über das Ziel und entscheidet sich anschließend für den Schritt, der ihn näher an sein Ziel kommen lässt (vgl. Abbildung 24). Hierfür betrachtet er primär die Folgen seiner Aktionen, sprich was seine Aktionen bewirken. Anhand des folgenden Umweltzustandes kann er entscheiden, ob dieser Schritt sinnvoll wäre oder nicht. Hierdurch kann er, im Gegensatz zu einem modelbasierten Agenten, besser auf eine Änderung des Zieles reagieren und dementsprechend agieren. Voraussetzung hierfür nennt Anders den hohen Rechenaufwand, was bei beschränkter Zeit oder Leistung zu einem Nachteil werden kann.[46]

[46] Vgl. Anders, C. (2010).

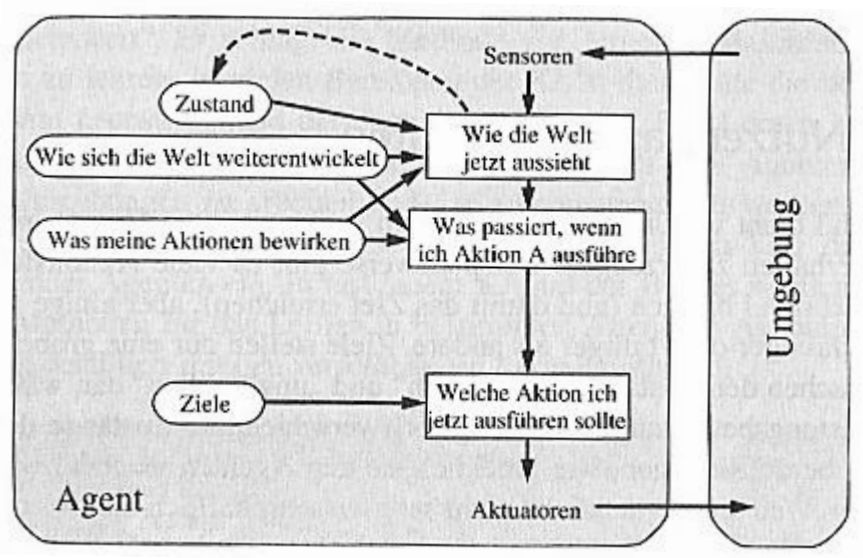

Abbildung 24: Zielbasierter Agent (Vgl. Anders, C. (2010), S.10)

Nutzenbasierte Agenten verfügen über eine genauere Abschätzung der möglichen Schritte. Während ein zielbasierter Agent nur feststellen kann, welcher Schritt ihn näher an sein Ziel bringt, kann ein nutzenbasierter Agent bei gleichwertigen Schritten genauer abwägen.[47] Vergleiche hierfür Abbildung 25. Hierfür besitzt er zusätzlich eine Nutzenfunktion, die die Zustände und deren Folgen auf die Umwelt auf eine reelle Zahl abbildet. Diese reelle Zahl gibt den Nutzen für den Agent an. Dieser Agent kann somit über mehrere Teilziele verfügen, um sein übergeordnetes Ziel zu erreichen. Der Agent zieht hier automatisch Teilziele mit dem größten Nutzen den anderen Teilzielen vor.

[47] Vgl. Kramann, G. (2014).

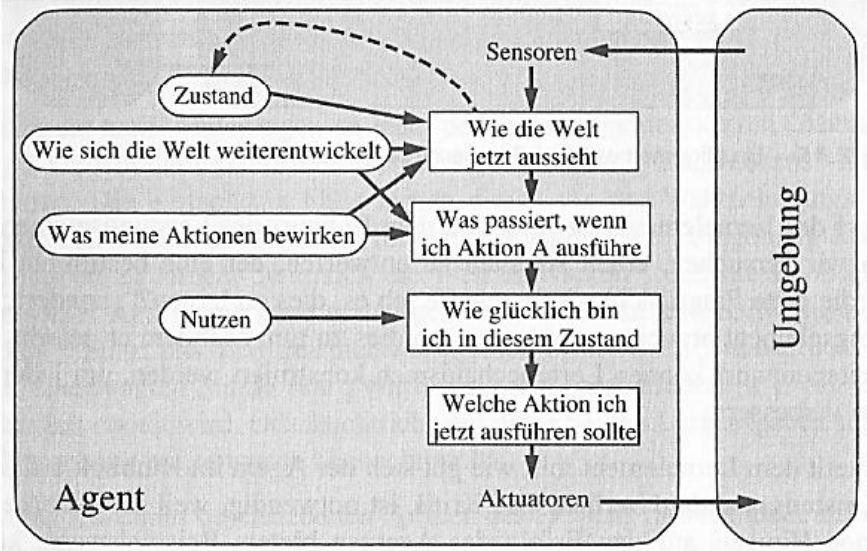

Abbildung 25: Nutzenbasierter Agent (Vgl. Anders, C. (2010), S.11)

Zu sehen ist hier, dass der nutzenbasierte Agent eine kleine Erweiterung, nämlich die Nutzenfunktion, besitzt, welche ihn zu einem neuen Agententyp werden lässt. So stellt auch der modellbasierte Reflex-Agent eine Erweiterung des einfachen Reflex-Agenten dar. Klar wird hierdurch, dass sich die Typen nur gering unterscheiden, wodurch einige Unstimmigkeiten über die richtige Klassifikation der Agententypen entstehen.

5.1 Subsumption Architecture

Die Subsumption Architecture, zu Deutsch Unterordnungsarchitektur, wurde von Rodney Brooks in der Mitte der 80er Jahren entwickelt. Brooks war der Auffassung, dass die ursprüngliche Robotersteuerungsmethode Sense-Plan-Act (vgl. Abbildung 26 (a)) nicht praktikabel ist. Die Sense-Plan-Act Methode ist vergleichbar mit dem einfachen Reflex-Agenten. Dieser sammelt Informationen seiner Umwelt mithilfe von Sensoren (Sense) und erstellt ein eigenes Modell seiner Umwelt und plant seinen nächsten Schritt bzw. eine abgeänderte Fahrbahn (Plan). Anhand seiner geplanten Vorgehensweise bewegt sich der Agent dann mithilfe seiner Aktorik in die zuvor geplante Richtung (Act).

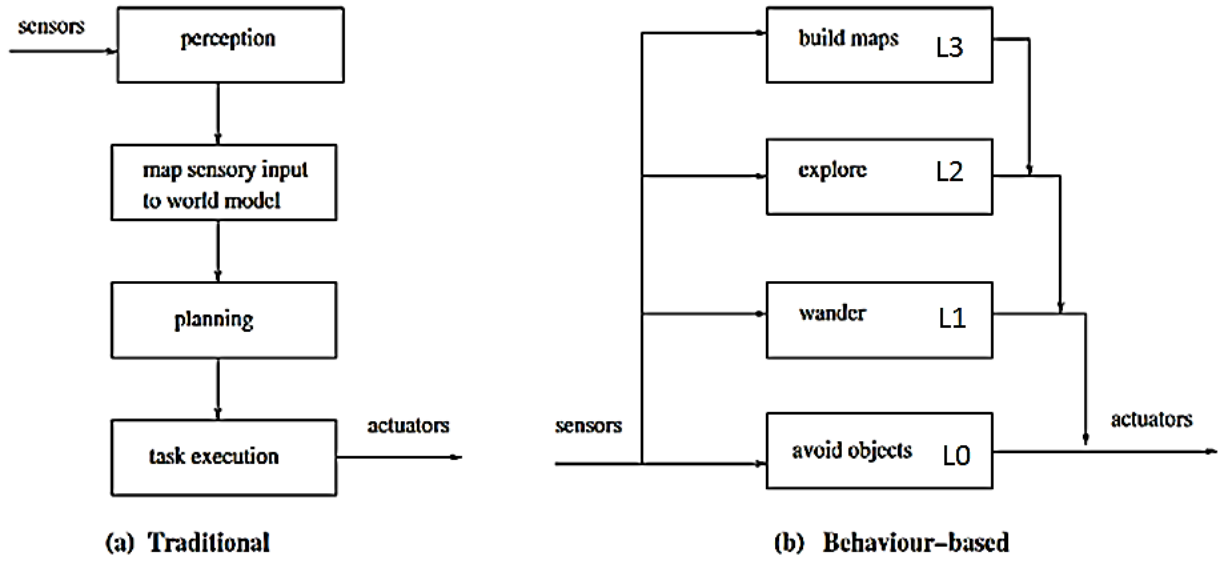

Abbildung 26: Sense-Plan-Act und Subsumption Architecture (Vgl. Greg Butler, Andrea Gantchev, Peter Grogono (2000), S.2)

Brooks hielt die Sense-Plan-Act Methode deshalb für impraktikabel, da sich ein weiterer Aufbau des Systems bei einer solchen Anordnung als schwierig gestaltet. Brooks ist auch der Meinung, dass die menschliche Intelligenz zu komplex ist, um sie in eine solche Struktur zu stecken.[48] Deshalb teilte er die Struktur in verhaltensbasierte Schichten ein (vgl. Abbildung 26 (b)). Jedes Verhalten bekommt einen bestimmten Level. Level 0 spiegelt das einfachste Verhalten dar. Je höher das Level, desto komplexer das jeweilige Verhalten. Level 4 könnte z.B. das strategische Vorausplanen anhand der internen Karte sein. So kann der Agent um weitere intelligente Verhaltensschichten erweitert werden, ohne, dass andere davon beeinträchtigt werden. Auch wenn einzelne Verhaltensschichten ausfallen sollten, können die noch intakten benutzt werden, um weiterhin das Ziel zu verfolgen. Dies hat den Ursprung darin, dass Ein- und Ausgänge unterdrückt werden können (vgl. Abbildung 27).

