Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

Bachelorarbeit

Entwicklung eines computergesteuerten Akteurs für das kooperative Spielen in einem dyadischen Geschicklichkeitsspiel

Melanie Budde

2budde@informatik.uni-hamburg.de
Studiengang Informatik
Matr.-Nr. 6494505

Erstgutachter: Dr. Julian Kunkel
Zweitgutachter: Prof. Dr. Thomas Ludwig

Abgabe: 01.2019

Inhaltsverzeichnis

1. Einleitung **1**
1.1. Motivation 2
1.2. Zielsetzung 2
1.3. Gliederung 3

2. Hintergrund **5**
2.1. Informationen zu der Studie 5
2.2. Aufbau des Ball-Game 7
2.2.1. Analytischer Aufbau des Ball-Game 7
2.2.2. Technische Realisierung des Ball-Game 9
2.3. Benutze Algorithmen, Formeln und Modelle 10
2.3.1. Das Client-Server Modell 10
2.3.2. Newtonsche Gesetze 11
2.3.3. Der A*-Algorithmus 13
2.3.4. Grundlegendes zur linearen Algebra 15

3. Design **17**
3.1. Übersicht der Teilprobleme 17
3.1.1. Die Auswahl der Sprache 17
3.1.2. Die Verbindung zum Zielsystem 18
3.1.3. Steuersignale - Wo wird der Akteur eingespeist? 19
3.1.4. Das Kontext-Bewusstsein des Akteurs 20
3.1.5. Die Entscheidungsprobleme 21
3.2. Die Eigenschaften und das taktische Verhalten des Akteurs 23
3.3. Vorgehen 25

4. Die Implementation des Akteurs **27**
4.1. Der Test-Server, eine Nach Implementation des Ball-Game 27
4.2. Verbindung zum Zielsystem 29
4.3. Die Wegeplanung und Pfaderstellung 32
4.3.1. Kontextinformationen und Sicht auf das Spielfeld 32
4.3.2. Wegefindung mit dem A*-Algorithmus 34
4.3.3. Modellierung der Pfade 35
4.3.4. Die natürliche Bewegung des Balles auf dem Pfad 36

4.4. Pfaderkennung 39
4.5. Die Modellierung der verschiedenen Eigenschaften des Akteurs 41

5. Zusammenfassung **45**

Literaturverzeichnis **47**

A. Anhang **49**
A.1. Benutzte Bibliotheken 49
A.2. Frames 50
A.3. Code Statistik 52

Eidesstattliche Versicherung **63**

1. Einleitung

Der Mensch strebt seit jeher danach sich künstlich abzubilden. Künstliche Menschen wie Shelleys *Frankenstein* oder Hoffmanns *Die Automate* sind in vielfältiger Art und Weise schon seit dem 18. Jahrhundert in der Literatur bekannt. Philosoph und Mathematiker Gottfried Wilhelm Leibniz (1646-1716), träumte davon, alle Begriffe und Aussagen in eine mathematische Sprache übersetzen zu können, um philosophische Fragen einfach durch Ausrechnen der Lösung beantworten zu können. Von ihm stammt der Leitsatz *Denken als Rechnen*. Ebenso legte der Mathematiker Alan Turing mit der Entwicklung der Turing Maschine einen der wichtigsten Grundsteine für die künstliche Intelligenz *KI*. Er definierte den Turing-Test. Dieser besagt ein Computer sei dann intelligent, wenn der Benutzer nicht mehr unterscheiden kann, ob er mit einem Computer kommuniziert der mit einer künstlichen Intelligenz *KI* ausgestattet ist oder mit einem Menschen. Die Anwendungsbereiche in denen die Abbildung menschlichen Verhaltens gefragt ist, sind so vielfältig wie die Informatik und ihre Themenvielfalt selber. Sei es durch Sprachsimulation mittels natürlicher Spracherkennung (natural language prozessing, NLP), die uns mittlerweile komplexe Gespräche mit digitalen Agenten führen lassen, bis hin zu Robotern, die sich bewegen wie wir und selbst Gesichtsausdrücke lesen und deuten können. Auch in Spielen nehmen sie viele Rollen ein. Von Gefährten bis zu Antagonisten, Fortschritte und Popularität der KI steigen, in vielen Teildisziplinen der Informatik.

1.1. Motivation

Das Institut für Neurophysiologie und Pathophysiologie des Universitätsklinikums Eppendorf (UKE), führt unter Leitung von Prof. Dr. med. Andreas K. Engel derzeit eine Studie mit dem Thema „Die Rolle sensomotorischer Kontingenzen bei sozialer Kopplung“ durch. Mit sensomotorischen Kontingenzen ist hier das Zusammenspiel von Sinneswahrnehmung, motorischer Koordination und den dazwischengeschalteten Lernprozessen gemeint. Gegeben meiner aktuellen Sinneseindrücke und Bewegungen, welche weiteren Handlungen bieten sich an, und auf welche Veränderungen in meiner Umwelt sollte ich besonders achten? Diese Studie untersucht insbesondere sensomotorische Kontingenzen die im zwischenmenschlichen Raum eine wichtige Rolle spielen, also dafür, um sich miteinander abzustimmen, gemeinsame Ziele zu entdecken und diese auch zu erreichen. Um einen experimentellen Beitrag zu der Studie zu leisten, wurde Studien-seitens ein digitales Geschicklichkeitsspiel entworfen, bei dem zwei Spieler, gemeinsam einen Ball steuern müssen. Dieses Spiel ist bereits seit Beginn der Studie für die Datenerhebung im Einsatz. Es wird nun ein Programm als Gegenspieler benötigt, welches sich möglichst menschlich verhält, also nicht von einem humanoiden Spieler zu unterscheiden ist. Dieser *Akteur* soll in einer erneuten Datenerhebung einen der zwei Spieler ersetzen. Es soll beobachtet werden, wie die Probanden ihre Partner wahrnehmen und bewerten, und ob sie erkennen, dass es sich um einen Computer handelt.

1.2. Zielsetzung

Im Rahmen dieser Arbeit wurde ein Computer gesteuerter Akteur für das dyadische Geschicklichkeitsspiel entworfen.

Hierbei kam es nicht nur darauf an, ein Programm zu entwickeln, welches mithilfe von Algorithmen aus dem Bereich der KI eigenständig in der Lage ist einen Ball zu navigieren und im Spiele Verlauf dazu zu lernen. Der Fokus wurde auf die Erstellung unterschiedlicher Verhaltensweisen bei der Navigation gelegt. Dabei sind folgende Eigenschaften von Bedeutung:

1. Initiative: Dominanz oder folgen in Bezug auf das Spielverhalten vom Mitspieler.
2. Exploratives oder erfolgsorientiertes Verhalten beim Spiel selber.

Dieser Akteur soll in einer erneuten Verhaltens-Datenerhebung einer Folgestudie zum Einsatz kommen. Probanden werden im Wechsel gegen einen Menschen und gegen den Akteur spielen.

1.3. Gliederung

In Kapitel 1 wird die zugrundeliegende Motivation für diese Arbeit dargelegt, sowie die Zielsetzungen geklärt. Kapitel 2 beinhaltet sämtliches Hintergrundwissen zu der Studie, auf die diese Arbeit aufbaut, zu dem Geschicklichkeitsspiel sowie benutzte Algorithmen und Formeln. Sämtliche Design Entscheidungen und beinhaltete Teilprobleme werden in Kapitel 3 erörtert. Informationen bezüglich der Abbildung des Designs in der Implementation des Akterus, befinden sich in Kapitel 4. Am Schluss werden in Kapitel 5 weitere Möglichkeiten und Alternativen, ergänzend zu dieser Arbeit genannt.

2. Hintergrund

Dieses Kapitel beschreibt relevantes Vorwissen. Im folgenden Abschnitt geht es neben der impliziten und expliziten Herstellung eines dyadischen Verhältnisses zwischen zwei Spielern, um Fragen zum Ablauf der Studie, dem inkludierten Ball-Game, der Umgebung des Akteurs sowie die damit zusammenhängenden technischen Details. Dabei werden in Kapitel 2.1 sämtliche Informationen der zugrunde liegenden Studie dargelegt. In Kapitel 2.2 wird der genauere Versuchsaufbau erläutert, sowie die Spielregeln und gegebene äußere Umstände dargelegt und in Zusammenhang gesetzt. Der Abschnitt 2.3 legt genutzte Modelle und grundlegende Algorithmen sowie Formeln dar, die bei der Implementation des Akteurs zum Einsatz kommen.

2.1. Informationen zu der Studie

Um die in Kapitel 1.1 genannte Studie zu stützen, wurde ein digitales dyadisches Geschicklichkeitsspiel entworfen, im Folgenden *Ball-Game* genannt. Dabei ist eine Zweierbeziehung, auch Dyade bzw. Dyadische Beziehung genannt, in den Sozialwissenschaften eine intensive soziale Beziehung von zwei Personen [Lenz(2009)]. Diese Beziehung wird in dem Ball-Game darüber erreicht, dass ein Ball von zwei Spielern gesteuert wird und diese miteinander kooperieren müssen, um gemeinsame Ziele zu erreichen. Das Forschungsvorhaben untersucht insbesondere sensomotorische Kontingenzen die im Zwischenmenschlichen Raum eine wichtige Rolle spielen, also dafür, um sich miteinander abzustimmen, gemeinsame Ziele zu entdecken und diese auch zu erreichen. Die theoretische Grundlage der durchgeführten Studien beinhaltet neben der Kontext-abhängigen Fähigkeit zur Vorhersage von Handlungen und Handlungskonsequenzen eine umfassende Gleichstellung der verschiedenen Ebenen auf denen menschliches Verhalten organisiert ist – weg von der Idee, dass Intelligenz und Kognition allgemein allein dem Gehirn entspringen. Konkret wird die Annahme getestet, inwiefern sich komplexe soziale Fähigkeiten, wie auch das subjektive Erleben einer Interaktion, auf der Ebene einfacher Bewegungen abspielen. Die Untersuchungen widmen sich konkret der Frage, ob das Erlernen und Beherrschen dieser SMCs im Rahmen eines zu zweit zu absolvierenden, digitalen Geschicklichkeitsspiels gleichermaßen zu einer veränderten sozialen Interaktion zwischen den Spielern führt. Der Frage nach dem Zusammenhang von sensomotorischer und sozialer Kopplung wird durch die Analyse von verschiedenen Daten nachgegangen, unter anderem: Bewegungs- und hirnphysiologische Parameter, Fragebogen-Daten, Probanden-Ratings zum subjektiv empfundenen Involviert-Sein, „Wir“-Gefühl und Qualität der Interaktion zwischen den

Probanden. Dabei wird die neuronale Aktivität des Gehirns durch Elektroenzephalografie gemessen. Zusätzlich wird die Rolle sozial relevanter SMCs bei Patienten mit Autismus-Spektrum-Störung (*ASS*) erörtert, die massive qualitative Beeinträchtigungen in sozialer Interaktion aufweisen.
Das hier beschriebene Vorhaben umfasst zwei Studien, die in drei bzw. zwei Teilexperimente unterteilt sind. Mit den laufenden Studien sollen folgende übergreifende Hypothesen getestet werden:

1. Das Erlernen und Beherrschen von SMCs in einem dyadischen Geschicklichkeitsspiel ist assoziiert mit dem sozialen Interaktionsmuster zwischen den beiden Spielern.

2. Soziale Kopplung bzw. Ratings zum subjektiven Involviert-Sein und der Qualität der Interaktion spiegeln sich wider in neuronaler Kopplung und in Korrelationen zwischen den erhobenen Bewegungsparametern der beiden Akteure.

3. Sozial-kognitive Defizite bei Patienten mit ASS sind assoziiert mit Defiziten im Erwerb und der Beherrschung von SMCs.

Primärer Endpunkt in beiden geplanten Studien ist eine Quantifizierung des Zusammenhangs zwischen sensomotorischer und sozialer Kopplung in dyadischer Interaktion sowie eine Darstellung der neuronalen Korrelate kooperativen Verhaltens. Sekundärer Endpunkt in beiden Studien ist eine Beschreibung dieses Zusammenhangs beim pathologischen Fall der ASS, sowie die Formulierung formaler Modelle zur Rolle von SMCs bei ASS-charakteristischen sozialen Defiziten. Sämtliche Details über den Studienaufbau und Verlauf kann dem Studienprotokoll aus dem Anhang entnommen werden.

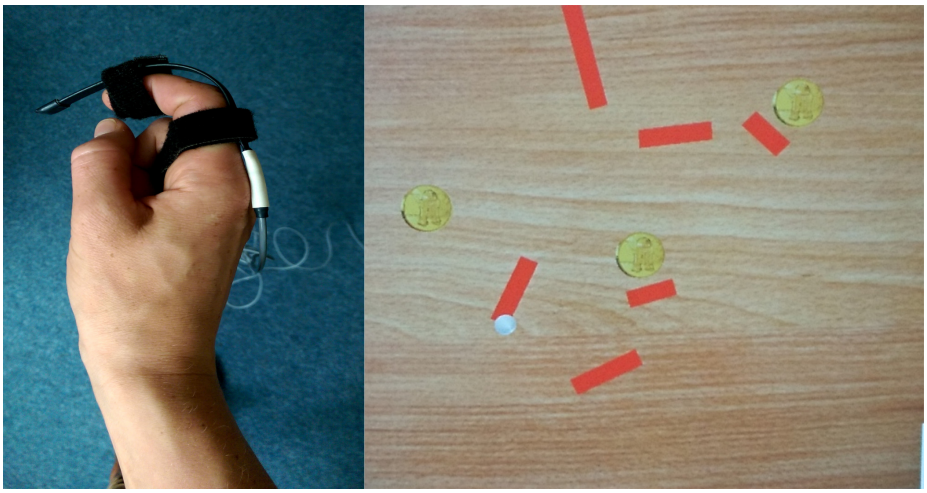

Abbildung 2.1.: Bimetall für die Steuerung *(links)*, Spielfeld *(rechts)*

2.2. Aufbau des Ball-Game

2.2.1. Analytischer Aufbau des Ball-Game

Bei dem Ball-Game spielen zwei Spieler gegeneinander ein 3-D Geschicklichkeitsspiel am Computer, wobei sie in zwei verschiedenen Räumen und somit auch an verschiedenen Bildschirmen sitzen. Dadurch fällt sämtlicher Kontakt und die Möglichkeit zur weiteren, verbalen Kommunikation während des Spiels weg. Die Größe des Spielfelds entspricht der Auflösung 1024×768 px. Das Ziel des Spieles ist es, mit einem Ball eine von drei Münzen einzusammeln die an vier Stellen auf dem Spielfeld erscheinen. Dabei wird der Ball durch beide Spieler gemeinsam gesteuert. Die Steuerung des Balls erfolgt durch Bimetalle, welche jeweils um beide Zeigefinger gelegt werden (siehe Abbildung 2.1). Der linke Zeigefinger von Spieler 1 steuert dabei die y-Koordination, während der rechte Zeigefinger die x-Koordination steuert (bei Spieler 2 invers). Die Bimetalle wurden gewählt, um kontinuierliche Bewegungen abzubilden, Muskelaktivität möglichst gering zu halten, und Chancengleichheit zwischen den Spielern zu wahren. Da dies eine nicht-native Steuerungsbewegung darstellt haben die Spieler, die z. B. viel an einer Spielekonsole gespielt haben, keinen Vorteil. Der Ball unterliegt den gängigen physikalischen Gesetzen und besitzt Trägheit sowie Masse. Wird eine Münze vom Ball berührt, erscheint an der selben Stelle eine neue, sobald eine weitere Münze eingesammelt wurde. Es gibt neun Hindernisse von denen nur sechs für jeden Spieler sichtbar sind. Die Verteilung der Hindernisse ist bei beiden Spielern gleich, jedoch sind nicht alle für beide gleich sichtbar. Außerdem sind die Hindernisse nicht blockierend, sondern mit einem Bremsfaktor versehen. Ein Spiel besteht aus mehreren Runden die jeweils 1 Minute dauern. Die Objekte in dem Spielfeld werden per Zufall vor jeder Runde anders platziert.