[48] Vgl. Butler, G., et al. (2000), S.1

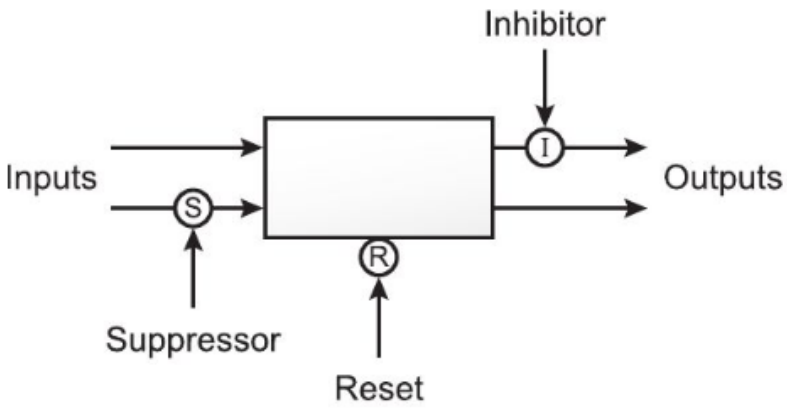

Abbildung 27: Subsumption Concept (Vgl. Jiuguang Wang (2008))

Dadurch können Verhaltensmuster mit höherem Level Verhaltensmuster mit niedrigerem Level bei Bedarf unterordnen. Sprich Level 1 kann Daten über die Schnittstelle von Level 0 verschicken und somit den normalen Datenfluss unterbrechen. Die untergeordneten Verhaltensmuster funktionieren trotzdem weiterhin so, als würde es keine übergeordneten Verhaltensmuster geben (vgl. Abbildung 28).

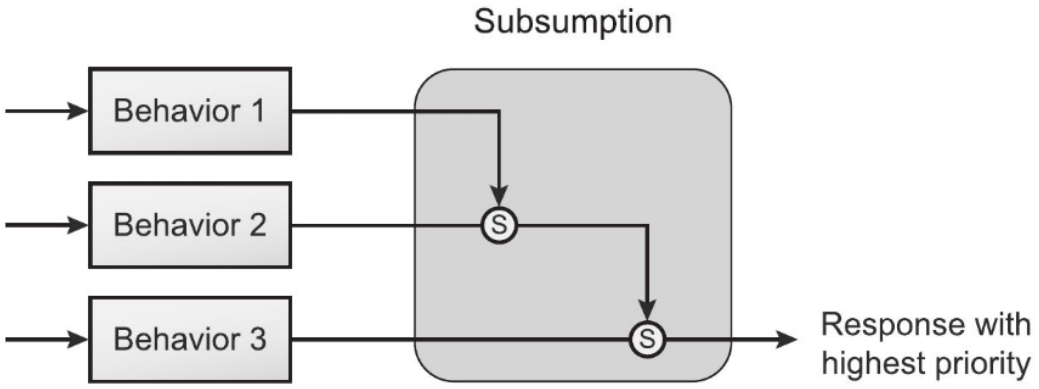

Abbildung 28: Subsumption Architecture (Vgl. Logan Kearsley (o.J.))

6. Grundfunktionen der Roboter

Zur Implementierung der einzelnen Funktionsweisen des Roboters gab es verschiedene mögliche Ansätze. Ganz zu Beginn war es am einfachsten grundlegende Methoden die das Vorwärtsfahren direkt über if-Abfragen zu implementieren (siehe Abbildung 29).

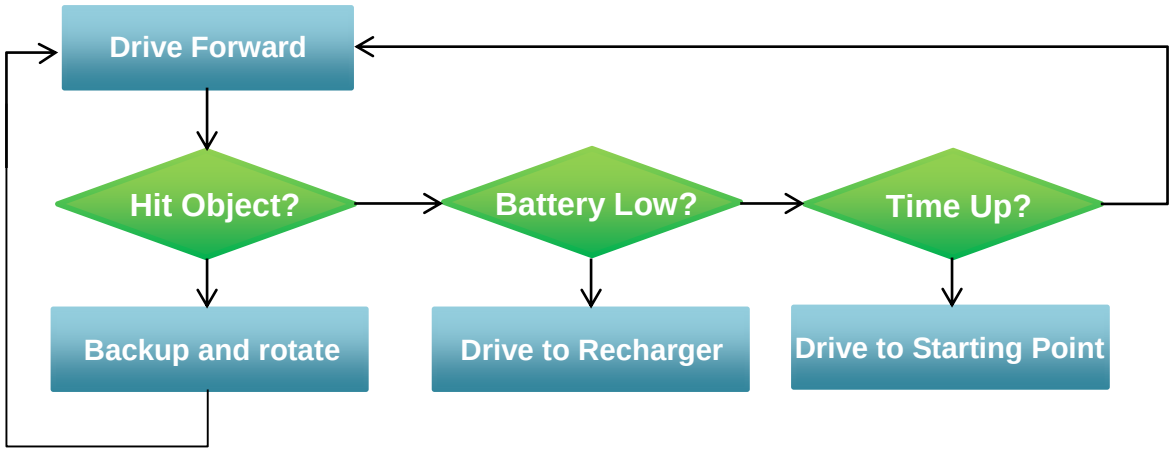

Abbildung 29: Verkettung von If-Abfragen

Dies ist zu Beginn sehr einfach, da keine weiteren Überlegungen über das endgültige Codedesign nötig sind und einfach eins nach dem anderen programmiert werden kann. Allerdings würde diese Methode den Code sehr schnell sehr unübersichtlich und schwer änderbar machen, da viele Verhalten ineinander verkettet sind und nicht einzeln für sich existieren. Es würde ein sogenannter „Spaghetti Code“ entwickelt werden.

Im Gegensatz dazu gibt es das sogenannte Behavior Control Model. Dies erfordert zu Beginn zwar mehr Planung für den zu entwickelnden Programmcode, sorgt aber am Ende für einen übersichtlichen und leicht änderbaren Code. Es werden einzelne Behaviors (Verhaltensmuster) implementiert, die getrennt koexistieren. Es kann also jederzeit ein weiteres Behavior implementiert oder ein anderes entfernt werden, ohne dass der gesamte Programmcode wie im vorigen Beispiel mühsam angepasst werden muss.

Über LeJOS lässt sich dies mit der Behavior API umsetzen. Über das Behavior Interface lassen sich die einzelnen Verhaltensmuster implementieren, wobei das Interface

selbst sehr grundlegend ausgelegt ist. Sobald alle Behaviors implementiert sind, werden sie einem Arbitrator (dt. Vermittler) übergeben, der sich darum kümmert, welche davon aktiviert werden. Das Behavior Interface ist so ausgelegt, dass es für jedes Verhalten drei grundlegende Methoden gibt, die dieses bestimmten.

```
1.  boolean takeControl()
```

Quellcode 1: takeControl

Bei der takeControl() Methode (siehe Quellcode1) wird ein Boolean-Wert zurückgegeben, der mitteilt, wann ein bestimmtes Verhalten aktiviert werden soll. Wenn somit beispielsweise ein Berührungssensor darauf hinweist, dass der Roboter gegen ein Hindernis gefahren ist, sollte diese Methode „wahr“ zurückgeben und somit das dazugehörige Verhalten anstoßen.

```
1.  void action()
```

Quellcode 2: action

Diese Methode definiert das Verhalten des Roboters, nachdem mithilfe von takeControl() herausgefunden wurde, dass auf etwas reagiert werden muss. Wenn wie im vorigen Beispiel der Roboter gegen ein Hindernis gefahren ist und die Methode action() dadurch aufgerufen wurde, kann hier beispielsweise definiert werden, dass der Roboter sich von dem Hindernis wegbewegen soll.

```
1.  void suppress()
```

Quellcode 3: suppress

Der Code in der suppress() Methode stoppt die action() Methode. Er kann auch dazu verwendet werden Informationen zu updaten bevor das Behavior vollständig beendet wird.