Derzeit genutzte Versionen
NI LabVIEW 2017 (32-bit) Windows 10 (64-bit)
Abhängigkeiten
LabVIEW Robotics Module 2017 ZeroMQ Socket Library v3.4.1.107 MATLAB R2017b

Tabelle 2.1.: Details zu LabVIEW.

Das Ball-Game erfüllt folgende Ansprüche:

1. Beide Spieler sollen in ihrem Verhalten möglichst autonom bleiben. Deswegen gibt es während des Spiels immer drei Münzen die als nächstes eingesammelt werden könnten. Hierbei wird redundante Kontrolle dadurch erreicht, dass beide Spieler jeweils die Bewegung in x- und y-Richtung steuern.

2. Es soll einen Grund zum Austausch geben. Deshalb sind einige Bremsregionen nur für den einen bzw. den anderen Spieler sichtbar. Die Spieler sollten vermeiden in diese Regionen zu fahren und müssen sich dafür auf das Gegenüber einlassen.

2.2.2. Technische Realisierung des Ball-Game

Realisiert ist das Ball-Game mittels LabVIEW. LabVIEW ist eine *system-design*-Plattform und Entwicklungsumgebung für visuelles Programmieren und kann auf dem Betriebssystem Windows ausgeführt werden. Details zu der genutzten Version und den benötigten Abhängigkeiten sind in Tabelle 2.1 aufgeführt. Hierbei fungiert LabVIEW als Server und vermittelt sämtliche Kontextinformationen, die die jeweiligen Runden definieren wie z.B. Position des Balls, Koordinaten, Größe und Lage der Hindernisse und Münzen an seine Clients (in diesem Fall die zwei Spieler, siehe Kapitel 2.3.1). Da einer der Spieler simuliert werden muss, ist eben diese Schnittstelle und die Informationen die sie vermittelt für die Implementation des Akteurs von Bedeutung. Dort stehen alle Informationen aus denen sich der Akteur sein Kontext-Bewustsein zieht um seine Handlungsoptionen zu planen. Es gibt vier verschiedene Frame Typen die vom Server mit ca. 35 FPS gesendet werden:

- *Start*:
  Legt das Protokoll fest, wird nur ganz am Anfang einmal ausgegeben.
- *Stop*:
  Pausiert das Spiel serverseitig.
- *End*:
  Beendet den aktuellen Versuch.
- *Experiment Definition*:
  Definiert wie viele Versuche es im gesamten Spiel gibt, wird ebenfalls nur einmal am Anfang ausgegeben.
- *Versuch Definition*:
  Gibt die Verteilung der Hindernisse und Münzen, sowie Startposition und Dauer pro Versuch an.
- *Aktualisierung*:
  Übermittelt sämtliche Informationen zum Spielfluss, wie aktuelle Ballposition und aktuelle Krafteinwirkungen auf den Ball.

Der Aufbau der einzelnen Frames kann dem Anhang entnommen werden (A.2), in dem exemplarische Frames von jedem Typ enthalten sind.

2.3. Benutze Algorithmen, Formeln und Modelle

2.3.1. Das Client-Server Modell

Die Kommunikation zwischen dem Ball-Game und dem Akteur ist als Client-Server Modell realisiert. Das Modell umschreibt die Möglichkeit, Aufgaben und Dienstleistungen innerhalb eines Netzwerks zu verteilen. Programme werden in Clients und Server unterteilt und sind für die Ausführung zuständig. Ein Client, als aktiver Part der Verbindung, kann ggf. einen Dienst vom Server anfordern, beispielsweise ein Betriebsmittel, der Server der sich auf einem beliebigen Rechner im Netzwerk befindet, beantwortet die Anfrage indem er z. B. ein Betriebsmittel bereitstellt (siehe Abbildung 3.1). Er muss sich ständig in einem Zustand befinden, in dem er Verbindungsaufforderungen von Clients entgegennehmen kann und ist somit der passive Teil.[Roland Bauch(2004)]. Um untereinander kommunizieren zu können wird eine Schnittstelle zwischen einem Prozess und einem gemeinsamen Transportprotokoll benötigt. In der dieser Arbeit zugrunde liegenden Implementation, nimmt das Ball-Game die Rolle des Servers, und der Akteur die des Clienten ein. Als Transportprotokoll wird TCP verwendet und um die vom Server gesendeten Daten vom Typ `cluster` einer vernünftigen Typumwandlung unterziehen zu können, werden die Daten als SON [Smith(2015)] formatiert und mittels TCP Sockets als Schnittstelle vermittelt [Xue and Zhu(2009)].
In Listing A.3 ist die Implementation des Sockets auf Clientseite dargestellt.

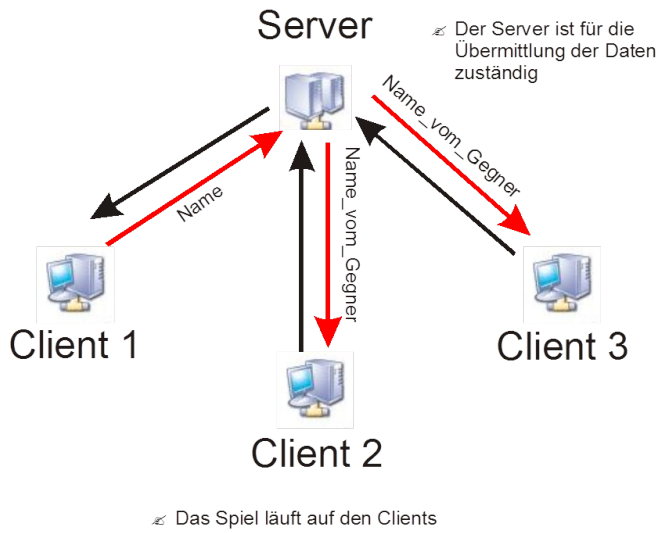

Abbildung 2.2.: Client-Server Model. Abbildung entnommen von [Blog(2015)]

2.3.2. Newtonsche Gesetze

Von den folgenden Newton'schen Axiomen wurde gebrauch gemacht: [Block(2010)].

- Trägheitsprinzip:
  Ein Körper bleibt so lange im Ruhezustand oder dem Zustand der gleichförmig geradlinigen Bewegung, bis er durch einwirkende Kräfte zur Änderung seines Zustands gezwungen wird. Wirkt also eine Kraft auf einen Massepunkt, so führt er eine gleichförmige Bewegung aus.

- Aktionsprinzip:
  Das Aktionsprinzip macht eine quantitative Aussage darüber, wie sich die Bewegung eines Massepunktes verändert, wenn eine Kraft auf ihn wirkt. Die Änderung der Bewegung ist proportional zu der Einwirkung der bewegenden Kraft und richtet sich nach der geraden Linie, auf der diese Kraft wirkt.

- Superpositionsprinzip der Kräfte:
  Wirken mehrere Kräfte auf einen Massepunkt so addieren sich diese vektoriell zu einer resultierenden Kraft auf.

Eine Kraft F wird gewöhnlich in Newton gemessen, eine Beschleunigung a in m/s^2, eine Geschwindigkeit v in m/s und eine Masse m in kg. Im Ballgame wird die Kraft, die auf die simulierte Masse des Balls wirkt, aus dem Eingabewert des Bimetalls abgeleitet. Bei der Simulation wird als Streckeneinheit *Pixel* und als Zeiteinheit *Frames* verwendet. Nach den Newtonschen Gesetzen gilt für die Kraft F:

$$F = m \cdot a \tag{2.1}$$

und somit für die Beschleunigung:

$$a = F/m \tag{2.2}$$

Bei Masse m und Kraft F wird ein Objekt um a beschleunigt. Als Einheit für die Geschwindigkeit ergibt sich somit $Pixel/Frame$ und für die Beschleunigung $Pixel/Frame^2$:

$$v = \frac{F}{m} \cdot t = a \cdot t \tag{2.3}$$

Zur Simulation der Ballbewegung wird die Formel für gleichmäßig beschleunigte Bewegungen herangezogen:

$$v(t) = v_0 + a \cdot t \tag{2.4}$$

Die Formel in dieser Form ist eine gleichmäßige Beschleunigung [Wikipedia(2018b)]. Mit Hilfe dieser Formel wird ersichtlich, dass der durch das Bimetall gemessene Wert über die

Kraft F eine Veränderung der Geschwindigkeit des Balles bewirkt.

Algorithm 1 A*-Algorithmus

```
open ← start
closed ← ∅
while open ≠ ∅ do
    sort(open)
    x ← open.pop()
    kids ← expand(x)
    for all kid ∈ kids do
        kid.f ← min{kid.f, (x.g + 1) + h(kid, goal)}
        if goal = kid then
            return makePath(kid)
        end if
        if kid ∩ closed = ∅ then
            open ← kid
        end if
    end for
    closed ← x
end while
return ∅
```

2.3.3. Der A*-Algorithmus

In Algorithmus 1 ist ein Pseudocode1 des A*-Algorithmus dargestellt.

Dieser Algorithmus findet einen optimalen Weg von A nach B und berücksichtigt dabei etwaige Hindernisse die im Weg liegen könnten. Dabei werden für jeden Knoten die Kosten abgeschätzt, die benötigt werden um von dem Startknoten A über den jeweiligen Knoten X zu dem Zielknoten B zu gelangen. Für diese Abschätzung werden die minimalen Kosten aller Nachbarknoten X_j des Knotens mit den bisher geringsten Kosten Y über folgende Formel berechnet:

$$f(X_j) = g(Y) + 1 + H(X_j, B) \tag{2.5}$$

Dabei gibt $f(X_j)$ die abgeschätzten Kosten des bisher besten Pfades vom Start- zum Zielknoten über den Knoten Y und X_j an, $g(Y)$ gibt die Kosten vom Startknoten zu dem Knoten Y an, 1 sind in dem hier betrachteten Fall die Kosten um von Knoten Y zu Knoten X_j zu gelangen und $H(X_j, B)$ ist eine heuristische Abschätzung der Kosten von dem Knoten X_j zum Zielknoten B. Als Heuristik wird hier die Manhattan Distanz (L1-Norm) verwendet.

$$H(X, B) = d(X, B) = \sum_i |x_i - b_i| \tag{2.6}$$

Wichtig hierbei ist, dass die Heuristik eine untere Schranke bildet, also die theoretisch minimalen Kosten zur Erreichung des Zielknotens verwendet, da die Suche sonst in die Irre geleitet werden kann. Bei jedem Schritt wird der Knoten expandiert (um seine Nachbarknoten erweitert), dessen Kosten f am geringsten sind. Da für den hier beschriebenen An-

wendungsfall neben den Kosten, auch der Pfad selbst gespeichert werden soll, muss der Vorgänger eines jeden Knotens gespeichert werden. Der A*-Algorithmus in Algorithmus 1 bekommt als Parameter den Startknoten *start* und den Zielknoten *goal*. Zuerst werden in den Zeilen 1 und 2 die Listen für die noch abzuarbeitenden (*open*) und schon abgearbeiteten Knoten (*closed*) initialisiert, wobei der Startknoten direkt in die Menge der noch abzuarbeitenden Knoten hinzugefügt wird. In der Schleife in Zeile drei werden folgende Schritte solange ausgeführt, wie der Zielknoten nicht erreicht ist und noch abzuarbeitende Knoten vorhanden sind:

- betrachte den nächsten noch unbearbeiteten Knoten x mit den bisher geringsten Kosten f (Zeile 5)
- berechne alle Nachbarknoten (*kids*) des aktuell betrachteten Knoten x (Zeile sechs)
- für jeden Kindsknoten *kid* werden wiederum folgende Schritte ausgeführt:
  - die abgeschätzten Kosten vom Startknoten über Knoten x und Knoten *kid* zum Zielknoten werden nach Formel 2.5 berechnet (Zeile 8). Hierbei ist zu beachten, dass jeder Knoten immer nur die bisher geringsten Kosten speichert. Wird der Knoten später wieder betrachtet und die abgeschätzten Kosten sind höher, werden diese verworfen.
  - wenn der Kindsknoten *kid* der Zielknoten ist, wird die Suche abgebrochen und der Pfad vom Startknoten zum Zielknoten rekonstruiert (Zeile 9 bis 11). Die Rekonstruktion des Pfades ist nicht in dem Pseudocode mit dargestellt.
  - wenn der Kindsknoten *kid* das erste Mal betrachtet wurde, wird er zu der Liste der noch abzuarbeitenden Knoten (*open*) hinzugefügt (Zeile 12 bis 14)
- der Knoten x wird als abgearbeitet markiert (Zeile 16)

Es werden somit zuerst die Knoten untersucht die wahrscheinlich am schnellsten zum Ziel führen. Dafür speichert sich jeder Knoten die geringsten Kosten auf dem Weg vom Start- über den jeweiligen Knoten zum Zielknoten. Als initialer Wert wird unendlich genommen, damit bei der ersten Betrachtung jedes Knotens die Kosten abgeschätzt werden.

2.3.4. Grundlegendes zur linearen Algebra

Interpolation

Für einen Satz diskreter Werte kann die lineare Interpolation andere Werte dazwischen approximieren, wobei eine lineare Entwicklung zwischen diesen diskreten Werten angenommen wird. Es werden zwei Datenpunkte (x_0, f_0) und (x_1, f_1) durch eine Strecke verbunden. Es gilt:

$$f(x) = f_0 + \frac{f_1 - f_0}{x_1 - x\breve{0}}(x - x_0) = f_0 \frac{x_1 - x}{x_1 - x\breve{0}} + f_1 \frac{x - x_0}{x_1 - x\breve{0}} \tag{2.7}$$

Somit entsteht eine Konvexkombination der Endpunkte (x_0, f_0) und (x_1, f_1). [Wikipedia(2014)]

Rotationsmatrix

Eine Rotationsmatrix ist eine Matrix, die zum Ausführen einer Rotation im euklidischen Raum verwendet wird. Zum Beispiel um einen Vektor zu drehen.

$$R = \begin{pmatrix} \cos_\alpha & -\sin_\alpha \\ \sin_\alpha & \cos_\alpha \end{pmatrix} \tag{2.8}$$

Dabei werden die Punkte auf der xy-Ebende gegen den Uhrzeigersinn um einen beliebigen Winkel um den Ursprung des kartesischen Koordinatensystems gedreht. Um die Rotation unter Verwendung einer Rotationsmatrix R durchzuführen, muss die Position jedes Punktes durch einen Spaltenvektor v dargestellt werden, der die Koordinaten des Punktes enthält. [Wikipedia(2018a)]

Vektornormalisierung

Ein Einheitsvektor in einem normierten Vektorraum ist ein Vektor der Länge 1. Solch ein Vektor, der zur Darstellung der räumlichen Richtung verwendet wird, wird Richtungsvektor genannt. Ein auf diese Weise dargestellter 2D-Richtungsvektor entspricht numerischen Punkten auf dem Einheitskreis. Ein vom Nullvektor verschiedener Vektor $\vec{v}$ kann normiert werden, indem man ihn durch seinen Betrag dividiert:

$$\vec{n} = \frac{\vec{v}}{||\vec{v}||} \tag{2.9}$$

Der daraus resultierende Vektor ist der Einheitsvektor, er zeigt in dieselbe Richtung wie $\vec{v}$. [Wikipedia(2018c)]

3. Design

Nachdem der Hintergrund der Studie, in die der Akteur eingebettet ist, sowie der genauere Versuchsaufbau in Abschnitt 2.2 durchleuchtet wurde, wird sich nun Gedanken zu der Umsetzung gemacht. Im folgenden Kapitel 3 geht es um das Design des Akteurs und die damit zusammenhängenden Frage- und Problemstellungen. Die dabei auftretenden Teilprobleme werden vorbereitend in Abschnitt 3.1 aufgezeigt und diskutiert, bevor im Folgeabschnitt 3.2 primär die Vorgaben zu dem Verhalten des Akteurs, sowie erste Gedanken zu der Modellierung dieser Vorgaben dargelegt werden. In Abschnitt 3.3 wird dann erweitert auf das Vorgehen bei der Implementation eingegangen.

3.1. Übersicht der Teilprobleme

3.1.1. Die Auswahl der Sprache

Eine der ersten Fragen die sich bei der Implementation des Akteurs stellen, ist in welcher Sprache dies geschehen soll. Es gibt zwei Möglichkeiten.

Ansatz 1: die Implementation in LabVIEW oder
Ansatz 2: in einer beliebigen anderen Sprache.

Ein großer Vorteil von LabVIEW ist die Einfachheit, mit der parallele Abläufe programmiert werden können. Es ist ausreichend zwei Elemente ohne Datenabhängigkeit nebeneinander zu legen, um sie zeitgleich mit Multithreading abzuarbeiten. Es muss allerdings, ähnlich wie in textbasierten Programmiersystemen, auf mögliche Race Conditions geachtet werden. Überall da wo es nötig ist, müssen Ressourcen gesperrt werden (z. B. durch Semaphoren). Darüber hinaus erhöht die grafische Darstellung des Programmflusses die Lesbarkeit deutlich. Die Nachteile sind jedoch, dass LabVIEW-Programme sich nur mit der originalen LabVIEW-Entwicklungsumgebung bearbeiten lassen, an deren Funktionsumfang man gebunden ist. Ausserdem können kleine Änderungen aufwendige Neustrukturierungen nach sich ziehen, da nach Verschiebung der Blockdiagramme die Drähte und Symbole oftmals neu geordnet werden müssen, um die Übersichtlichkeit wieder zu erlangen [Wikipedia(2019)]. Außerdem ist es für Studierende nicht immer leicht, an Lizenzen für die Plattform zu kommen da diese von den Universitäten zum Teil nur begrenzt zur Verfügung stehen und teuer sind. Die Vorteile von LabVIEW sind für das dieser Arbeit zugrunde liegende Ziel nichtig, da es keinen Anspruch an parallele Programmierung gibt und die aufwendige Neustrukturierung bei kleineren Änderungen als zu aufwendig für

den Entwicklungsprozess empfunden wird.
Aufgrund des freien Zugangs, der Wiederverwendbarkeit des Codes und der Vertrautheit mit Python, dessen zahlreichen OpenSource Bibliotheken und der guten Lesbarkeit sowie dem hier dargelegten, spezifischen Anwendungsfall (siehe Tabelle 3.1), wurde sich für die Entwicklung einer Hybridlösung entschieden. Dies bringt jedoch einen zusätzlichen Aufwand mit sich, da explizit eine Verbindung zum Austausch der Daten zwischen LabVIEW und dem Pythonprogramm hergestellt werden muss.

	Zeit	Vertrautheit	Lizenzen	Wiederverwendbarkeit
Ansatz 1	+ -	+ +	+ +	+ -
Ansatz 2	- +	- -	- +	- +

Tabelle 3.1.: Gegenüberstellung der Vor- und Nachteile von LabVIEW gegenüber Python

3.1.2. Die Verbindung zum Zielsystem

Die Kommunikation zwischen dem Ball-Game und dem Akteur ist als Client-Server Modell realisiert (siehe Kapitel 2.3.1). Um untereinander kommunizieren zu können wird Interprozesskommunikation benötigt, weil das Ball-Game und der Akteur verschiedene Prozesse sind und miteinander kommunizieren müssen. Hierfür wird in der Praxis häufig TCP/IP verwendet, welches auf der Transportschicht des OSI-Models liegt. Dabei besteht der Vorteil, dass das TCP/IP-Protokoll der Anwendung erlaubt, sowohl vollständig auf einem einzigen physikalischen Rechner ausgeführt zu werden, als auch auf verschiedenen Rechnern – oder gar virtuellen Maschinen, die über ein Netzwerk miteinander kommunizieren. TCP ist ein bidirektionales Transportprotokoll, welches sich im Gegensatz zu UDP darum kümmert, dass jedes Paket ankommt – auch in der richtigen Reihenfolge und ohne fehlende Vorgänger. Für das Ball-Game bedeutet dies, dass auch eine nicht ganz stabile Verbindung zwischen Client und Server keine Probleme bereitet. Man kann sich also darauf verlassen, dass auf dem Kanal alles abgesendete auch am jeweils anderen Ende ankommt. Des Weiteren spielen Latenzzeiten wie z.B. im Bereich MMOs (massive-multi-player online game) keine Rolle, was ebenfalls die Vorteile von UDP eliminiert. In der dieser Arbeit zugrunde liegenden Implementation, nimmt das Ball-Game die Rolle des Servers, und der Akteur die des Clients ein. Um die vom Server gesendeten Daten vom Typ `cluster` einer vernünftigen Typumwandlung unterziehen zu können, werden die Daten serialisiert, also von strukturierten Daten auf eine sequenzielle Darstellungsform gebracht. Als Format wird hierfür JSON verwendet und das Ganze wird mittels TCP Sockets vermittelt. Die Darstellungsform JSON kann am anderen Ende dann wieder in die lokalen Datentypen eingelesen werden [Mandl et al.(2010)Mandl, Bakomenko, and Weiß]

3.1.3. Steuersignale - Wo wird der Akteur eingespeist?

Für diese Arbeit sind vor allem die Bimetalle für die Steuerung des Balls interessant, da der Akteur die von ihnen erzeugte Eingabe simulieren muss. Die Bimetalle geben die Beugung der Finger in Form von Spannung an die in LabVIEW enthaltene *Physik-Engine* weiter. Diese berechnet daraus die folgenden Werte:

1. unverarbeitete Eingabespannung der Bimetalle:
   Dies ist der Rohwert von 0-5V Spannung der von den Bimetallen erzeugt wird. Je größer der Fingerausschlag, desto höher ist die Spannung.

2. kalibrierte Eingabespannung der Bimetalle:
   Für die Kalibrierung werden initial die Fingerbewegungen der Spieler in einem kurzen Zeitraum aufgezeichnet. Davon werden 90% der Spannung des Vollausschlags in beide Richtungen als Maximalwerte für die Steuerung übernommen. Dieser Vorgang wird wiederholt, bis empirisch/subjektiv für die Spieler eine gute Steuerungsumsetzung durch die Fingerbewegungen vorliegt.

3. normalisierte Eingabespannung der Bimetalle:
   Die Eingabespannung der Bimetalle wird zusätzlich auf den Bereich $[0, 1]$ normalisiert.

Die Verarbeitungskette der Eingabespannung der Bimetalle ist in Abbildung 3.1 dargestellt. Die Kurve in dem Diagramm beschreibt dabei die unverarbeitete Eingabespannung der Bimetalle, auch mit s bezeichnet. Die kalibrierte Eingabespannung der Bimetalle wird durch die Funktion $f(s)$ abgebildet. Das Vorgehen zur Kalibrierung ist im vorherigen Abschnitt genauer beschrieben. Durch die Normalisierung der kalibrierten Eingabespannung der Bimetalle, dargestellt durch die Funktion $g(s)$, wird der Kraftvektor berechnet, der auf den Ball im Ball-Game einwirkt. Der Akteur muss die Eingabespannung der Bimetalle simulieren, um das Ball-Game spielen zu können. Dabei kann der Akteur direkt den in Punkt 3 dargestellten kalibrierten und normalisierten Kraftvektor (2.9) in das Spiel eingeben, da dieser Wert in den Spiel-Frames (A.2) direkt mitgegeben wird. Somit kann die unverarbeitete Eingabespannung und die kalibrierte Eingabespannung der Bimetalle vernachlässigt werden und unnötige Berechnungen werden vermieden.

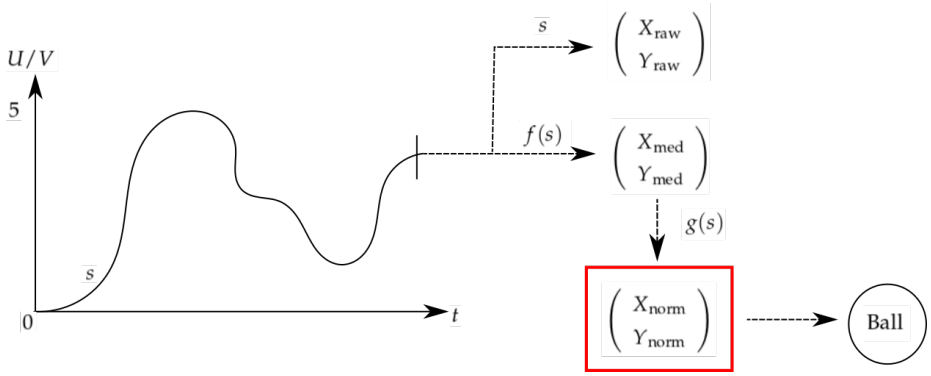

Abbildung 3.1.: Verarbeitung der Eingabespannung der Bimetalle

3.1.4. Das Kontext-Bewusstsein des Akteurs

Nachdem alle anfänglichen Fragen in Bezug auf die Vorbereitungen dargelegt wurden, kann sich nun einem der Kernteile dieser Arbeit gewidmet werden. Um Entscheidungen treffen zu können braucht der Akteur zwingend das Bewusstsein darüber, in welchem Kontext er sich bewegt. Eine Grundannahme in den Neurowissenschaften ist es, dass das Bewusstsein ein Produkt unseres Gehirns ist. Nach Jürgen Schmidhuber ist das Bewusstsein nur ein Nebenprodukt des Problemlösens des Gehirns. So sei auch bei künstlichen Problemlösern von Vorteil, wenn diese sich ihrer selbst und ihrer Umgebung 'bewusst' seien [Schmidhuber(2016)]. Ein intelligenter Akteur muss über seine Möglichkeiten und Fähigkeiten Bescheid wissen, um abzuschätzen was er als nächstes tun kann. In dem hier vorliegendem Fall hat der Akteur ein zustandsbasiertes Modell seiner Umgebung mit einem Start, Ziel und keinerlei Unsicherheit bei der Entscheidung über die als nächstes durchzuführende Handlung. Keinerlei Unsicherheit ist hierbei, wenn der Akteur entscheiden muss, welche Maßnahme er ergreifen soll, obwohl er die Ergebnisse seiner Maßnahmen nicht genau vorhersagen kann. In dem hier vorliegenden Fall ist das Modell seiner Umwelt, eine flache, nicht hierarchische Repräsentation. Es gibt bekannte Start- und Zielpositionen, sowie einen bekannten Raum der durchforstet wird. Darüber hinaus besitzt der Akteur Selbstbewusstsein, er kennt seine Größe, Position und weiß somit z. B. wann er mit einem Hindernis oder Ziel kollidiert[Poole and Mackworth(2010)]. Ein in solch ein Bewusstsein eingebetteter Akteur hat folgende Eigenschaften (siehe auch Abbildung 3.2).

- Bestimmte Fähigkeiten, also primitive Handlungen die er ausführen kann.
- Ziele die er zu erreichen versucht.
- Frühere Erfahrungen mit früheren Handlungen und Beobachtungen oder anderen Daten, aus denen er lernen kann.

Aus der Interaktion mit seiner Umwelt kann der Akteur ebenfalls eine gewisse Vorerfahrung ableiten und Beobachtungen anstellen, die ihm bei der Entscheidungsfindung darüber, was er als nächstes tun wird, helfen.

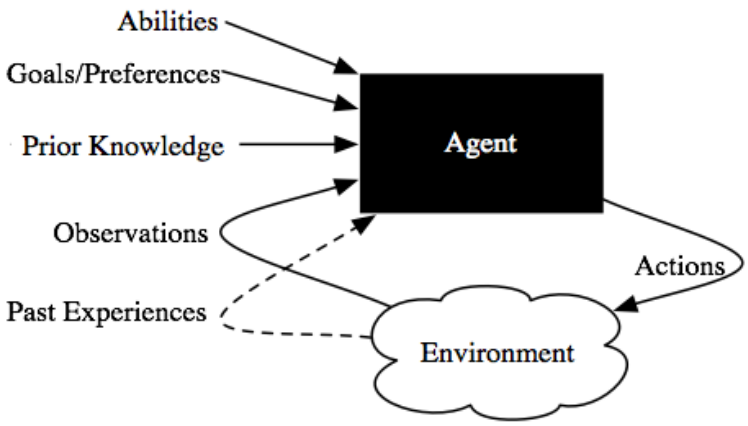

Abbildung 3.2.: Ein mit seiner Umwelt interagierender Akteur. Abbildung entnommen aus [Poole and Mackworth(2010)]

3.1.5. Die Entscheidungsprobleme

Es gibt für den Akteur im Grunde genommen zwei große Entscheidungsprobleme: „Wo“ möchte ich hin und „Wie“ kann ich das erreichen. Erstere Frage ist das Hauptproblem dieser Arbeit. Denn die unterschiedlichen Verhaltensweisen des Akteurs spiegeln sich hauptsächlich darin wider, dass er sich je nach Typ für andere Ziele entscheidet und weniger von sensorischen Faktoren abhängt. Die Physik-Engine des Ball-Game und die Art und Weise wie mit den Bimetallen gesteuert werden kann, vereinfacht es, menschliches Verhalten zu simulieren. Die Steuerung ist sehr ungenau und da der Ball sowohl Masse als auch Trägheit besitzt, werden leichtere Bewegungen, wie z. B. ein Zittern in den Händen oder abrupte Beschleunigungen nicht direkt abgebildet und können somit vernachlässigt werden. Des Weiteren sind für die Folgestudie andere Verhaltensfaktoren von Bedeutung (siehe Kapitel 1.2). Mit dem „Wie“ ist die Entscheidung darüber gemeint, was der für ihn erfolgversprechende Pfad ist um sein Ziel zu erreichen. Diese Problemstellung ist gegenüber dem „Wo“ Problem etwas trivialer. Da die Entscheidung darüber wo der Akteur als nächstes hin will zum Teil von dem Pfad abhängt, der ihn an sein Ziel führt, sollte als erstes die Wegfindung des Akteurs betrachtet werden.