Mithilfe dieser drei Methoden lassen sich die Verhaltensweisen leicht implementieren. Wenn ein Roboter also drei unabhängige Behaviors haben soll, dann müssen drei Klassen implementiert werden, wobei jede das Behavior Interface implementiert. Wenn die

Klassen fertig geschrieben sind müssen die einzelnen Behaviors an den Arbitrator gegeben werden.

```
1.  public Arbitrator(Behavior [] behaviors)
```

Quellcode 4: Arbitrator

Mithilfe dieser Methode in Quellcode 4 wird ein Arbitrator erzeugt, welcher regelt wann die einzelnen Behaviors aktiviert werden. Dazu wird der Methode ein Array mit allen gewünschten Behaviors übergeben. Wichtig zu wissen ist hierbei, dass die Priorität der Behaviors von ihrer Array-Nummer abhängt. Je höher diese ist, desto höher ist die Priorität des Behaviors.

```
1.  public void start()
```

Quellcode 5: start Arbitrator

Mithilfe von start() wird der Arbitrator gestartet und beginnt damit zu entscheiden welche Verhaltensweisen der Roboter anwendet. Dabei werden die takeControl() Methoden aller Behaviors aufgerufen, beginnend mit der höchsten Arraynummer. Er arbeitet sich somit durch die ganzen Behaviors, bis eine der Funktionen „wahr“ zurückgibt und somit die jeweilige action() Methode dazu aufgerufen wird. Wenn zwei Behaviors beide unter takeControl() “wahr” zurückgeben, wird nur die aufgerufen, welche höher priorisiert ist (siehe Abbildung 30).[49]

[49] Vgl. o.V. (o.J.b), S. 55ff.

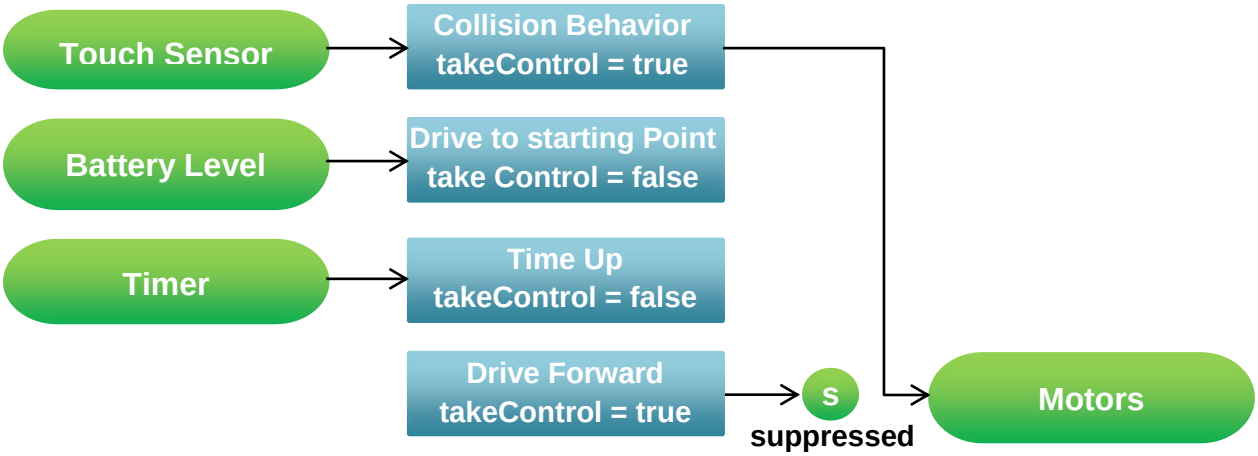

Abbildung 30: Priorisierte Behaviors

6.1 Ausweichen

Bevor die einzelnen Roboter komplexere Verhaltensweisen annehmen können, müssen diese zunächst in der Lage sein, sich frei Fortbewegen zu können und Hindernisse zu bewältigen. Hierzu wurde mithilfe des Behavior Models ein HitObject Behavior implementiert. Zunächst muss der Roboter in der Lage sein zu erkennen, dass sich ein Hindernis auf seinem Weg befinden, welches es zu bewältigen gilt. Hierfür werden sowohl der Berührungs-, als auch der Ultraschallsensor verwendet (siehe Quellcode 6).

```
1.  public boolean takeControl() {
2.          return touch.isPressed() || ultrasonic.getDistance() < 25;
3.      }
```

Quellcode 6: HitObject - takeControl

Sobald entweder der Berührungssensor gedrückt wird, oder der Ultraschallsensor eine Entfernung von unter 25cm empfängt, wird die action-Methode des Behaviors aufgerufen. Obwohl im Normalfall ein Sensor hier ausgereicht hätte, hat es sich als sinnvoll erwiesen, beide zu verwenden, da zum einem einige Objekte, dem Ultraschallsensor verborgen blieben, wenn sie zu klein waren, oder der Berührungssensor nicht reagiert, wenn der Roboter nicht frontal genug auf das Hindernis zufährt. Mithilfe von beiden Sensoren wurde damit die größtmögliche Hindernisabdeckung erzeugt.

```
1.  public void action() {
2.        suppressed = false;
3.        Motor.B.rotate(-180, true);
4.        Motor.C.rotate(-360, true);
5.
6.        while( Motor.C.isMoving() && !suppressed )
7.          Thread.yield();
8.
9.        Motor.B.stop();
10.       Motor.C.stop();
11.   }
```

Quellcode 7: HitObject – action

Die beiden Motoren B (links) und C (rechts) reagieren auf das erkannte Hindernis, indem der Roboter zunächst mit beiden Rädern zurückfährt und dann nur noch eines zum Zurückfahren verwendet, um sich zu drehen. Auffällig ist hierbei, dass der Roboter also, sobald er ein Hindernis erkennt, dieses um die rechte Seite umfährt. Aufgrund der Ungenauigkeit der Sensoren und Motoren bei den Lego NXT ist schon eine gewisse Zufälligkeit der letztendlichen Richtung, die der Roboter nach einem Hindernis einschlägt, erreicht. Somit drehen sich die Roboter nicht immer genau um den gleichen Winkel, sondern es gibt dennoch starke Abweichungen, welche in diesem Fall sinnvoll sind, damit der Roboter im Zweifelsfall, nicht immer dieselbe Route abfährt, sondern in der Lage ist, nach gewisser Zeit die gesamte Fläche abzufahren.

6.2 Suchen

Das Suchen wurde am Beispiel des finden eines roten Gegenstandes mithilfe des Farbsensors implementiert. Sobald der Farbsensor einen roten Gegenstand wahrnimmt, wird die action des Behaviors aufgerufen (siehe Quellcode 8).

```
1.  public boolean takeControl() {
2.        return color.getColorID() == Color.RED;
3.        }
```

Quellcode 8: FindObject - takeControl

Obwohl die Farbe erkannt wird, muss diese sehr nah am Sensor sein, um registriert zu werden. Nachdem dies erfolgt ist, reagiert der Roboter, indem er den anderen Robotern eine Information schickt, dass das zu suchende Objekt gefunden worden ist.

```
1.  public void action() {
2.              new SendInformation().sendinformation(1);
3.          }
```

Quellcode 9: Find Object – action

Wie das Senden von Informationen funktioniert, wird unter *7.1 Kommunikation unter den Robotern* näher beschrieben.

6.3 Folgen

Das Folgen wurde mithilfe des Geräuschsensors implementiert. Voraussetzung ist ein durchgängiges lauteres Geräusch und eine ansonsten ruhige Umgebung, sodass der Geräuschsensor das primäre Geräusch als das lauteste erkennt und nicht von Hintergrundgeräuschen gestört wird.

```
1.  public boolean takeControl() {
2.       return sound.readValue()<90;
3.    }
```

Quellcode 10: FindSound - takeControl

Sobald die takeControl() des Behavios angibt, dass der Geräuschpegel kleiner als 90 ist (siehe Quellcode 10), wird die action Methode aufgerufen (siehe Quellcode 11).

```
1.   public void action() {
2.      suppressed = false;
3.      Motor.B.forward();
4.      Motor.C.forward();
5.      for(int i=0; i<65 && !suppressed;i++){
6.          Delay.msDelay(60);
7.          Thread.yield();
8.          }
9.      Motor.B.stop();
10.     Motor.C.stop();
11.     if (suppressed) return;
12.     int volume = 0;
13.     while (volume < sound.readValue()) {
```

```
14.         volume = sound.readValue();
15.         Motor.B.forward();
16.         Delay.msDelay(200);
17.         Motor.B.stop();
18.     }
19. }
```

Quellcode 11: FindSound – action

Zunächst bewegen sich die Roboter drei Sekunden lange vorwärts. Dies wurde mithilfe einer while-Schleife implementiert, da berücksichtigt werden muss, dass ein höher Priorisiertes Behavior, wie das Ausweichen auftreten könnte, sodass diese 3 Sekunden nicht vollständig eingehalten werden würden. Mithilfe der for-Schleife wird somit 65 Mal jeweils 60ms der Thread am Laufen gehalten, woraufhin abgefragt wird, ob ein anderes Behavior aktiv wird, oder die drei Sekunden abgelaufen sind. Nach dieser Schleife werden die Motoren zunächst angehalten. Falls die Schleife nun aufgrund des suppressed Wertes beendet wurde, wird die Methode mithilfe von return beendet. Falls dies nicht der Fall ist, richtet sich der Roboter nun in Richtung des Geräusches aus. Dies macht er, indem er die Lautstärke, die er empfängt mit einer vergleicht, die er kurz zuvor empfangen hat. Wenn die, die er im Moment empfängt lauter ist, dreht sich er Roboter 200ms und vergleicht erneut, bis diese Bedingung nicht mehr zutrifft. Daraufhin beginnt die Methode erneut und der Roboter fährt wieder drei Sekunden geradeaus.