Die Wegefindung

Um einen geeigneten Pfad von seiner aktuellen Position zu seinem ausgewählten Ziel zu bekommen, muss der Zustandsraum genauer betrachtet werden. Ein Zustandsraum wäre das Spielfeld und ein Zustand enthält alle Informationen, die erforderlich sind, um die Auswirkungen einer Aktion vorherzusagen und um zu bestimmen, ob ein Zustand das Ziel erfüllt. Die Suche in dem Zustandsraum setzt voraus, dass der Akteur perfektes Wissen über den Zustandsraum hat. D. h. er kennt alle für ihn sichtbaren Hindernisse und möglichen Ziele.

Ein Zustandsraumproblem besteht aus

- einer Reihe von Zuständen,
- einem unterscheidbaren Zustand, der als Startzustand bezeichnet wird.
- Für jeden Zustand gibt es eine Reihe von Aktionen, die dem Agenten zur Verfügung stehen.
- Eine Funktion, die bei einem Zustand und einer Aktion einen neuen Zustand zurück gibt, sowie
- ein Ziel, das als boolesche Funktion angegeben ist.

Das ist wahr, wenn Zustand s das Ziel erfüllt. In diesem Fall sagen wir s ist ein Zielzustand. Der Akteur kann abschätzen wie er sein Ziel erreicht, indem er in der ihm bekannten Repräsentation seiner Umgebung nach einer Möglichkeit sucht, von seinem aktuellen Zustand in seinen Zielzustand zu gelangen. Dieses Problem kann in das mathematische Problem, einen Pfad von A nach B in einem gerichteten Graphen zu finden, abstrahiert werden. Die Schwierigkeit der Suche und die Tatsache, dass Menschen einige Suchprobleme effizient lösen können, legen nahe, dass Computerakteure das Wissen über Spezialfälle nutzen sollten, um sie zu einer Lösung zu führen. Dieses zusätzliche Wissen jenseits des Suchraums wird als heuristisches Wissen bezeichnet.[Poole and Mackworth(2010)] Der Akteur kann abschätzen wie er sein Ziel erreicht, indem er in der ihm bekannten Repräsentation seiner Umgebung nach einer Möglichkeit sucht, von seinem aktuellen Zustand in seinen Zielzustand zu gelangen.
Der A*Algorithmus ist ein informiertes Suchverfahren und dient der Ermittlung des kostengünstigsten Pfades zwischen zwei Knoten innerhalb eines Suchraums und ist dabei in der Lage, sämtliche Hindernisse zu umgehen und dabei eine Heuristik zu benutzen (siehe Kapitel 2.3.3). Somit kann der Akteur zu einem vorgegebenen Ziel navigieren.

Die Entscheidung für ein Ziel

Die Entscheidung für ein Ziel ist fundamental für die Modellierung der unterschiedlichen Verhaltensweisen. Ein Spieler dem es wichtig ist möglichst viele Punkte zu sammeln, wird sich eher für ein Ziel entscheiden welches sich gut erreichen lässt und verifiziert keine unsichtbaren Hindernisse im Weg hat. Dabei ist nicht immer der kürzeste Weg der Beste. Denn aufgrund der physikalischen Eigenschaften des Balls, durch Trägheit und Masse, ist es nicht einfach den Ball feinmotorisch zu navigieren. Wenn man unbedingt eine Kollision verhindern möchte, bietet es sich nicht an, durch viele nah bei einander liegende Hindernisse zu fahren. Darüber hinaus kann ein Weg, der wenig bis keine Hindernisse beinhaltet, auch gefährlich sein, da sich hier mit erhöhter Wahrscheinlichkeit ein unsichtbares Hindernis befindet. Im Gegensatz hierzu gibt es Spieler die neugieriger sind und das

ganze Spielfeld erkunden wollen. Diese werden tendenziell erst mal versuchen, alle Münzen einzusammeln und je nach Lernrate, bestimmte Wege kein zweites Mal aufsuchen.

3.2. Die Eigenschaften und das taktische Verhalten des Akteurs

Für den Akteur wurden folgende 4 Eigenschaften vorgegeben:

1. Der Kundschafter:
   Bewegt sich über das ganze Spielfeld und probiert verschiedene Pfade aus. Er geht mit dem Bewegungsfluss des Balles mit und versucht die Münzen auf dem Weg einzusammeln.

2. Der Erfolgstyp:
   Sucht den erfolgreichsten Pfad und verbleibt auf diesem.

3. Der Ignorant:
   Nimmt keine Rücksicht auf seinen Gegenspieler und geht auch nicht auf ihn ein.

4. Der Folgende:
   Ist gänzlich passiv und folgt den Bewegungen des Gegenspielers.

Dabei ist es wichtig die Unterschiede zwischen 1 und 2; dem Kundschafter und dem Erfolgstyp, sowie zwischen 3 und 4; dem ignoranten und folgendem Verhalten abzubilden. Da das Zusammenspiel dieser Eigenschaften wichtig ist, wurde überlegt, wie die dazu passenden Charaktere designt werden können. Hierbei ist es ebenfalls wichtig, ein lernendes Verhalten zu simulieren. Der Akteur sollte in der Lage sein zu lernen, wo die unsichtbaren Hindernisse des Mitspielers liegen. Dafür wird ein Lernfaktor benötigt. Mit Lernfaktor ist in diesem Zusammenhang die Rate gemeint, wie oft ein Akteur über ein unsichtbares Hindernis fahren kann bevor er gelernt hat, dass sich dort eines befindet und einen anderen Pfad einschlägt. Dieser Lernfaktor sollte während der Experimente gesetzt werden können:

Typ 1 - Der ignorante Kundschafter:
Dieses Verhalten setzt sich aus den Eigenschaften 1 und 3 zusammen. Hier untersucht der Akteur das gesamte Spielfeld und versucht herauszufinden, wo sich beim Gegenüber die Hindernisse befinden. Er geht dabei nicht auf seinen Mitspieler ein.

Typ 2 - Der ignorante Erfolgstyp:
Dieser Typ besteht aus den Eigenschaften 2 und 3. Er möchte möglichst viele Punkte sammeln und wählt den für ihn am erfolgversprechendsten Pfad und bleibt auf diesem. Erfolgversprechend sind entweder besonders kurze, oder leicht ab zufahrende Pfade. Da es schwierig ist, durch viele Hindernisse zu navigieren wobei jede Kollision Abzug in der Gesamtwertung bedeutet, macht es Sinn einen weiteren Weg zu nehmen der dafür jedoch

möglichst wenig Hindernisse beinhaltet.

Typ 3 - Der folgsame Kundschafter:
Dieser besteht aus den Eigenschaften 1 und 2. Er ist passiv und wartet erst einmal ab, was sein Mitspieler tut. Dabei versucht er anhand der Bewegungen des Mitspielers herauszufinden, wo sich die unsichtbaren Hindernisse befinden.

Typ 4 - Der folgende Erfolgstyp:
Dieses Verhalten setzt sich aus den Eigenschaften 2 und 4 zusammen. Dieser verhält sich ähnlich wie Typ 3, mit dem Unterschied, dass er auch versucht herauszufinden was sein Mitspieler macht, um einen global-idealen Pfad zu finden.

Es gibt zwei Möglichkeiten, die in Kapitel 1.2 dargelegten Ziele mittels KI umzusetzen: maschinelles Lernen oder durch einen Akteur, der sich Algorithmen der klassischen KI bedient. Um Verhalten mittels maschinellem Lernen zu simulieren hätte es mehr Datensätze gebraucht als durch die experimentelle Datenerhebung vorhanden waren. Es gibt nur 25 Datensätze und die müssten auch noch jeweils in die verschiedenen Verhaltenstypen aufgesplittet werden. Ein neuronales Netz *NN* hingegen, benötigt wesentlich mehr Trainingsdaten. Somit wurde sich für die Lösung mit dem Akteur entschieden.

	Zeit	Livetests	Wiederverwendbarkeit	Zugänglichkeit
Ansatz 1	+ +	+ +	- +	- +
Ansatz 2	- -	- -	+-	+ +

Tabelle 3.2.: Gegenüberstellung der Vor- und Nachteile von der Liveumgebung gegenüber einer Nachimplementation

3.3. Vorgehen

Nachdem sämtliche Design-Entscheidungen rund um den Akteur getroffen wurden stand noch die Frage im Raum, wie entwickelt wird. Hierfür standen zwei Möglichkeiten zur Auswahl:

Ansatz 1: Die Entwicklung auf dem Livesystem vor Ort oder
Ansatz 2: die Nachimplementation des Ball-Game.

Bei Ansatz 1 gibt es erhebliche Vorteile gegenüber dem Testen der Software, sowohl während der Entwicklung, als auch im Einsatz. Außerdem spart dieser Ansatz wertvolle Zeit ein. Sowohl bei der Implementation als auch beim nachträglichen einspielen des Programms. Dafür ist sie aufgrund von zeitlichen Zugangsbeschränkungen zum Labor, nicht flexibel und es muss sich beim Entwickeln an freie Termine gehalten werden. Ansatz 2 hingegen ist sehr zeitintensiv. Sowohl in der Entwicklung als auch beim Nachträglichen einbinden und etwaigem Debuggen, da sich die Systeme evtl nicht zu 100% gleich Verhalten. Dafür gewinnt man hier mehr Flexibilität und hat ein portables Programm zu Demonstrationszwecken. Aufgrund der sich aufsummierenden Vorteile wurde sich für Ansatz 1 entschieden (siehe Tabelle 3.2). Im Laufe des Projekts stellte sich jedoch heraus, dass die Zugänglichkeit zu dem System ein Risiko darstellt, da es sehr häufig für Experimente zu Verfügung stehen muss. Somit wurde sich nachträglich für Ansatz 2 entscheiden.

4. Die Implementation des Akteurs

In folgendem Kapitel geht es um die Umsetzung des Designs aus Kapitel 3. Dabei wird in Kapitel 4.2 das Ball-Game für Testzwecke nach implementiert. Hauptaugenmerk wurde auf die Wegplanung und Pfaderstellung in Kapitel 4.3 gelegt, bevor es in Kapitel 4.4 an die Modellierung der Pfaderkennung geht. Denn diese ist für die Erstellung eines dyadischen Verhältnisses von Bedeutung. Auf alle Kapitel aufbauend, wird zum Schluss in Kapitel 4.5 auf die Abbildung der verschiedenen Verhaltenscharaktere gegangen.

4.1. Der Test-Server, eine Nach Implementation des Ball-Game

Zu Demonstrations- und Testzwecken wurde das Ball-Game reimplementiert (siehe Abbildung 4.1). Hierfür wurde die Python-Bibliothek pygame verwendet, die für Multimediaanwendungen wie Spiele entwickelt wurde. Weitere Abhängigkeiten umfassen unter anderem die Bibliotheken json, socket und math (weitere Informationen können der Tabelle A.2 entnommen werden). Bei der Implementation des Test-Servers sind folgende Punkte

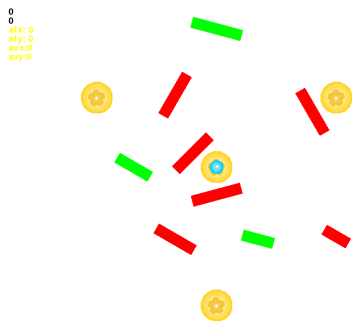

Abbildung 4.1.: Nachimplementation des Ball-Game

zu beachten. Die Hindernisse, die Münzen und der Ball werden in models.py modelliert und anschließend mittels einer Funktion aus game.py auf das Spielfeld gezeichnet. Hierbei ist zu beachten, dass die Position jedes Objektes in dem Spielfeld mit -1 multipliziert werden muss. Dies ist der Fall, da LabView im ersten Quadranten des kartesischen Koordinatensystems rechnet [man()], im Gegensatz zu den meisten gängigen Programmen, die im vierten Quadranten des kartesischen Koordinatensystem rechnen. Die Maße und Ausrichtung der Objekte wurden exemplarisch einigen Beispiel Frames des ursprünglichen Ball-Game entnommen. Im Unterschied zu der ursprünglichen Implementation des Ball-Game werden bei der Reimplementation die für den Akteur unsichtbaren Hindernisse

grün in das Spielfeld eingezeichnet. Da für das Test-Setup keine Bimetalle zur Verfügung stehen, wurde die Steuerung über die Pfeiltasten realisiert. Hierbei ist nochmal darauf hinzuweisen, dass sich in dem Experiment bewusst für die Bimetalle als unintuitive Steuerung entschieden wurde, da kein Spieler einen Vorteil gegenüber dem anderen haben soll. Hat z. B. ein Spieler bereits viel Zeit an einer Spielekonsole verbracht und würden in dem Experiment ähnliche Controller eingesetzt werden, hätte er einen Geschicklichkeitsvorteil gegenüber seinem Mitspieler und würde evtl. generell weniger Hindernisse streifen als der andere. Dieses müsste in der Interpretation des Spielverlaufes berücksichtigt werden, was wiederum auch bedeutet, dass es einen Messwert für die Geschicklichkeit der einzelnen Spieler geben müsste. Um die Ergebnisse eindeutiger interpretieren zu können, wird dieser Faktor durch die unnative Steuerung eliminiert. Das Steuerungserlebnis über die Pfeiltasten weicht erheblich von dem Steuerungserlebnis mit den Bimetallen ab, da es etwas einfacher und die Übersetzung direkter ist.

Hauptaugenmerk wurde bei der Entwicklung vor allem auf die korrekte Implementation der Physik-Engine gelegt um zu gewährleisten, dass sich der Akteur auf dem Zielsystem möglichst genauso verhält wie in der Test-Umgebung. Des weiteren vereinfachen die physikalischen Gegebenheiten im Spiel die Modellierung der natürlichen Bewegung des Balles (siehe Kapitel 2.3.2). Wird der Ball als ein gängiges Objekt einer analogen Welt betrachtet, so besitzt er Masse und damit Trägheit. Nach dem Trägheitsprinzip bleibt ein Körper so lange im Ruhezustand oder dem Zustand der gleichförmigen Bewegung, bis er durch einwirkende Kräfte(2.1) zur Änderung seines Zustands gezwungen wird. Hierbei kommt uns das Superpositionsprinzip der Kräfte zugute, denn wirken mehrere Kräfte auf einen Massepunkt, so addieren sich diese vektoriell zu einer Kraft auf und der Ball wird immer in Abhängigkeit von seiner aktuellen Richtung umgelenkt. D. h. dass der Ball keine scharfen Kurven fahren kann und automatisch größere Runden fährt. Dadurch müssen wir keine weiteren Elemente der Kurvendiskussion bei der Implementation der natürlichen Bewegung des Akteurs bedenken (siehe Kapitel 4.3.4). Der Ball unterliegt ebenfalls der Rollreibung wodurch er langsamer wird und nach dem Ausrollen zu einem Stopp kommt, vorausgesetzt es wirken keine weiteren Kräfte auf ihn ein.