Neben dieser Methode, welche sehr von dem Geräuschsensor abhängig ist, besteht noch die Möglichkeit der Orts- bzw. Positionsbestimmung der Roboter. Hierbei gibt es einen Unterschied zwischen der globalen (absoluten) und lokalen (relativen) Lokalisierung. Bei der globalen Lokalisierung wird die Position des Roboters anhand von sogenannten bekannten Landmarken bestimmt. Diese sind markante Punkte, welche verschiedene Eigenschaften besitzen können und sich somit eindeutig von den Robotern erkennen lassen. Wenn ein Roboter somit registriert, dass ein solcher Punkt gefunden wurde, lässt sich die Position des Roboters selbst schlussfolgern. Bei der lokalen Lokalisierung hingegen bestimmt der Roboter seine Position anhand der Aufsummierung von Positionsänderungen. Ein Beispiel hierfür ist die Odometrieberechnung. Hierbei werden über die Drehsensoren Roboter an den Rädern berechnet, welche Strecke die-

se zurückgelegt haben. Standardmäßig lässt sich die zurückgelegte Strecke anhand der Geschwindigkeit v und der Zeit t berechnen.

$$\Delta s_l = v_l \cdot \Delta t$$

$$\Delta s_r = v_r \cdot \Delta t$$

Formel 1: Streckenberechnung anhand der Geschwindigkeit (Vgl. Berns, Karsten/ Schmidt, Daniel (2010), S. 157)

Da der NXT Roboter allerdings nicht über einen Geschwindigkeitssensor verfügt, sondern lediglich die Umdrehungen der Räder berechnen kann, muss eine andere Methode der Berechnung herangezogen werden. Somit kann die zurückgelegte Strecke in einer bestimmten Zeit über die Radumdrehungen T und den Radumfang U berechnet werden.

$$\Delta s_l = T_l \cdot U$$

$$\Delta s_r = T_r \cdot U$$

Formel 2: Streckenberechnung anhand der Radumdrehung (Vgl. Berns, Karsten/ Schmidt, Daniel (2010), S. 157)

Die zurückgelegte Strecke ausgehend vom Roboterzentrum s_m ergibt sich aus den gemittelten Einzelstrecken des linken und rechten Rades.[50]

$$\Delta s_m = \frac{\Delta s_l + \Delta s_r}{2}$$

Formel 3: Berechnung der zurückgelegten Strecke (Vgl. Berns, Karsten/ Schmidt, Daniel (2010), S. 157)

Auch wenn diese Methode sehr genau sein kann, wurde sie aufgrund der zu ungenauen Sensoren der NXTs hier nicht verwendet. Bei kleineren Messfehlern werden diese mit den richtigen Werten aufsummiert und verfälschen damit das Ergebnis enorm. Somit konnte festgestellt werden, dass man die NXTs nicht mehrmals mit der gleichen Me-

[50] Vgl. Berns, Karsten/ Schmidt, Daniel (2010), S. 155ff.

thode um beispielsweise 90° drehen kann, ohne dass große Abweichungen in den Ergebnissen entstehen. In der Praxis mit besseren Sensoren wäre diese Methode allerdings anwendungsfreundlicher und genauer. Die Roboter wären unabhängig von Hintergrundgeräuschen und sonstigen Störungen von außen.

7. Praktische Realisierung eines Roboterschwarms

Nachdem das Schwarmverhalten in der Theorie schon anhand mehrerer Beispiele in der Tierwelt erläutert wurde, wird in diesem Kapitel ein konkretes Beispiel zur Implementierung erläutert. Hierbei werden die zuvor erläuterten, implementierten Behaviors *6.1 Ausweichen, 6.2 Suchen* und *6.3 Folgen* verwendet.

7.1 Kommunikation unter den Robotern

Die Kommunikation unter den Robotern ist ein wichtiger Bestandteil um diese zu einem funktionierenden Schwarm zu machen. Ohne Kommunikation würde jeder Roboter die Aufgaben für sich erledigen, sodass keine Vorteile durch Schwarmverhalten entstehen könnten. Es gibt hierbei zwei Möglichkeiten, wie diese Kommunikation stattfinden kann. Zum einem können die Roboter direkt miteinander kommunizieren, indem sie benötigte Informationen untereinander verschicken. Die zweite Möglichkeit wäre es, eine zentrale Station (wie den PC) zu implementieren, welche alle Informationen entgegennimmt, diese verarbeitet und an die benötigten Roboter weiterleitet. Dies wäre eine indirekte Methode, da die Roboter nicht den direkten Weg der Kommunikation wählen, sondern ihre Anweisungen von einer zentralen Station bekommen und dieser wiederum ihre aufgenommenen Informationen zurückschicken. In den folgenden Beispielen wurde die erste Methode der direkten Kommunikation gewählt, da lediglich mit drei Robotern der Schwarm simuliert wurde, was die direkte Kommunikation leicht möglich macht. Sollten allerdings mehr Roboter dazukommen, bzw. komplexere Verwendungsweisen aufkommen, bei denen die Auswertung der Informationen eine größere Rolle spielt, macht es mehr Sinn, die zweite Methode zu wählen. Somit könnte die Bearbeitung der Informati-

onen auf den PC ausgelagert werden, da sie für die einzelnen Roboter nicht von Belangen und diese somit weiterhin einfach zu verstehen wären.

Die Kommunikation erfolgt mithilfe von Bluetooth und zwei geschriebenen Klassen. Die RecieveInformation-Klasse sorgt dafür, dass Nachrichten empfangen werden, während die SendInformation-Klasse Nachrichten versendet.

7.1.1 Informationen senden

Damit ein Roboter anderen Robotern oder Instanzen Informationen zusenden kann, muss zunächst eine Verbindung zu einem anderen Roboter über Bluetooth hergestellt werden. Hierzu werden die Namen der anderen Roboter benötigt, mit denen der Roboter bereits eine Verbindung hatte. Diese kann zuvor einmal manuell eingestellt werden. Wenn dann ein RemoteDevice über den Namen definiert wurde, wird eine Bluetooth-Connection mit diesem aufgebaut. Falls diese erfolgreich ist, wird ein DataOutputStream implementiert, durch welchen die Nachrichten verschickt werden können. Über diesen wird hier ein Integer versendet, welcher für den Empfänger eine bestimmte Bedeutung hat. Sobald die Nachricht versendet wurde, wird der Outputstream wieder gelöscht und die Verbindung geschlossen, damit bei Bedarf eine neue Verbindung aufgebaut werden kann. Dieser Prozess wird in einer Schleife mit allen Robotern ausgeführt. Darunter ist auch der Name des Geräts, welcher die Nachricht versendet, damit das Array mit den Namen nicht auf jeden Roboter individuell angepasst werden muss. Da dieser dem Gerät aber als Bluetooth Verbindung nicht bekannt ist, wird dieser Versuch einer Verbindung von dem jeweiligen Roboter schnell übersprungen.

```
1.  String name[] = {"GD2017-1", "GD2017-4", "GD2017-3"};
2.  for (int i = 0; i<name.length; i++) {
3.      RemoteDevice remotedevice = Bluetooth.getKnownDevice(name[i]);
4.      if (remotedevice == null) {
5.          Delay.msDelay(2000);
6.      }
7.      else {
8.           BTConnection btconnection = Bluetooth.connect(remotedevice);
9.          if (btconnection == null) {
10.             Delay.msDelay(2000);
11.         }
```

```
12.         else {
13.             DataOutputStream dos = btconnection.openDataOutputStream();
14.             try {
15.                 // send this information
16.                 dos.writeInt(information);
17.                 dos.flush();
18.             } catch (IOException ioe) {}
19.             try {
20.                 dos.close();
21.                 btconnection.close();
22.             } catch (IOException ioe) {}
23.             Delay.msDelay(2000);
24.     }
25. }
```

Quellcode 12: SendInformation

7.1.2 Informationen empfangen

Das Empfangen von Informationen verläuft, indem das jeweilige Gerät auf die Anfrage einer Verbindung wartet. Sobald ein anderes Gerät eine Verbindung aufbauen will, wird diese aufgebaut. Daraufhin wird ein DataInputStream implementiert, welcher im Gegensatz zum DataOutputStream für das Empfangen von Nachrichten sorgt. Über diesen wird die Nachricht, in diesem Fall ein Integer-Wert, empfangen und gespeichert, woraufhin der InputStream und daraufhin die Bluetooth-Connection wieder geschlossen werden.

```
1.  btconnection = Bluetooth.waitForConnection();
2.          DataInputStream dis = btconnection.openDataInputStream();
3.          Delay.msDelay(500);
4.          try {
5.              int n = dis.readInt();
6.              dis.close();
7.              Delay.msDelay(3000);
8.              btconnection.close();
9.              return n;
10.         } catch (IOException e) {}
11.         return 0;
```

Quellcode 13: RecieveInformation

7.2 Suchen eines Objektes

Die in den Unterkapiteln von Kapitel *6 Grundfunktionen der Roboter* genannten Behaviors und die in Kapitel *7.1 Kommunikation unter den Robotern* erläuterten Methoden

werden in diesem Kapitel zusammen zur Implementierung eines Szenarios verwendet. In diesem sollen die Roboter im Schwarm nach einem roten Objekt suchen. Sobald ein Roboter dieses gefunden hat, sendet er eine Nachricht an die anderen Roboter und macht ein durchgängiges Geräusch, welchem die anderen folgen, um zu dem Roboter zu fahren bzw. um diesen finden zu können.