4.2. Verbindung zum Zielsystem

Die Verbindung zum Zielsystem erfolgt über TCP/IP (siehe Kapitel 2.3.1). Eine vollständige Nachricht zwischen dem Akteur und dem Zielsystem wird hier Frame genannt und besteht aus einem UTF-8 encodiertem serialisiertem JSON-Objekt. Das Ende eines Frames wird durch einen Wagenrücklauf `('\r', 0x0a)` und / oder einem Zeilenvorschub `('\n', 0x0d)` gekennzeichnet. Unter Unix und Unix ähnlichen Betriebssystemen kann der Wagenrücklauf entfallen. JSON ordnet jedem Schlüssel einen Wert zu. Im obersten Level eines Frames wird immer zuerst der Schlüssel `MsgType` beachtet. Dieser legt fest welchem Typ das Frame entspricht. Wenn ein Frame nicht korrekt empfangen werden kann, beispielsweise durch eine Störung im Netzwerk, führt der Akteur dieselben Schritte durch wie beim Frametyp Ende. `MsgTypes` sind zulässig (siehe auch Kapitel 2.2.2):

- `Frame`
- `Experiment Definition`
- `Start`
- `Stop`
- `End`

Ablauf der Kommunikation

Die Kommunikation erfolgt erst, nachdem eine Verbindung auf TCP/IP hergestellt worden ist. Der Code für den Socket befindet sich in Listing A.3. Dabei werden die Informationen mit einer Rate von ca 35 FPS, wie folgt ausgetauscht:

1. Der Server generiert ein neues Spiel und sendet an den Client eine Versuch Definition (siehe Kapitel 2.2.2).
2. Es folgt ein Start Frame durch den Server an den Client.
3. Der Client wartet auf ein Frame.
4. Es wird ein Frame empfangen, der Client beantwortet dieses ebenfalls mit einem Frame welches die Information enthält, welche Kraft der Akteur auf den Ball ausüben möchte. Anschließend fährt der Client mit Schritt 3 fort.
5. Es wird ein Ende Frame vom Server empfangen. Der Client schließt die Verbindung und beendet die Runde.
6. Alle anderen Frametypen werden ignoriert.

Der Informationswechsel findet immer pro Frame statt, d. h. dass die Berechnungen über den nächsten Schritt ebenfalls pro Frame ausgeführt. Dies ist wichtig, damit adäquat auf den Mitspieler eingegangen werden kann. In den letzten zwei Kapiteln wurde die Grundlage geschaffen, um im folgenden mit der eigentlichen Implementation des Akteurs zu beginnen.

```
import socket
import json

class LabViewConnector:
def __init__(self, ip, port):
self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
self.sock.connect((ip, port))

def receive_fr(self):
try:
byteA = bytearray()
while True:
data = self.sock.recv(1)
if data[0] == 10 or data[0] == '\n':
break
else:
byteA.append(data[0])
str = byteA.decode("utf-8")
a = json.loads(str)
return a
except:
return json.loads('{"MsgType": "End"}')

def send_fr(self, obj):
str = json.dumps(obj).encode()
str += b"\r\n"
self.sock.send(str)

def close(self):
self.sock.close()
```

Listing 4.1: :Socket Code zur Kommunikation zwischen dem Spielserver und dem Akteur.

4.3. Die Wegeplanung und Pfaderstellung

4.3.1. Kontextinformationen und Sicht auf das Spielfeld

Mit den Informationen aus der Versuch Definition die vom Trial Definition Frame übermittelt werden, wird das Spielfeld mit den Hindernissen wie folgt in das Kontextbewustsein des Akteurs geladen:
Das Frame mit der Versuch Definition übergibt die Mittelpunkte, die Länge und Breite der Hindernisse in Form von x- und y-Koordinaten bzw. den Radius für die Münzen und den Ball, sowie die z-Koordinate für die Höhe und die Grad Zahl, um die die Hindernisse gedreht sind. Um die Implementation zu vereinfachen, werden sämtliche Formen nur zweidimensional dargestellt und die Höhe somit vernachlässigt. Da es sich hierbei lediglich um Kreise und Rechtecke handelt, können diese zweidimensional dargestellt werden, indem nur die äußersten Punkte auf das Feld gezeichnet werden. Das Spielfeld selbst hat eine Größe von 768×124px und wird durch eine Nullmatrix repräsentiert. Jedes Hindernis wird zunächst einzeln in eine Matrix (als Repräsentant für das Spielfeld) platziert, wobei an die durch die Versuch Definition vorgegebene Position Einsen geschrieben werden. Um die Positionen der rotierten Hindernisse zu berechnen werden folgende Schritte durchgeführt: Seien k Einträge mit einer 1 in der Matrix, dann werden die Indizes in eine $k \times 2$ große Matrix O geschrieben.

$$\text{O}= \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_k & y_k \end{pmatrix}$$

Um das Hindernis um eine beliebige Grad Zahl drehen zu können, wird das Objekt auf den Ursprung projiziert. Dies wird dadurch erreicht, dass der Mittelpunkt (x_m, y_m) des Objektes elementweise von allen Einträgen in der Matrix subtrahiert wird.

$$M = O \ominus (y_m, y_m) = \begin{pmatrix} x_1 - x_m & y_1 - y_m \\ x_2 - x_m & y_2 - y_m \\ \vdots & \vdots \\ x_k - x_m & y_k - y_m \end{pmatrix}$$

Die daraus resultierende Matrix M, wird dann mit einer Rotationsmatrix (2.8) multipliziert und wir erhalten M'. Anschließend werden Mittelpunkte wieder hinzuaddiert und das rotierte Objekt O' in das eigentliche Spielfeld übertragen.

$$O' = M' \oplus (y_m, y_m) = \begin{pmatrix} x'_1 + x_m & y'_1 + y_m \\ x'_2 + x_m & y'_2 + y_m \\ \vdots & \vdots \\ x'_k + x_m & y'_k + y_m \end{pmatrix}$$

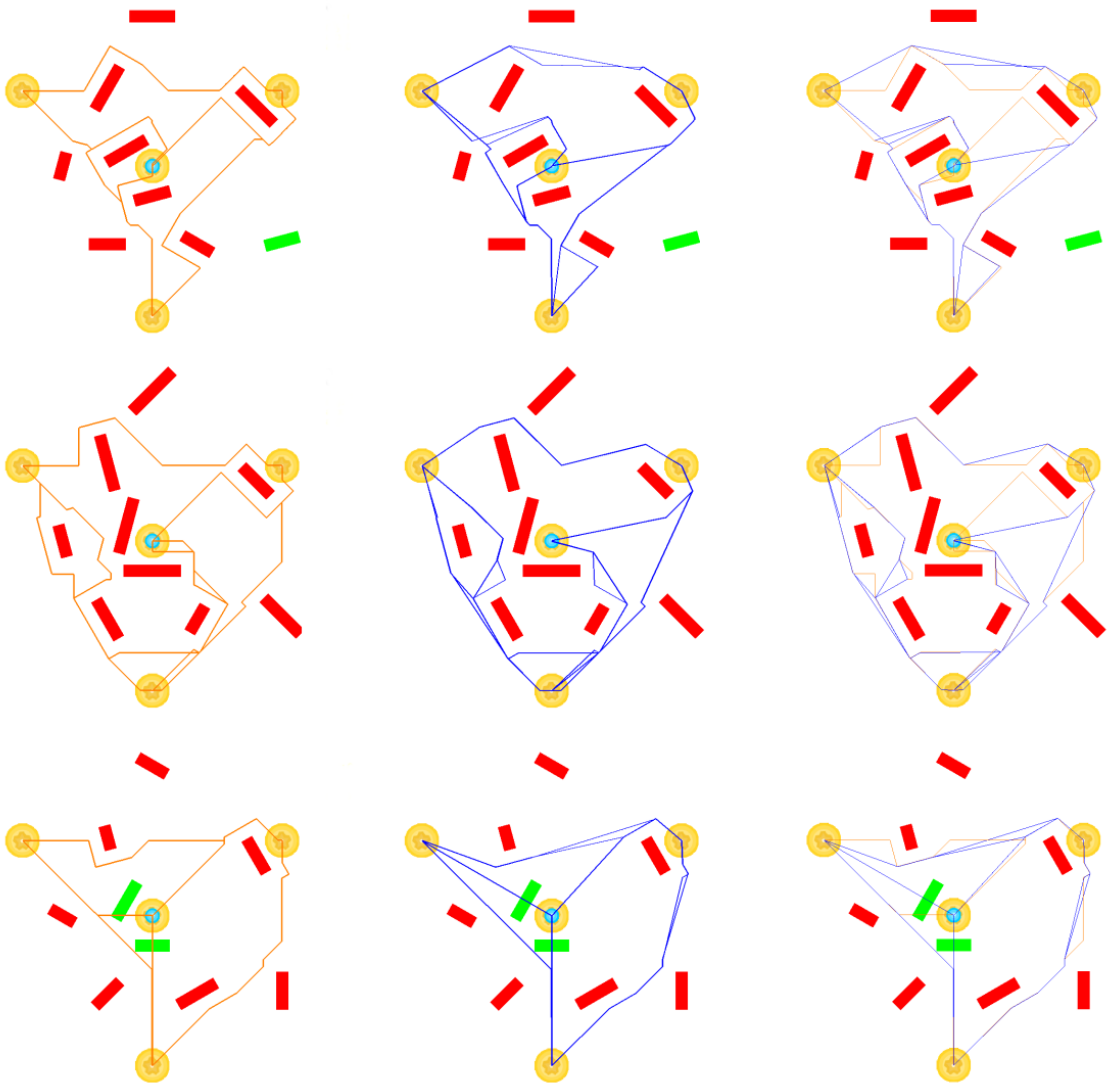

Abbildung 4.2.: Die Pfade nach der Vereinfachung *(links)*, nach der Interpolation *(mitte)*, sowie beide Pfade übereinander gelegt. *(rechts)*

4.3.2. Wegefindung mit dem A*-Algorithmus

Im vorherigen Abschnitt wurde dargelegt, wie der Akteur seine Sicht auf das Spielfeld generiert. Er hat nun alle relevanten Informationen um mithilfe des A*-Algorithmus einen Weg zwischen den Hindernissen hindurch, zu einer Münze zu finden (siehe Kapitel (3.1.5)). In der Implementation des Akteurs wird der A*-Algorithmus wie folgt eingesetzt: Es werden von jeder Münze alle Pfade zu jeder anderen Münze berechnet und gespeichert (siehe Abbildung 4.2). Wenn die Münzen als Knoten eines Graphen betrachtet werden, dann gibt es bei 4 Knoten

$$\binom{4}{2} = \frac{4!}{2! \cdot (n-k)!} = 6$$

Kanten [Perl(1981)]. Das heißt, dass der A*-Algorithmus sechsmal aufgerufen wird. Dieser Schritt war nötig, um die Rechenzeit zu minimieren, da der A*-Algorithmus diesbezüglich

sehr teuer ist und zu viele Aufrufe das Programm zu stark verlangsamt hätten. Wie in Abbildung 4.2 deutlich zu erkennen, wirken die vom A*-Algorithmus berechneten Pfade durch ihre Kanten innerhalb des Pfades nicht natürlich. In Kapitel 4.3.4 wird erläutert, wie das Problem mittels Interpolation gelöst wird.

4.3.3. Modellierung der Pfade

Eine wesentliche Kernaufgabe des Akteurs ist es, die Münzen einzusammeln. Mittels des A*-Algorithmus lassen sich zwar die Pfade um die Hindernisse herum bestimmen. Diese Pfade sind jedoch Arrays mit Tupel, die je eine x und eine y Koordinate haben. Um einfach und effizient mit diesen Daten zu arbeiten, müssen diese zunächst aufgearbeitet werden. Dafür werden die Tupel in Punkte vom Typ `Point` umgewandelt. Anschließend wird aus dem Array, bestehend aus `Points`, ein `Path` Objekt erzeugt. Dies schafft enorme Flexibilität, da nun auf einer höheren Abstraktionsebene objektorientiert mit dem Pfad gearbeitet werden kann. Es werden zu jedem Pfad auch Zusatzinformationen gespeichert, z.B. ob es von diesem Pfad bereits eine durch Interpolation (siehe Kapitel 4.3.4) vereinfachte Version gibt. Hierfür werden `Journey` Objekte benutzt. Sie koppeln mehrere Pfade mit dem gleichen Start und Ziel zusammen. Jede Münze wird durch ein `Target` Obejekt repräsentiert. Ein `Target` kennt mindestens drei `Journey` Objekte die jeweils von dieser Münze zu einer anderen führen. Die Abstraktion durch diese Objekte ermöglicht dem Akteur mittels einfacher Funktionen komplexe Abfragen an die Kontextinformationen des Spielfelds zustellen (siehe Abbildung 4.3).