Zunächst müssen die Roboter hierzu wie bereits erwähnt eine zufällige Route abfahren, dabei Hindernissen ausweichen und auf ein aufkommendes rotes Objekt reagieren können (siehe Quellcode 14).

```
1.  public static class FindObjectBehavior extends Thread {
2.          @Override
3.          public void run() {
4.          Behavior driveforward = new DriveForward();
5.              Behavior hitobstacle = new HitObstacle(SensorPort.S2, SensorPort.S3);
6.              Behavior findobject = new FindObject(SensorPort.S1);
7.              Behavior [] behaviorarray = {driveforward, hitobstacle, findobject};
8.              Arbitrator arby = new Arbitrator(behaviorarray);
9.              arby.start();
10.         }
11.     }
```

Quellcode 14: FindObject – Thread

Die beiden Behaviors HitObstacle und FindObject wurden bereits in den Kapiteln *6.1 Ausweichen* und *6.2 Suchen* näher erläutert. Das Behavior DriveForward lässt den Roboter hier lediglich vorwärtsfahren. Durch die Reihenfolge dieser Behaviors im Array werden diese priorisiert. Demnach ist die Suche nach dem Objekt, bzw. die Aktion, sobald er dieses gefunden hat, am wichtigsten und danach folgt das Ausweichen von Hindernissen.

Neben diesem Verhaltensmodell der Roboter gibt es noch jenes, welches die Roboter nachdem das Objekt gefunden wurde, zu dem entsprechenden Roboter fahren lässt, welcher es gefunden hat (siehe Quellcode 15).

```
1.  public static class FindSoundBehavior extends Thread {
2.          @Override
3.          public void run() {
4.              Behavior hitobstacle = new HitObstacle(SensorPort.S2, SensorPort.S3);
```

```
5.              Behavior findsound = new FindSound(SensorPort.S4);
6.              Behavior [] behaviorarray = {findsound, hitobstacle};
7.              Arbitrator arby = new Arbitrator(behaviorarray);
8.              arby.start();
9.          }
10.     }
```

Quellcode 15: FindSound - Thread

Hierbei werden die bereits erläuterten Behaviors HitObstacle und FindSound verwendet. Die Priorisierung sorgt hierbei wiederum dafür, dass der Roboter versucht das Objekt zu finden, ohne dabei mit Hindernissen zu kollidieren.

Beide Verhaltensmodelle wurden in Form von Threads implementiert. Um das bereits beschriebene Szenario ausführen zu lassen müssen alle Roboter zunächst das erste Verhalten ausführen, d.h. der Thread FindObjectBehavior wird gestartet. Alle Roboter führen nun dieses Verhalten aus. Um allerdings zu bemerken, ob ein anderer Roboter das Objekt bereits gefunden hat, muss gleichzeitig die Methode recieveInformation() ablaufen, die es den Robotern möglich macht, zu jeder Zeit Nachrichten empfangen zu können, unabhängig ihres momentanen Verhaltens. Sobald ein Roboter also ein rotes Objekt findet. Sendet er ein Signal mit dem Integerwert „1“ an alle anderen Roboter. Dieses sorgt dafür, dass ein Interrupt des Threads mit dem momentanen Verhaltensmodell gesendet wird, welcher dieses stoppt. Sobald dies erfolgt ist, wird das Verhalten FindSoundBehavior gestartet, welches dafür sorgt, dass die übrigen Roboter dem Geräusch folgen, welches der Roboter erzeugt, der das Objekt gefunden hat.

```
1.  public static void main(String[] args) {
2.
3.          Thread findobjectbehavior = new FindObjectBehavior();
4.          findobjectbehavior.start();
5.          Thread findsoundbehavior = new FindSoundBehavior();
6.
7.          if (new RecieveInformation().recieveinformation() == 1){
8.              findobjectbehavior.interrupt();
9.              while (!findobjectbehavior.isInterrupted());
10.             findsoundbehavior.start();
11.         }
12.     }
```

Quellcode 16: Kontrolle der Verhalten über Threads

In Abbildung 31 lassen sich die Zusammenhänge nochmals anhand eines Klassendiagramms erklären.

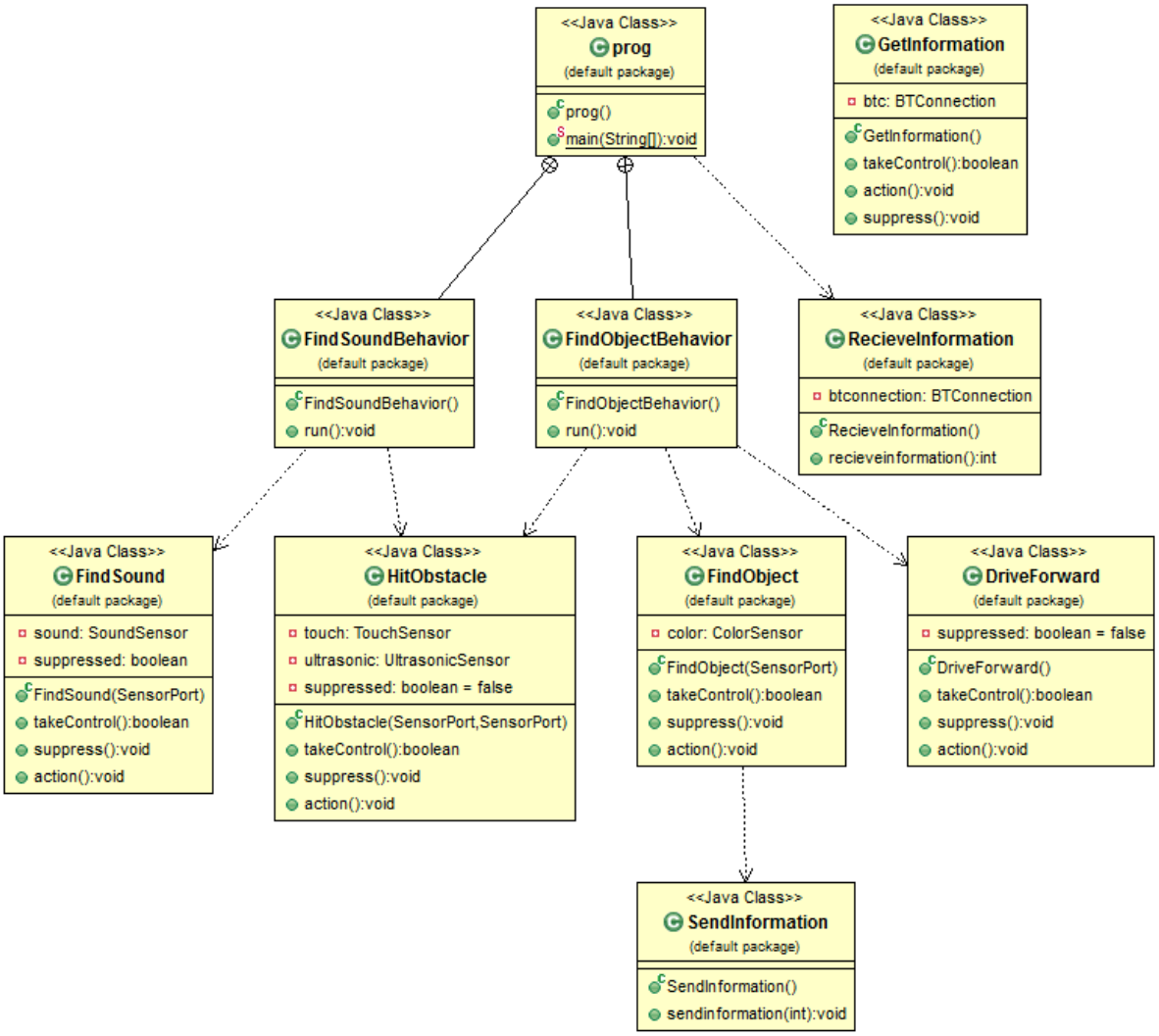

Abbildung 31: Klassendiagramm

7.3 Schwarmintelligenz der Roboter

Da, wie bereits zuvor erklärt, die Begriffe Schwarmverhalten und Schwarmintelligenz oft als Synonym benutzt werden und sich hier die Experten nicht einig sind, ab wann man genau von Schwarmintelligenz sprechen kann, lässt dieses Kapitel viel Interpretationsfreiheit.