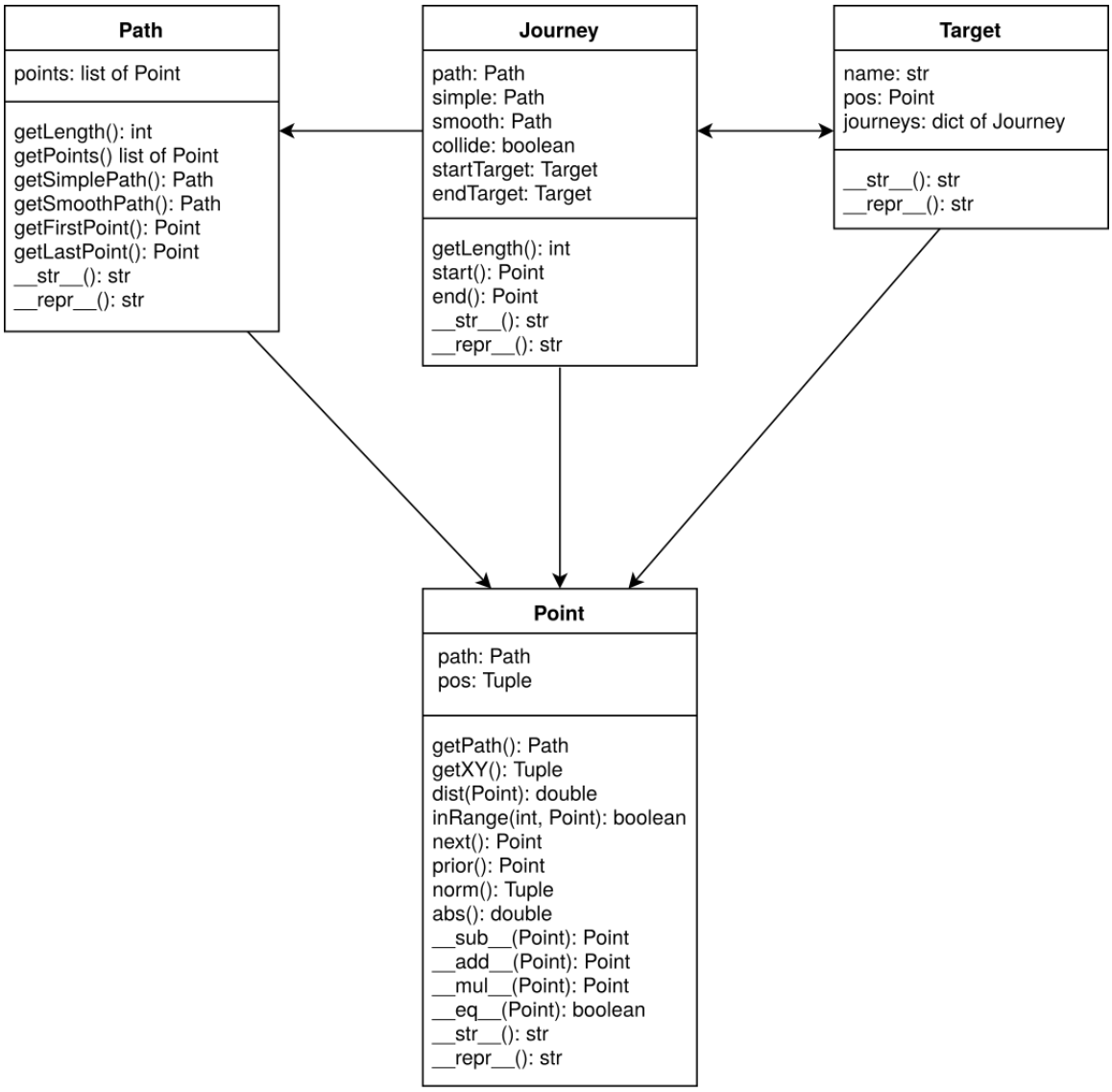

Abbildung 4.3.: Die Relation des Abstrahier Vorgangs der Pfade

4.3.4. Die natürliche Bewegung des Balles auf dem Pfad

Da sich das Verhalten des Akteurs im Endeffekt nicht sonderlich von dem eines humanoiden Spielers unterscheiden soll, ist es wichtig die Pfade mit ihren Kanten, die bei der Berechnung mit dem A*-Algorithmus entstehen (siehe Abbildung 4.2), zu glätten. Um die Rechenzeit zu minimieren wird zuvor der Pfad vereinfacht, indem sämtliche Punkte auf Geraden entfernt werden. Dadurch wird ein Pfad erzeugt, der weniger Punkte enthält. Um dies zu erreichen wird durch den Pfad, der eine Liste von Punkten der Form $P = (x, y)$ enthält, iteriert. Für jeden Punkt P_n und seinen Nachfolger P_{n+1} gilt: entspricht die Differenz $d = P_{n+1} - P_n$ des aktuell betrachteten Punktes P_n zu seinem Nachfolger P_{n+1} der Differenz aus vorherigem Iterationsschritt, so wird P_n aus der Liste entfernt (siehe Abbildung 4.4). Somit entsteht ein vereinfachter Pfad, der nur noch aus Punkten besteht, die entweder einen Richtungswechsel angeben oder das Ziel darstellen.

Im nächsten Schritt wird geprüft, ob sich zwischen einem Punkt P_1 und dem Folgepunkt P_2 ein Hindernis befindet. Ist dies nicht der Fall, wird P_2 ebenfalls von der Liste genommen und gespeichert. Befindet sich nun zwischen P_1 und P_3 ebenfalls kein Hindernis, wird P_2 gelöscht und P_3 gespeichert. Wenn nun zwischen P_1 und P_4 ein Hindernis liegt, dann wird der zwischengespeicherte Punkt P_3 angesteuert. Um eine direkte Verbindung auf dem Raster zwischen den Zwei Punkten herstellen zu können, muss der Ansatz der linearen Interpolation erweitert werden, indem beide, x und y Werte, interpoliert werden, statt nur den x Wert. Um zu entscheiden, wie viele Punkte zwischen P_1 und P_3 gesetzt werden, muss zuerst die diagonale Distanz N zwischen den zwei Punkten berechnet werden. Die Distanz fließt in den Laufparameter t mit ein. Denn zu viele Punkte können zu einer Überlagerung führen und zu wenige zu Lücken innerhalb des Pfades. Eine Überlagerung wäre rechenintensiver als nötig und bei zu großen Lücken, kann nicht mehr garantiert werden, dass Hindernisse die auf dem Weg liegen, zuverlässig erkannt werden. Es werden zuerst die x sowie die y Werte von P_1 von P_3 abgezogen:

$$x_d = |x_3 - x_1|$$
$$y_d = |y_3 - y_1|$$

Anschliessend wird das Maximum der Beträge beider Werte berechnet

$$N = max(x_d, y_d) = \begin{cases} x & ; x \leq y \\ y & ; \text{sonst} \end{cases}$$

und jeder Punkt auf der Linie wird mit ihrem Laufparameter t, interpoliert.

$$t = \frac{step}{N} \text{ mit } 0 \leq step \leq N$$

$$x_{P_a} = x_1 + t \cdot (x_1 - x_3)$$
$$y_{P_a} = y_1 + t \cdot (y_1 - y_3)$$

Es wurde lineare Interpolation (2.7) in 2D angewandt, um die Pfade in einem zweidimensionalen Raum zu glätten. Des Weiteren unterstützt die Physik-Engine aus dem Ball-Game die natürliche Bewegung auf dem Pfad immens. Durch die in Kapitel 2.3.2 dargelegten Axiome, ist es nicht möglich z.B. die Bewegung des Balles, durch einen ungünstigen Pfadverlauf, zu kantig zu gestalten. Der Ball fährt aufgrund seiner Masse und Trägheit ohnehin in einem größeren Bogen um Kurven. Die einzige Erweiterung die von Implementationsseite her noch getan wurde, um den Ballverlauf natürlich wirken zu lassen, ist das Abbremsen kurz vor einem Ziel. Damit der Ball nicht über das Ziel hinaus schießt, wird auf den letzten 10 Pixeln vor dem Ziel der Richtungsvektor halbiert, wenn dessen Absolutwert ≤ 1 ist, und in die entgegengesetzte Richtung gedreht.

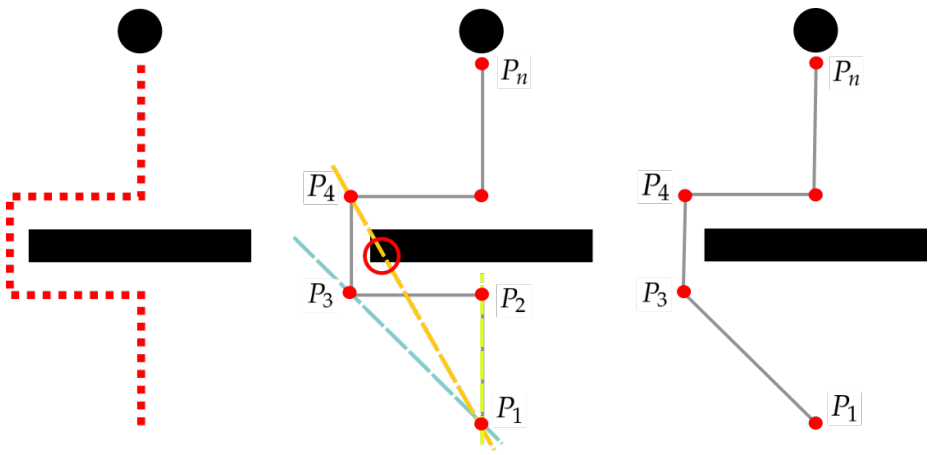

Abbildung 4.4.: Die Pfade nach Berechnung mit A* *links*, nach der Vereinfachung (*mitte*), sowie nach der Interpolation (*rechts*)

Der Akteur ist nun in der Lage, sich auf eine natürlich wirkende Art und Weise, auf den vordefinierten Pfaden zu einem gegebenen Hindernis zu bewegen.

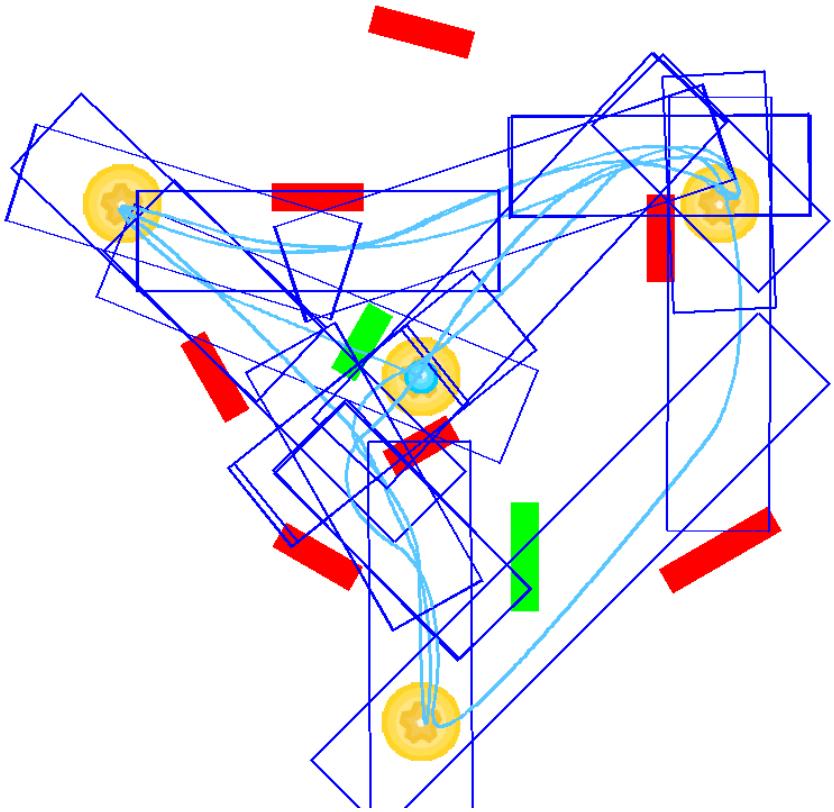

Abbildung 4.5.: Die vom A* berechneten Pfade mit Bounding Box für Pfaderkennung

4.4. Pfaderkennung

Da es sich bei dem Ball-Game um ein dyadisches Spiel handelt, muss mit einer zweiten Kraft umgegangen werden, die ebenfalls auf den Ball einwirkt. Als Folge kann der Ball dadurch von seinem exakten Pfad abgebracht werden. Jedoch ist es für den Akteur oft notwendig zu wissen, auf welchem Pfad er sich befindet oder ob er seinen aktuellen Pfad verlassen hat. Dies ist insbesondere für die Folgenden-Verhaltensaspekte notwendig (siehe Kapitel 3.2), da dort der Akteur schnell und zuverlässig erkennen muss, ob es klüger ist, dem Mitspieler weiter zu folgen oder seinen gewählten Pfad abzuarbeiten.

Es ist möglich, ausgehend von der aktuellen Position des Balles, die Punkte aller Pfade dahingehend zu prüfen, ob sich ein weiterer Punkt oder mehrere in der Nähe befinden. Dies ist aber sehr rechenintensiv und das Ergebnis ist nicht in allen Fällen eindeutig, z.B. bei Überschneidungen von Pfaden. Für eine effiziente Pfaderkennung werden bei Auswahl eines Pfades durch den Akteur eine Liste von Bounding Boxen angelegt. Eine Box wird um je zwei aufeinander folgende Punkte des Pfades gezeichnet (siehe 4.4). Es folgt dass es bei $n \leq 2$ Punkten $n - 1$ Boxen gibt. Die Box wird künstlich vergrößert, um nicht nur die Punkte, sondern auch den Raum um sie herum mit einzubeziehen. Die Boxen er-

möglichen nun, dass sich der Akteur frei auf seinem Pfad bewegen kann, ohne ihm exakt zu folgen. Außerdem wird die Prüfung, ob der Akteur sich noch auf einem Pfad befindet, von vielen Berechnungen auf Punkten zu wenigen Kollisionsprüfungen mit den Boxen reduziert. Der Wert der Vergrößerung einer Box beeinflusst das Verhalten dahingehend, dass kleinere Boxen den Akteur öfters dem Mitspieler folgen lassen, während größere die Chance erhöhen, dass ein Pfad wieder aufgenommen wird (und daher das Folgen beendet). Als guter Wert hat sich der vierfache Ballradius erwiesen. In den Abbildungen 4.5 - 4.9 wurden zu Demonstrationszwecken, die vom Ball während eines Versuchs zurückgelegten Pfade, auf die Spielfläche gezeichnet. In den letzten Schritten wurden die Pfade inkl. derer Metainformationen modelliert. Dies sind die benötigten Grundlagen, um darauf aufbauend im Folgekapitel die eigentliche Verhaltensmodellierung vorzustellen.

```
1  def bounding_box(self, journey):
2  self.rects = []
3  point = journey.smooth.getFirstPoint()
4  while point.next() is not None:
5  width = point.dist(point.next()) + 190
6  height = 90
7  cx, cy = (point.next() - point).getXY()
8  center = (point + sP.Point(None, (cx/2, cy/2))).getXY()
9  angle = math.atan2(cy, cx)
10 self.rects.append(self.Rect(center, height, width, angle))
11 point = point.next()
```

4.5. Die Modellierung der verschiedenen Eigenschaften des Akteurs

Um die in Kapitel 3.2 dargelegten Überlegungen bezüglich der verschiedenen Verhaltensweisen umzusetzen, war es am schwierigsten das Folgeverhalten zu simulieren. Der Ignorante Typ ist hingegen recht simpel, da er nicht auf den Input seines Mitspielers eingeht, muss nichts weiter beachtet werden. Egal was passiert, er wird immer versuchen zurück zu seinem Pfad zu gelangen. Mit Input ist hier eine Eingabe des anderen Spielers gemeint. Der Typ (Kundschafter oder Erfolstyp) entscheidet der Akteur welches sein aktueller Pfad ist auf den er zurück möchte. Nicht so einfach war es das Folgeverhalten zu simulieren. Hierbei geht der Akteur, anders als bei ignorantem Verhalten, mit dem Mitspieler mit wenn dieser woanders hin möchte. Dieses Verhalten wird dadurch erreicht, dass der Akteur den Input des Mitspielers verstärkt, also mit in seine Richtung steuert. Hierbei spielen die mittels `Bounding Box` erweiterten Pfade aus voherigem Kapitel 4.4 eine fundamentale Rolle. Denn bringt der Mitspieler den Akteur weit genug von seinem aktuellen Pfad ab, fängt dieser an ihm zu folgen. Weit genug vom Pfad abgekommen ist der Akteur dann, wenn er sich nicht mehr innerhalb der Boxen, die um den eigentlichen Pfad liegen, befindet. Wird er stark genug vom Kurs abgebracht, so halbiert er lediglich den Input des Mitspielers. Die Kraft des anderen Spielers erhält der Akteur aus dem Versuch Frame (A.2). Hört der andere Spieler nun auf zu Steuern, so übernimmt der Akteur wieder eigenständig. Was er dann tut, hängt von dem jeweiligen Typ ab. Im Folgenden ist das modellierte Verhalten für die jeweiligen Typen, aufgelistet.

Typ 1 - Der ignorante Kundschafter:

- Fahre jeden Pfad einmal ab.
- Lösche dabei die Pfade, bei denen auf ein unsichtbares Hindernis getroffen wurde.
- Fahre weiterhin alle übrigen Pfade ab, bis die Runde vorbei ist.
- Ignoriere das Gegensteuern des Mitspielers und versuche wieder auf einen der Pfade zu gelangen. Je nachdem wo die Gegenkraft hinlenkt, nimm den nächst dichtesten Pfad aus der Liste mit den Pfaden, die entweder noch nicht besucht wurden oder keine Hindernisse im Weg enthalten.

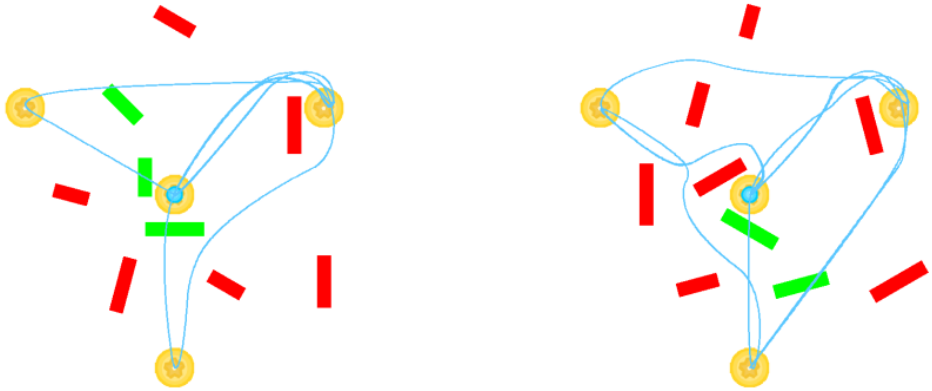

Abbildung 4.6.: Der gelaufene Pfad von Typ 1 mit Lernrate 2

Typ 2 - Der ignorante Erfolgstyp:

- Wähle den Pfad, der am wenigsten Punkte nach der Interpolation hat.
- Wenn zweimal ein unsichtbares Hindernis gestreift wurde, lösche den Pfad und wähle den nächst kürzesten Pfad.
- Ignoriere das Gegensteuern des Mitspielers und versuche immer wieder auf den aktuellen Pfad zu gelangen.
- Es sei denn, der Kurs ist zu abtrünnig, dann versuche auf den am dichtesten gelegenen Erfolgspfad zu landen.

Um dem letzten Punkt zu realisieren werden die Pfade bei der Initialisierung klassifiziert. Die Pfade werden aufsteigend, nach ihrer Punktanzahl sortiert und jede mögliche Position auf dem Spielfeld wird einem Pfad zugeordnet. Somit weiß der Akteur, in Abhängigkeit von seiner, durch den Mitspieler verschobenen Position, wo er sich als nächstes hinbewegen muss um trotzdem noch erfolgreich zu sein.

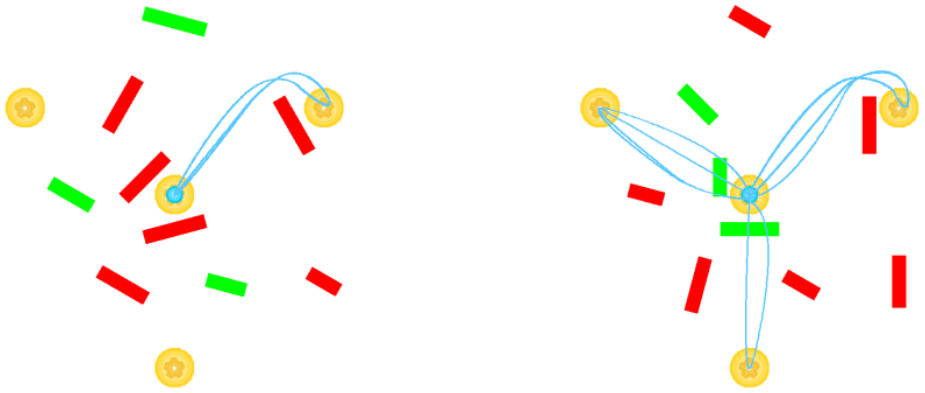

Abbildung 4.7.: Der gelaufene Pfad von Typ 2, ohne unsichtbares Hindernis *(links)*, sowie mit unsichtbarem Hindernis und Lernrate 2. *(rechts)*

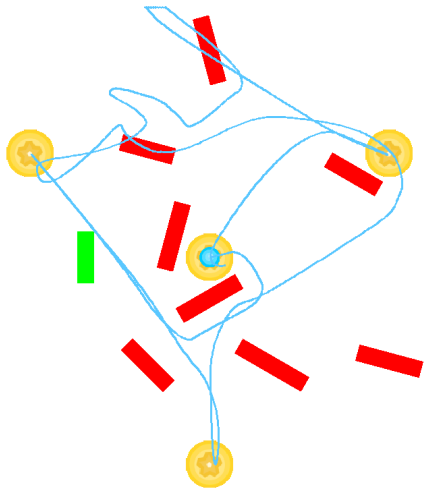

Abbildung 4.8.: Der gelaufene Pfad von Typ 3 mit Gegenkraft

Typ 3 - Der folgsame Kundschafter:

- Wenn kein Input vom Gegenspieler kommt, verhalte dich wie Typ 1.
- Sobald Input vom Gegenspieler kommt, tue nichts.
- Merke dir währenddessen, welche Münzen bereits eingesammelt wurden.
- Wenn kein Input mehr kommt:
  - Steuere den nächst gelegenen Pfad zu einer Münze an, die noch nie eingesammelt wurde, unsichtbare Hindernisse sind nicht bekannt.
  - Wenn alle Münzen bereits eingesammelt wurden, wähle den dichtesten Pfad und umgehe dabei die unsichtbaren Hindernisse.

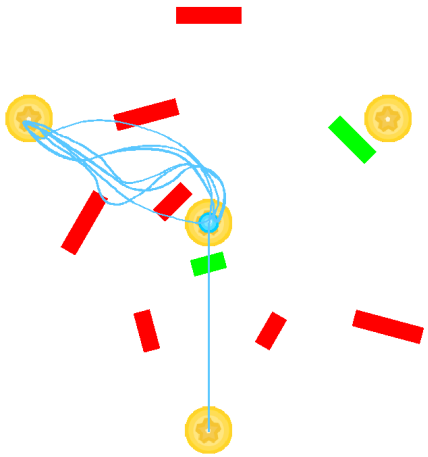

Abbildung 4.9.: Der gelaufene Pfad von Typ 4 mit Gegenkraft

Typ 4 - Der folgende Erfolgstyp:

- Wenn kein Input vom Gegenspieler kommt, Verhalte dich wie Typ 2.
- Sobald Input vom Gegenspieler kommt, tue nichts, außer es befindet sich ein für dich sichtbares Hindernis im Weg, dann steuere dagegen.
- Sobald ein unsichtbares Hindernis gestreift wurde, merke dir dieses.

5. Zusammenfassung

In dieser Arbeit ist es gelungen, einen Computer gesteuerten Akteur zu entwickeln, der sich parametrisiert auf die in Kapitel 3.2 dargelegten Verhaltensweisen einstellen lässt. Hierbei wurde auf klassische Algorithmen der KI zurückgegriffen. Aufgrund der Tatsache, dass i. d. R. permanent von einem Spieler gegen die Bewegungen des Akteurs an- bzw. mitgesteuert wird, ist das Verhalten wie hier implementiert, hinreichend um während des Spiels nicht von einem humanoiden Spieler unterscheidbar zu sein. Was aus zeitlichen Gründen nicht mit berücksichtigt wurde, ist die Einflussnahme durch die Steuerung selber. Ein humanoider Spieler weist über die hier geforderten Charakteristiken noch mehr Eigenschaften auf. Er könnte z. B. sehr stark zittern, was sich durch die Multiplikation mit einem zufällig generierten Vektor auf die Ausgabe des Akteurs hätte simulieren lassen. Eine Möglichkeit zögerliches oder unsicheres Verhalten zu simulieren wäre es gewesen, in ungleichen Abständen, um verschiedene Zeitabstände, nur auf jedes 5 - 20te Frame zu reagieren. Oder um generelle Spielgeschwindigkeiten abzubilden, nur jedes xte Frame in Betracht zu ziehen. Des Weiteren läuft das Spiel zwar dahingehend, dass die Verbindung, sowie der Datenaustausch auf dem Lifesystem funktioniert. Jedoch muss noch das Interface für die Bedienung und Parametervergabe durch LabVIEW, bereitgestellt werden. Die Integration während des Ablaufs der Datenerhebung hat ebenfalls noch nicht stattgefunden. Die letzten beiden Punkte liegen im Aufgabenbereich der Experiment Leitung. Darüber hinaus wäre auch ein völlig anderer Ansatz denkbar gewesen. Es werden in Zukunft noch weitere Datensätze erhoben. Bei hinreichender Datenmenge wäre auch die in Kapitel 3.2 erwähnte Lösung mittels eines NN denkbar. Hierbei wird der Input aus den Bimetallen zu Trainingszwecken direkt in das NN gegeben. Dabei würden die Spielverläufe aller erhobenen Aufzeichnungen während des Spiels als Trainingsdaten verwendet. Dadurch wird eine KI erzeugt, welche von vorneherein den direkten Input der Bimetalle, also des Spielers, simuliert. Eigenschaften wie Zögerlichkeit oder Aggressivität wären bereits abgebildet.

Literaturverzeichnis

[man()] *Handbuch zu LabVEIW*.

[Block(2010)] S. Block. *Physik*. Compact SilverLine : Taschenbuch. Compact Verlag GmbH, 2010. ISBN 9783817490646. URL https://books.google.de/books?id=85SQp4oDCM8C.

[Blog(2015)] Tobi's MINT Blog. Client-server-modell, 2015. URL https://mineyourlife.net/client-server-modell/.

[Lenz(2009)] Karl Lenz. *Soziologie der Zweierbeziehung: Eine Einführung*. Springer, 2009.

[Mandl et al.(2010)Mandl, Bakomenko, and Weiß] Peter Mandl, Andreas Bakomenko, and Johannes Weiß. *Grundkurs Datenkommunikation: TCP/IP-basierte Kommunikation: Grundlagen, Konzepte und Standards*. Springer-Verlag, 2010.

[Perl(1981)] Jürgen Perl. *Graphentheorie: Grundlagen und Anwendungen;[Studienbuch für Studenten der Mathematik und Informatik, der Wirtschafts-und Ingenieurwissenschaften]*. Akad. Verlag-Ges., 1981.

[Poole and Mackworth(2010)] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[Roland Bauch(2004)] Thomas Beer Roland Bauch. *Netzwerke – Grundlagen*. HERDT-Verlag für Bildungsmedien, 2004. ISBN 978-3-8237-1140-7.

[Schmidhuber(2016)] Jürgen Schmidhuber. Künstliche intelligenz wird alles ändern, 2016. URL https://www.youtube.com/watch?v=rafhHIQgd2A.

[Smith(2015)] Ben Smith. *Beginning JSON*. Apress, 2015. ISBN 978-1-4842-0202-9. URL http://dx.doi.org/10.1007/978-1-4842-0202-9.

[Wikipedia(2014)] Wikipedia. Interpolation, 2014. URL https://de.wikipedia.org/wiki/Interpolation_(Mathematik). *Stand vom 04.08.2014*.

[Wikipedia(2018a)] Wikipedia. Drehmatrix, 2018a. URL https://de.wikipedia.org/wiki/Drehmatrix. *Stand vom 11.08.2018*.

[Wikipedia(2018b)] Wikipedia. Gleichmäßig beschleunigte bewegung, 2018b. URL https://de.wikipedia.org/wiki/Gleichm%C3%A4%C3%9Fig_beschleunigte_Bewegung. *Stand vom 13.12.2018*.

[Wikipedia(2018c)] Wikipedia. Einehitsvektor, 2018c. URL https://de.wikipedia.org/wiki/Einheitsvektor. *Stand vom 03.01.2019*.

[Wikipedia(2019)] Wikipedia. Labview, 2019. URL https://de.wikipedia.org/wiki/LabVIEW. [Stand vom 07.01.2019].

[Xue and Zhu(2009)] M. Xue and C. Zhu. The socket programming and software design for communication based on client/server. In *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, pages 775–777, May 2009. doi: 10.1109/PACCS.2009.89.

A. Anhang

A.1. Benutzte Bibliotheken

Version
Python 3.6
Bibliotheken
heapq v 2.3.0 numpy v 1.11.0 scipy v 0.17.0

Tabelle A.1.: Details zu benutzten Bibliotheken des Akteurs.

Version
Python 3.6
Bibliotheken
pygame v 1.9.4 json v 2.0.9 socket math

Tabelle A.2.: Details zu den Bibliotheken des Testservers.