Das oben implementierte Szenario wurde anhand des Schwarmverhaltens realisiert. Zu sehen ist hier, wie auch bei den Tierschwärmen, dass die Agenten lediglich auf ihre Umwelt reagieren, ohne groß Entscheidungen treffen zu müssen. Sei es bei der Kollisionsvermeidung mit einem Objekt oder das Folgen des lauten Tons, der Agent besitzt keinerlei Intelligenz, sprich er kann aufgrund seiner Sensordaten kein vernünftiges und zweckvolles Handeln ableiten. Die Agenten stellen hier also einfache Reflexagenten dar.

Möchte man nun den nächsten Schritt gehen und Schwarmintelligenz realisieren, bräuchte man schon nutzenbasierte Agententypen. Diese planen bzw. berechnen ihre Fahrt voraus und stellen für jede Möglichkeit den Nutzen in Frage. Dieser Agententyp schaut also, mit welcher Strecke er den höchsten Nutzen erzielen kann und handelt dann dementsprechend, um sein Ziel zu erreichen. Er handelt also genau nach der Definition für Intelligenz, nämlich sinn- und zweckvolles Handeln anhand der Sensordaten abzuleiten. Mit den Lego NXT Agenten war es jedoch nicht möglich, solch komplexe Verhaltensweisen darzustellen, da wie bereits zuvor erwähnt, die Sensoren sehr ungenau sind und man keinen Nutzen aus diesem Agententyp hätte hervorrufen können. Ein weiteres Konzept wäre, die recht einfachen Agenten an eine zentrale Stelle zu koppeln und mit dieser kommunizieren zu lassen. Diese Zentraleinheit bildet das Gehirn des Schwarms. Es verarbeitet sämtliche Daten, die die Agenten mit ihren Sensoren erfassen, wertet diese aus und leitet zweckvolles Handeln ab. Ist dies geschehen, gibt es den einzelnen Agenten Anweisungen, wie diese nun weiter Verfahren, bzw. in welche Richtung sich diese nun bewegen sollen. Als Beispiel hier könnte man einen Unterwas-

serschwarm nehmen, der nach Wrackteilen sucht. Die einzelnen Roboter können durchaus einfache Reflexagenten sein, die lediglich dazu dienen, den Meeresgrund abzusuchen. Die Zentraleinheit bildet anhand der eintreffenden Sensordaten ein Modell der Unterwasserumgebung und besitzt zusätzlich ein Suchraster bzw. ein Suchverfahren, welches es darauf anwendet. So dirigiert es den Roboterschwarm systematisch, um eine große Unterwasserfläche absuchen zu können. Dieses Konzept vereint Schwarmverhalten und Schwarmintelligenz. Die Suchagenten verfahren lediglich nach den Instinkten des Schwarmverhaltens, während die Zentraleinheit das Gehirn bildet und die Intelligenz für den Schwarm implementiert. So können die Agenten auch einfach als erweiterte und unabhängige Sensoren gesehen werden, die die Zentraleinheit mit Informationen versorgen, was bezweifeln lässt, ob dies noch etwas mit Schwarmintelligenz zu tun hat, da nicht jedes Individuum im Schwarm auch selbst zweckvolles Handeln ableiten kann.

8. Reflexion

Schwarmverhalten und Schwarmintelligenz bieten schon jetzt enorm viele Einsatz- und Optimierungsmöglichkeiten für bestehende Aufgaben und Probleme. Die Forschung im Bereich Schwarmverhalten nimmt mehr und mehr zu, was sehr vielversprechende Zukunftsaussichten zulässt. Bisherige Lösungsansätze der Technik werden in Zukunft höchstwahrscheinlich durch Roboterschwärme ersetzt, wie z.B. in den unter *2.2.4 Swarm Bots* genannten Anwendungsmöglichkeiten.

Wichtig für einen genau agierenden und funktionierenden Schwarm sind in erster Linie die Sensoren an den einzelnen Swarm Bots. Sind diese ungenau, oder besitzen eine sehr hohe Ausfallwahrscheinlichkeit, ist der Schwarm ebenfalls ungenau bei der Positionsbestimmung und neigt zu Fehlverhalten. Selbiges Problem bestand auch hier bei den Lego NXT Robotern, welches unter 6*.3 Folgen* genauer erläutert wurde. Dies machte eine exakte Positionsbestimmung unmöglich, wodurch oft Fehlverhalten zutage

getreten sind. Möchte man beispielsweise einen Unterwasserschwarm herstellen, der auf dem Grund der Meere nach Wracks sucht, sind die Sensoren zur Positionsbestimmung ungemein wichtig. Funktionieren diese nur ungenau, oder fallen gar aus, dann sind die Swarm Bots nicht mehr in der Lage, ihre Position genau zu bestimmen oder Informationen zu sammeln. Ebenso wichtig ist ein geeignetes Kommunikationsnetz für den Schwarm. In dieser Studienarbeit funktionierte dies sehr gut über Bluetooth, was bei Unterwasserschwärmen nicht empfehlenswert wäre, da hier viel größere Distanzen zur Kommunikation überbrückt werden müssen.

Insgesamt gesehen erlauben Roboterschwärme eine autonome und automatisierte Lösung für verschiedenste Aufgabengebiete. Dieser Wandel wird auf der einen Seite viele Arbeitsstellen vernichten, aber dafür in den Bereichen wie Entwicklung und Herstellung solcher Roboterschwärme verhältnismäßig mehr Arbeitsstellen schaffen. Dies hat den Grund, da die Hard- und Software der Schwärme an die verschiedensten Aufgaben, sprich an die dort vorherrschenden Anforderungen, adaptiert werden müssen, um einen gut funktionierenden und widerstandsfähigen Schwarm entwickeln zu können.

Literaturverzeichnis

Anders, C. (2010) Grundlagen der Künstlichen Intelligenz, 2010, unter: https://www.cs.hs-rm.de/~linn/fachsem0910/anders/Anders.pdf.

Axel Springer SE (2009) Menschen verhalten sich in Gruppen wie Fische, unter: https://www.welt.de/wissenschaft/article5057937/Menschen-verhalten-sich-in-Gruppen-wie-Fische.html, rev. 11.02.2018.

Bäcker, B., et al. (2016) Partikelschwarmoptimierung am Beispiel des Traveling Salesman Problem, SS16, unter: http://image.informatik.htw-aalen.de/Thierauf/Seminar/Ausarbeitungen-16SS/PSO.pdf, rev. 09.03.2018.

Barrie, A (2018) How deadly drone swarms will help US troops on the frontline, unter: http://www.foxnews.com/tech/2018/01/11/how-deadly-drone-swarms-will-help-us-troops-on-frontline.html, rev. 24.03.2018.

Bayrischer Rundfunk (2017) Roboterschwärme: Können Sie uns helfen? - Faszination Wissen, unter: https://www.youtube.com/watch?v=dQi1kgvaQHE, rev. 24.03.2018.

Berns, Karsten/ Schmidt, Daniel (2010) Programmierung mit LEGO MINDSTORMS NXT, Springer, 2010.

Bezirksimkerei Metzingen e.V. (2017) Wie viele Bienen leben in eine Bienenvolk?, unter: http://www.imkerverein-metzingen.de/wissen/faq/wie-viele-bienen-leben-einem-bienenvolk, rev. 14.02.2018.

Bibliographisches Institut GmbH (2018a) Verhalten, unter: https://www.duden.de/rechtschreibung/verhalten_handeln_sein_reagieren#Bedeutung1a, rev. 07.02.2018.

Bibliographisches Institut GmbH (2018b) Intelligenz, unter: https://www.duden.de/rechtschreibung/Intelligenz, rev. 07.02.2018.

Bienengarten Ahrensburg (2018) Bienengarten, 2018, unter: http://www.bienenlehrgarten-ahrensburg.de/images/bienen_infos/rundtanz.png, rev. 21.04.2018.

Blum, Daniel (2003) Ant Colony Optimization (ACO), Universität Dortmund, 2003, unter: http://ls11-www.cs.tu-dortmund.de/lehre/SoSe03/PG431/Ausarbeitungen/ACO.pdf, rev. 08.02.2018.

Bogon, Tjorben (2012) Agentenbasierte Schwarmintelligenz, Dissertation Universität Trier, Springer Vieweg, 2012.

Bornemann, K. (o.J.) Schwarmintelligenz, unter: http://schreibwerkstatt.katrin-krieger.de/kai/wp-content/uploads/sites/28/2016/06/SchwarmIntelligenzKaiBornemann.pdf, rev. 13.02.2018.

Bosse, Sascha (2016) Optimierung der Kosten und Verfügbarkeit von IT-Dienstleistungen durch Lösung eines Redundanz-Allokation-Problems, unter: https://d-nb.info/1103022083/34, rev. 16.03.2018.

Butler, G., et al. (2000) Object-Oriented Design of the Subsumption Architecture, unter: http://users.encs.concordia.ca/~gregb/home/PDF/bgg-spe2001.pdf, rev. 24.03.2018.

Dambeck, H. (2007) Schwarm-Experiment: Menschen sind auch nur Fische, unter: http://www.spiegel.de/wissenschaft/natur/schwarm-experiment-menschen-sind-auch-nur-fische-a-471179.html, rev. 11.02.2018.

Deutsche Post AG (2016) DHL Paketkopter 3.0, unter: http://www.dpdhl.com/de/presse/specials/paketkopter.html, rev. 24.03.2018.

Dorigo, M. / Stützle, T. (2004) Ant Colony Optimization, The MIT Press, 2004, S. 33.

Ephramac (2017) Partikelschwarmoptimierung, 2017, unter: https://de.wikipedia.org/wiki/Partikelschwarmoptimierung#/media/File:ParticleSwarmArrowsAnimation.gif, rev. 21.04.2018.

Fendt (2018) MARS: Robotiksystem zur Aussaat und exakten Dokumentation, unter: https://www.fendt.com/de/fendt-mars.html, rev. 24.03.2018.

FOCUS Online (2010) Kommunikationsnetze für Krisen- und Einsatzgebiete, unter: https://www.focus.de/wissen/videos/fliegende-roboterschwaerme-kommunikationsnetze-fuer-krisen-und-einsatzgebiete_vid_20203.html, rev. 24.03.2018.

Galton, F. (1907) Vox Populi, Nature No.75,unter: http://galton.org/cgi-bin/searchImages/galton/search/essays/pages/galton-1907-vox-

populi_1.htm ,1907, S.450 f.

Greg Butler, Andrea Gantchev, Peter Grogono (2000) Object-Oriented Design of the Subsumption Architecture, 2000, unter: http://users.encs.concordia.ca/~gregb/home/PDF/bgg-spe2001.pdf.

Hans-Ulrich Mährlen (2004) Armeisenalgorithmus, 2004, unter: https://upload.wikimedia.org/wikipedia/commons/1/1a/TSP_screendump.jpg, rev. 21.04.2018.

Himani/ Girdhar, A. (o. J.) Swarm Intelligence and Flocking Behavior, unter: http://research.ijcaonline.org/icaet2015/number10/icaet4146.pdf, rev. 02.02.2018.

honig-und-bienen.de (o.J.) Wie Bienen Kommunizieren, unter: http://honig-und-bienen.de/wie-bienen-kommunizieren/, rev. 14.02.2018.

Imkerpate.de (2014) Schwänzeltanz. Wie Bienen sich über die besten Futterquellen unterhalten: Wie ermitteln Bienen die Distanz?, unter: http://www.imkerpate.de/schwaenzeltanz/, rev. 14.02.2018.

ITG-Fotoflug (2017) Löschdrohne als Unterstüzung für die Feuerwehr, unter. https://www.itg-fotoflug.de/loesch-drohne-feuerwehr/, rev. 24.03.2018.

Jiuguang Wang (2008) Subsumption Architecture, 2008, unter: https://en.wikipedia.org/wiki/File:Subsumption_architecture.svg, rev. 21.04.2018.

Johann Dréo (2006) Ameisenalgorithmus, 2006, unter: https://de.wikipedia.org/wiki/Ameisenalgorithmus, rev. 04.03.2018.

Kramann, G. (2014) Agentensysteme, unter: http://www.kramann.info/72_COACH2/13_Skript/04_Agentensysteme/index.php, rev. 23.02.2018.

Ledermann, Thomas (2012) Partikel-Schwarm-Optimierung zur Objektlageerkennung in Tiefendaten, unter: https://elib.uni-stuttgart.de/bitstream/11682/4478/1/Diss_Ledermann.pdf, rev. 09.03.2018.

Liu, Yushan (2014) Partikelschwarmoptimierung für diskrete Probleme, unter: http://wwwmayr.informatik.tu- muenchen.de/konferenzen/Ferienakademie14/slides_papers/paper_Yushan_Liu.pdf, rev. 09.03.2018.

Logan Kearsley (o.J.) Hybrid AI & Machine Learning Systems, unter: http://slideplayer.com/slide/7383369/, rev. 21.04.2018.

Lorenz, J. /Rauhut, H. (2011) How social influence can undermine the wisdom of crowd effect, in: Proceedings of the National Academy of Sciences, 2011, unter: http://www.pnas.org/content/pnas/early/2011/05/10/1008636108.full.pdf, rev. 10.02.2018.

Loung Dinh Le (2013) Artificial Bee Colony Algorithm for Solving Optimal Power Flow Problem, 2013, unter: https://www.researchgate.net/figure/Flowchart-of-the-ABC-algorithm_fig1_259957594, rev. 21.04.2018.

Lümkemann, Ant Colony Optimization: Ausarbeitung zum Vortrag im Seminar „In-

Jan (2003) telligente Algorithmen“, Universität Bielefeld, 2003, unter: https://www.techfak.uni-biele-feld.de/ags/wbski/lehre/digiSA/WS0304/IntAlg/Ausarbeitungen/AntColony.pdf, rev. 08.02.2018.

Mandyam V. Srinivasan, et al. (2000) Honeybee Navigation: Nature and Calibration of the „Odometer“, in: Science No.5454 Vol. 287, S. 851-853, 2000, unter: https://www.hobos.de/media/user_upload/Images_pdfs_etc/publikationen_ebooks/77_Honey_Bee_Navigation_Kleinhenz.pdf, rev. 14.02.2018.

Morillo, Oscar (2014) Parameteroptimierung des Ballmodells humanoider Fußball spielender Roboter, unter: http://www.mi.fu-berlin.de/inf/groups/ag-ki/Theses/Completed-theses/Bachelor-theses/2014/Morillo-Victoria/Bachelor-Morillovictoria.pdf?1392198205, rev. 16.03.2018.

o.V. (o.J.a) Introduction: What is LeJOS NXJ, unter: http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/Intro.htm, rev. 12.02.2018.

o.V. (o.J.b) The leJOS NXJ Tutorial, unter: http://www.ctestlabs.org/robotteams/leJOSNXJTutorial.pdf, rev. 06.02.2018.

Ola Tande (2006) Fendts MARS-prosjekt, 2016, unter: http://gardsdrift/bildefelt/fendt_mars_2.jpg?itok=H9JXNVVe&c=7be8aa673d757b262ccc602b3aecf2b8, rev. 21.04.2018.

Pintscher, Lydia (2008) Schwarmintelligenz, Seminar Organic Computing, Universität Karlsruhe, 2008, unter: http://www.lydiapintscher.de/uni/schwarmintelligenzodp.pdf, rev. 05.12.2017.

Riccó, J. (2008) Weisheit der Bremer siegt über Experten, unter: http://scienceblogs.de/neurons/2008/08/27/weisheit-der-bremer-siegt-uber-experten/, rev. 09.02.2018.

S. Killen, S, et al. (2011) Aerobic capacity influences the spatial position of individuals within fish schools, in: Proceedings of the royal Society, 2011, unter: http://rspb.royalsocietypublishing.org/content/royprsb/early/2011/06/07/rspb.2011.1006.full.pdf, rev. 11.02.2018.

Schmidt, D. / Berns, K. (2010) Programmierung mit Lego Mindstorms NXT: Robotersysteme, Entwurfsmethodik, Algorithmen, Springer Verlag, 2010, S.54.

Schneider, G. B. (2009) Wenn Agenten sich streiten: Ein Agentenmodell zur Erforschung sozialer Konflikte, Kassel University Press GmbH, 2009.

Sedlacek, Klaus-Dieter (2010) Emergenz: Strukturen der Selbstorganisation in Natur und Technik, Book On Demand GmbH, 1.Auflage, Norderstedt, 2010.

Serban, N. (2017) Schwarmintelligenz in der Tourenplanung: Konzeption und Umsetzung eines didaktischen Beispiels, GRIN Verlag,2017, S.2

Spektrum Akademischer Verlag (1999) Ameisen, unter: http://www.spektrum.de/lexikon/biologie/ameisen/2698, rev.13.02.3018

Spektrum Akademischer Verlag (2001) Schwarmverhalten, unter: http://www.spektrum.de/lexikon/biologie-kompakt/schwarmverhalten/10520, rev. 06.02.2018

Stober, A. (2016) Schwarmintelligenz: von den Fischen lernen, unter: https://www.planet-wissen.de/technik/verkehr/logistik_waren_unterwegs/pwieschwarmintelligenzvondenfischenlernen100.html, rev. 11.02.2018

The LEGO Group (2016a) Großer EV3 Servomotor, 2016, unter: https://sh-s7-live-s.legocdn.com/is/image/LEGO/45502?$main$, rev. 21.04.2018.

The LEGO Group (2016b) NXT Farbsensor, 2016, unter: http://cdn1.preisroboter.de/detail/6ee2f8557b18c986f5261afd397de763, rev. 21.04.2018.

The LEGO Group (2016c) Tonsensor, 2016, unter: https://images-na.ssl-images-amazon.com/images/I/41TXaSTzUkL.jpg, rev. 21.04.2018.

Thomas Seilnacht (2013) Biene, 2013 unter: http://www.digitalefolien.de/biologie/tiere/insekt/biene/tnricht.JPG, rev. 21.04.2018.

Werfel, J. (2016) Building structures with robot swarms, unter: https://www.oreilly.com/ideas/building-structures-with-robot-swarms, rev. 24.03.2018

Wooldridge, M. (2002) Intelligent Agents: The Key Concepts, Springer-Verlag, 2002, unter: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.5634&rep=rep1&type=pdf, rev. 22.02.2018.

Wooldridge, M. / Jennings, R. N. (o.J.) Intelligent Agents: Theory and Practice, unter: http://www.cs.upc.edu/~jvazquez/teaching/masd/docs/wooldridge95intelligent.pdf, rev. 22.02.2018, S.5

Anhang

A1. Versuch von Dirk Helbing und dessen Ergebnisse

Table 1. The wisdom of crowd effect exists with respect to the geometric mean but not with respect to the arithmetic mean

		Wisdom-of-crowd aggregation		
Question	True value	Arithmetic mean	Geometric mean	Median
1. Population density of Switzerland	184	2,644 (+1,337.2%)	132 (–28.1%)	130 (–29.3%)
2. Border length, Switzerland/Italy	734	1,959 (+166.9%)	338 (–54%)	300 (–59.1%)
3. New immigrants to Zurich	10,067	26,773 (+165.9%)	8,178 (–18.8%)	10,000 (–0.7%)
4. Murders, 2006, Switzerland	198	838 (+323.2%)	174 (–11.9%)	170 (–14.1%)
5. Rapes, 2006, Switzerland	639	1,017 (+59.1%)	285 (–55.4%)	250 (–60.9%)
6. Assaults, 2006, Switzerland	9,272	135,051 (+1,356.5%)	6,039 (–34.9%)	4,000 (–56.9%)

The aggregate measures arithmetic mean, geometric mean, and median are computed on the set of all first estimates regardless of the information condition. Values in parentheses are deviations from the true value as percentages.

Die obige Tabelle zeigt alle im Versuch gestellten Fragen sowie deren Schätzungen im arithmetischen Mittel, geometrischen Mittel und im Median.

B

	(1) groups' wisdom-of-crowd indicator	(2) individuals' increase in confidence
intercept	3.92***	-0.049
	(15.1)	(-0.97)
aggregated information	-1.39**	0.20**
	(-3.29)	(3.08)
full information	-0.98*	0.28***
	(-2.21)	(4.14)
N	288	864

t statistics in parentheses (robust std. errors), * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Die obige Grafik zeigt die Auswirkungen der sozialen Beeinflussung des Experimentes. Hier wurde die Frage gestellt, wie viele Morde in der Schweiz im Jahr 2006 registriert wurden. Des Weiteren wird hier der berechnete Indikator in Bezug auf die Weisheit der

Masse (Werte rot markiert) sowie die Sicherheit der Teilnehmer in ihren Schätzungen (Werte blau markiert) veranschaulicht. Sie unterteilt sich im Groben in 3 Zeilen. Die erste Zeile enthält die Daten der Teilnehmer, die vor ihrer Schätzung keine Informationen als Grundlage hatten. Die zweite Zeile enthält Daten der Gruppe von Teilnehmern, die gesammelte Informationen als Grundlage für ihre Schätzung nutzen konnten. Letzte Gruppe hatte alle Informationen der beiden vorherigen Gruppendurchläufe. Vergleicht man die Zeile „intercept“ mit den beiden anderen Spalten, kann man den Effekt des sozialen Einflusses sehr genau erkennen. Ein Indikator in Bezug auf die Weisheit (rot) der Masse mit dem Wert 0 deutet darauf hin, dass die Wahrheit bzw. der richtige Wert außerhalb der Schätzwerte liegt. Je höher der Indikator für die Sicherheit der Teilnehmer in ihren Schätzungen (blau) ist, desto selbstsicherer sind die Teilnehmer. Die berechneten Indikatoren zeigen sehr genau, dass die Weisheit der Masse sinkt, wenn diese Informationen in ihre Schätzungen einbeziehen dürfen, sprich sozial beeinflusst werden. Je mehr die Teilnehmer beeinflusst werden, desto selbstsicherer sind sie in der Annahme, dass ihre Schätzung zutrifft. Dies zeigt auch, dass der Mensch der Meinung ist: Was die meisten Denken bzw. Schätzen, kann also gar nicht so falsch sein. Außerdem wird klar: Die Vielfalt der Schätzungen führt zu einem besseren Ergebnis.

Dieser Versuch zeigt, dass vermeintliche Schwarmintelligenz in Schwarmdummheit umschlagen kann, wenn soziale Beeinflussung und / oder Unsicherheit der Teilnehmer stattfindet bzw. diese betrifft. Die Teilnehmer fallen dadurch in ein Schwarmverhalten bzw. ein Herdenverhalten, wodurch sie sich an anderen orientieren und das Ergebnis der Teilnehmer im Schwarmverbund wesentlich schlechter wird.

A2. Aufsetzen von LeJOS

Beim Aufsetzen der Umgebung zur Verwendung von Eclipse zur Programmierung der NXT Bausteine müssen folgende Schritte beachtet, bzw. befolgt werden.

1. 32bit JDK herunterladen

Da LeJOS auf einer 32 Bit Umgebung ausgelegt ist, muss dementsprechend eine 32 Bit JDK runtergeladen werden. Eine solche findet man direkt auf ORACLE (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) unter *Windowsx86*.

2. Treiber für NXT Bausteine herunterladen und installieren

Damit man über USB die NXT Bausteine ansprechen kann müssen entsprechende Treiber installiert werden. Für ein 32bit System ein 32 Bit Treiber und für ein 64 Bit System ein 64 Bit Treiber. Allerdings kann man im Zweifelsfall auch beide installieren, da diese nicht miteinander korrelieren. Im Gerätemanager kann man dann schauen, ob diese auch erkannt werden, da es öfter vorkommt, dass für den NXT Arduino Treiber angesprochen werden. In diesem Fall muss man manuell den richtigen Treiber auswählen. Die Treiber findet man unter (https://www.lego.com/de-de/mindstorms/downloads) im Abschnitt *NXT-SOFTWARE HERUNTERLADEN (PC/MAC)* unter *NXT Fandom Driver herunterladen*. Falls zudem schon mit der NXJ Software gearbeitet wurde, sind die Treiber schon installiert.

3. LeJOS herunterladen und installieren

LeJOS lässt sich über folgenden Link herunterladen (http://www.lejos.org/nxj-downloads.php) und schließlich installieren.

4. Eclipse (32 Bit) herunterladen und installieren

Eclipse muss wegen LeJOS zudem in einer 32 Bit Version installiert werden. Die Neon Version bietet sich hierfür an. Zudem muss während der Installation auf die 32

Bit JDK verwiesen werden, welche in Schritt 1 runtergeladen wurden. Eine solche findet man unter http://www.eclipse.org/downloads/packages/release/Neon/3. Dabei eignet sich die *Eclipse IDE for Java Developers.*

5. Lejos Plugin einbinden in Eclipse

In Eclipse unter *Help* und dann *Eclipse Marketplace…* öffnet sich ein Fenster mit Suchleiste. In dieser muss man „lejos“ eingeben und dann das angegebene Plugin installieren. Während der Installation wird ein restart von Eclipse gefordert. Dieser muss durchgeführt werden. Bei einer erfolgreichen Installation findet man in der Menüleiste einen neuen Eintrag namens *leJOS NXJ*. Es ist sinnvoll an diesem Punkt zu überprüfen, ob unter *Window* in *Preferences* im Punkt *leJOS NXJ* bei *NXJ_HOME* der richtige Pfad zu LeJOS, welches in Punkt 3 installiert wurde angegeben ist.

6. Flash NXT

Nun kann der NXT angeschlossen werden. Über *leJOS NXJ* wird *Upload Firmware* aufgerufen. Dabei öffnet sich ein weiteres Fenster, in welchem man auf den Button *Flash leJOS firmware* drücken muss. Sobald dies erfolgreich war kann nun programmiert werden.

Beim Anlegen eines neuen Projektes kann nun auch ein leJOS NXT oder leJOS PC Project zur Programmierung des NXT Bausteins angelegt werden.