A.2. Frames

MsgType: Start
Protocol Version:1

MsgType: Experiment Definition
Nr of Trials: 30

MsgType: Trial Definition
Level Data : Trial Type: 1
    Trial Duration [ms]: 60000
    Nr of Targets: 4
    Nr of Obstacles: 9
    Ball:
        X: 512
        Y: 384
        Z: 15
        Radius: 15
    Targets:
        X: [512,246,778,512]
        Y: [691,230,230,384]
        Z: [5,5,5,5]
        Z-size: [10,10,10,10]
        Radius: [35,35,35,35]
    Obstacles:
        X: [512,459,725,420,328,420,512,604,778]
        Y: [445,353,261,544,384,224,77,544,538]
        Z: [5,5,5,5,5,5,5,5,5]
        X-size: [25.600000000000001421,25.600000000000001421,25.600000000000001421,
            25.600000000000001421,25.600000000000001421,25.600000000000001421,
            25.600000000000001421,25.600000000000001421,25.600000000000001421]
        Y-size: [68.266666666666694141,68.266666666666694141,68.266666666666694141,
            102.40000000000000568,102.40000000000000568,102.40000000000000568,
            153.59999999999999432,102.40000000000000568,153.59999999999999432]
        Z-size: [10,10,10,10,10,10,10,10,10]
        Z-angle-deg: [90,210,330,20,100,80,180,280,300]
        geometric type: [1,1,1,1,1,1,1,1,1]
        slowdown factor: [0.05000000000000002776,0.05000000000000002776,
            0.05000000000000002776,0.05000000000000002776,
            0.05000000000000002776,0.05000000000000002776,

0.0500000000000000002776,0.0500000000000000002776,
0.0500000000000000002776]
visibility: [3,3,3,3,3,4,3,4,4]
AI Type: 0
AI 1 Length of Memory: 18
Questionair Text:I didnt define a cool Questionair Text
Blink-Wavelength-Screen: 40
Blink-Wavelength-OWG: 255
Screen-Flicker-Target-Radius: 20
Max Force: 0.00050000002374872565269
ShowBar: false
Nr-of-Frames-to-Skip-at-Start-of-Trial: 120
Show-Fixation-Cross: true
Visibility per Frame - Ball- Player 1:[]
Visibility per Frame - Targets - Player 1:[]
Visibility per Frame - Obstacles - Player 1:[]
Visibility per Frame - Obstacles 2 - Player 1:[]
Visibility per Frame - Ball- Player 2:[]
Visibility per Frame - Targets - Player 2:[]
Visibility per Frame - Obstacles - Player 2:[]
Visibility per Frame - Obstacles 2 - Player 2:[]
Replay File: empty string
Replay X Position Ball Player 1:[]
Replay Y Position Ball Player 1:[]
Replay Z Position Ball Player 1:[]
Replay X Position Ball Player 2:[]
Replay Y Position Ball Player 2:[]
Replay Z Position Ball Player 2:[]
Replay Trigger States:[]
Replay X Axis Rotation Ball Player 1:[]
Replay Y Axis Rotation Ball Player 1:[]
Replay Z Axis Rotation Ball Player 1:[]
Replay Angle Rotation Ball Player 1:[]
Replay X Axis Rotation Ball Player 2:[]
Replay Y Axis Rotation Ball Player 2:[]
Replay Z Axis Rotation Ball Player 2:[]
Replay Angle Rotation Ball Player 2:[]
Nr-of-Frames-to-Show-Fixation-Cross: 120
Current Trial Nr: 1

A.3. Code Statistik

Anzahl an Dateien im git-Projekt: 54

```
1 git ls-tree -r --name-only HEAD | grep -v "pyc" | wc -l
```

Listing A.1: :Zählt alle dateien im Git Projekt

Da viele von den Dateien auf die Visualisierung(Bilder der Models) und auch auf statische Spieldefinition zwecks Debugging entfallen, macht es Sinn noch einmal einen Blick auf den reinen Code zu werfen. Anzahl an Python-Dateien im Projekt: 15

```
1 find ./ -name "*.py" | wc -l
```

Listing A.2: :Listet Dateien mit endung ".pyrekursiv auf den Aktuellen Ordner

Anzahl an Codezeilen(ohne Leerzeilen) in den Python-Dateien: 1720. Davon verfallen 1168 Zeilen auf den Akteur und 455 Zeilen auf den Testserver. Die verbliebenden 97 Zeilen entfallen auf ein Skript das zur Visualisierung einzelner Verhaltensweisen dient und passende Bilder generiert.

```
1 find ./ -name "*.py" -exec cat {} \; | awk "NF" | wc -l
```

Listing A.3: :Zählt alle Zeilen in allen Python-Dateien ohne Leerzeilen

Universitätsklinikum Hamburg-Eppendorf

Institut für Neurophysiologie und Pathophysiologie

Prof. Dr. med. Andreas K. Engel
Institutsdirektor

Zentrum für Experimentelle Medizin
Martinistraße 52
20246 Hamburg
Telefon: (040) 7410-56170
Telefax: (040) 7410-57752
Email: ak.engel@uke.de
www.uke.de/neurophysiologie

Studienprotokoll

„Die Rolle sensomotorischer Kontingenzen bei sozialer Kopplung“

Zielsetzungen und Zweck des Forschungsvorhabens

Das übergreifende Ziel des hier beschriebenen Vorhabens ist es, neuronale und Verhaltenskorrelate sozialer Interaktion und interpersonaler Kopplung zu untersuchen. Theoretische Grundlage der geplanten Experimente ist ein handlungsorientierter Zugang zu sozialer Kognition, der selbst komplexe Modi menschlicher Interaktion zurückführt auf basale sensomotorische Interaktionsmuster, so genannte sensomotorische Kontingenzen (sensorimotor contingencies, SMCs). Als grundlegend für soziale Interaktion wird hierbei die Fähigkeit zur Vorhersage von Handlungen und Handlungskonsequenzen anderer Akteure angesehen. Die Untersuchungen werden sich konkret der Frage widmen, ob das Erlernen und Beherrschen dieser SMCs im Rahmen eines zu zweit zu absolvierenden Geschicklichkeitsspiels auch zu einer veränderten sozialen Interaktion zwischen den Spielern führt. Der Frage nach dem Zusammenhang von sensomotorischer und sozialer Kopplung soll durch die Analyse von Bewegungs- und hirnphysiologischen Parametern sowie Fragebogen-Daten (Probanden-Ratings zum subjektiv empfundenen Involviert-Sein, „Wir“-Gefühl und der Qualität der Interaktion zwischen den Akteuren) nachgegangen werden. Die neuronale Aktivität des Gehirns soll durch Elektroenzephalographie (EEG) gemessen werden. Zusätzlich soll die Rolle sozial relevanter SMCs bei Patienten mit Autismus-Spektrum-Störung (ASS), die massive qualitative Beeinträchtigungen in sozialer Interaktion aufweisen, erörtert werden. Das hier beschriebene Vorhaben umfasst 2 Studien, die in 3 (erste Studie) bzw. 2 (zweite Studie) Teilexperimente unterteilt sind. Mit den geplanten Studien sollen folgende übergreifende Hypothesen getestet werden: (1) Das Erlernen und Beherrschen von SMCs in einem dyadischen Geschicklichkeitsspiel ist assoziiert mit dem sozialen Interaktionsmuster zwischen den beiden Spielern. (2) Soziale Kopplung bzw. Ratings zum subjektiven Involviert-Sein und der Qualität der Interaktion spiegeln sich wider in neuronaler Kopplung und in Korrelationen zwischen den erhobenen Bewegungsparametern der beiden Akteure. (3) Sozial-kognitive Defizite bei Patienten mit ASS sind assoziiert mit Defiziten im Erwerb und der Beherrschung von SMCs. Primärer Endpunkt in beiden geplanten Studien (siehe unten) ist eine Quantifizierung des Zusammenhangs zwischen sensomotorischer und sozialer Kopplung in dyadischer Interaktion sowie eine Darstellung der neuronalen Korrelate kooperativen Verhaltens. Sekundärer Endpunkt in beiden Studien ist eine Beschreibung dieses Zusammenhangs beim pathologischen Fall der ASS, sowie die Formulierung formaler Modelle zur Rolle von SMCs bei ASS-charakteristischen sozialen Defiziten.

Darstellung des bisherigen Wissensstandes

Klassische repräsentationsorientierte Zugänge zu sozialer Kognition sehen die Grundlage menschlicher Interaktion in einer „theory of mind“, d.h. in komplexen kognitiven Modellen einzelner Akteure über die Absichten und Vorstellungen anderer (Carruthers und Smith, 1996). Das vorliegende Forschungsvorhaben ist inspiriert von einer alternativen Sichtweise, die sensomotorische Interaktionsmuster (oder sensomotorische Kontingenzen, SMCs) als die Basis sozialer Interaktion und dynamischer sozialer Kopplung sieht (Di Paolo und De Jaegher, 2012). Die Relevanz von SMCs für kognitive Prozesse ist bereits gut dokumentiert (O'Regan und Noë, 2001; Engel et al., 2013), ebenso eine Erweiterung des Modells für die Definition von Objekt-Konzepten sowie die Planung von Handlungen (Bergström et al, 2011; Maye und Engel, 2013; Sánchez-Fibla et al., 2013). Die zentrale Annahme des vorliegenden Vorhabens ist, dass das Erlernen und Beherrschen von SMCs auch kritisch ist für die Vorhersage der Handlungskonsequenzen anderer Akteure und damit effektive soziale Kopplung erst ermöglicht. Das hier vorgeschlagene Modell sozial relevanter SMCs (socSMCs) hat konzeptuelle Gemeinsamkeiten sowohl mit interaktionistischen Zugängen zu sozialer Kognition (De Jaegher et al., 2010; Di Paolo und De Jaegher, 2012), als auch dem „joint action“ Modell von Knoblich und Sebanz (2008). Arbeiten zu den neuronalen Grundlagen sozialer Kognition fokussieren v.a. auf die Fähigkeit des Gehirns, die Handlungen anderer zu spiegeln (Rizzolatti und Fabbri-Destro, 2008) und gehen der Frage nach, welche kortikalen Regionen Imitation oder Empathie vermitteln (Stanley und Adolphs, 2013). Im Gegensatz zur Mehrzahl der verfügbaren Studien aus den sozialen Neurowissenschaften wird im aktuellen Projekt der Schwerpunkt auf basale Interaktionen im Bereich der Sensomotorik gelegt und deren Potenzial untersucht, neuronales Entrainment zu ermöglichen. In diesem Zusammenhang ist die Erforschung von Synchronizität und Kopplung oszillatorischer Signale sowohl innerhalb eines Gehirns (Engel et al., 2001, 2013; Hipp et al., 2011), als auch zwischen Gehirnen (Hasson et al., 2012) von Interesse. Basierend auf vorhergehenden Studien zu EEG hyperscanning (Sänger et al., 2013) erwarten wir, dass höhere soziale Kopplung zwischen zwei Akteuren in einem Geschicklichkeitsspiel sich in gesteigerter neuronaler Synchronizität zwischen den Gehirnen widerspiegelt. Obwohl hinsichtlich der Pathogenese der ASS unterschiedliche Modelle existieren, vertreten doch einige Autoren die Ansicht, dass erfahrungsabhängige synaptische Plastizität und Konnektivität von zentraler Bedeutung sind (Rinaldi et al., 2008). Bildgebungsstudien zeigen veränderte funktionelle Kopplung bzw. Konnektivitätsmuster bei Probanden mit ASS (Müller et al., 2011; Hamamy et al., 2015). Allerdings gibt es kaum EEG/MEG Evidenz zu Synchronisationsmechanismen während sozialer Prozesse bzw. sozialer Kognition, weder im neurotypischen noch im autistischen Gehirn. Noch seltener sind so genannte duale interaktive Settings oder "online" Paradigmen in den sozialen Neurowissenschaften (Timmermans & Schillbach, 2014; Tanabe et al., 2012). Darüber hinaus sind motorische Defizite bei Patienten mit ASS zwar gut dokumentiert (Chukoskie et al., 2013), aber die Modulation sozialer Interaktion durch sensomotorisches Entrainment stellt einen neuartigen und viel versprechenden Zugang dar.

Studienplanung und -durchführung

Studiendesign und Studienablauf

Es sollen 2 Studien durchgeführt werden, bestehend aus 3 (Studie 1) bzw. 2 (Studie 2) Teilexperimenten (siehe auch Tabelle 1). Die neurophysiologischen Messungen (dies betrifft

3 von 5 Teilexperimenten) werden mit Hilfe der Elektroenzephalographie (EEG) durchgeführt. In Teilexperiment 1 (Studie 1) werden Verhaltensparameter (Geschwindigkeit und Genauigkeit der Leistung in einem Geschicklichkeitsspiel), sowie Bewegungsdaten (Augenbewegungen, elektromyographische (EMG) Aktivität der Arme und des

Studienname	**Teilexperiment Nr.**	**Probanden**	**Anzahl Probanden-paare**	**Paradigma**	**Methode**	**Anzahl der Messungen**	**Untersuchungsdauer (h)**
Studie 1 Neuronale und sensomotorische Basis von sozialer Kopplung	1	Gesunde Probanden	25	Interaktives Tablet-Geschicklichkeitsspiel	Verhalten, EMG	8	0,75
	2	Gesunde Probanden	20	Interaktives Tablet-Geschicklichkeitsspiel	Verhalten, EMG, EEG hyperscanning	1	2,5
	3a	Gesunde Probanden	3 x 25	Interaktives Imitations-spiel	Verhalten, Bewegung, mobiles EEG	1	2
	3b	Patienten mit ASS, Gesunde Kontroll-Probanden (parallelisiert)	25	Interaktives Imitations-spiel	Verhalten, Bewegung, mobiles EEG	1	2,5
Studie 2 Soziale Salienz und interpersonale Kopplung bei ASS	4	Gesunde Probanden	25	Virtuelle Version des Tablet-Geschicklich-keitsspiels	Verhalten	1	2
	5	Patienten mit ASS, Gesunde Kontroll-Probanden (parallelisiert)	2 x 25	Virtuelle Version des Tablet-Geschicklich-keitsspiels	Verhalten, EEG hyperscanning	1	3,5

Tabelle 1. Geplante Teilexperimente (Ein- und Ausschlusskriterien für Probanden siehe unten).

des Rumpfes) von 25 Versuchspersonen-Paaren (gesunde Probanden ohne psychiatrische oder neurologische Vorgeschichte, 50 Personen insgesamt) erhoben. Vorgesehen sind 8 Trainingssitzungen à ca. 45 Minuten (plus 60 Minuten Aufklärung und Vorbereitung vor der ersten Messung), verteilt auf 4 Tage (also 2 Trainingseinheiten täglich). Teilexperiment 2 (Studie 1) sieht die simultane EEG-Aufzeichnung an 2 Probanden vor („EEG hyperscanning“). Mindestens 20 im ersten Teilexperiment trainierte Paare (gesunde Probanden) sollen untersucht werden. In Teilexperiment 3a werden 25 gesunde Probanden-Paare mittels Bewegungstracking untersucht. Die einmalige Sitzung dauert etwa 2 Stunden, inklusive Aufklärung und Vorbereitung. Teilexperiment 3b wiederholt diesen Aufbau, mit dem Unterschied, dass 25 Personen mit der Diagnose ASS, sowie 25 gesunde Probanden, parallelisiert in Geschlecht, Alter und Bildungsstand untersucht werden. Zusätzlich zum Bewegungstracking setzen wir hier außerdem mobiles EEG ein. Für diese Messung ist mit einer Dauer von insgesamt 2,5 Stunden zu rechnen.

In Studie 2 ist ein Verhaltensexperiment (Teilexperiment 4) mit 25 Paaren gesunder Probanden geplant. Inklusive Vor- und Nachbereitung des Experiments sind Sitzungen mit einer Dauer von jeweils 2 Stunden vorgesehen. In Teilexperiment 5 sollen außerdem neurophysiologische Messungen mittels EEG durchgeführt werden. Die Daten von 25 Personen mit Diagnose ASS, sowie 25 gesunden Kontrollprobanden (parallelisiert im Hinblick auf Geschlecht, Alter und Bildungsstand) sollen erhoben werden, sowie eine Kontrollkohorte von 25 Paaren, die nur aus gesunden Probanden besteht. Für den einzelnen Probanden ergibt sich in Teilexperiment 5 eine Gesamtdauer von ca. 3,5 Stunden (inklusive Aufklärung und Vor- und Nachbereitung der EEG Messung).

Für Teilexperimente 1 und 2 aus Studie 1 ist ein interaktives Setting geplant, in dem 2 Probanden gemeinsam ein Geschicklichkeitsspiel auf einem Tablet (iPad) spielen. Das Spiel ist eine simplifizierte Variante des so genannten Labyrinth-Spiels, wobei jeder der Probanden eine der beiden Achsen mit der rechten Hand kontrolliert. Ziel ist es, durch koordiniertes Manipulieren des Spielgeräts eine auf dem Bildschirm zu sehende Kugel an einer bestimmten, im Laufe des Spiels variierenden Position (gekennzeichnet durch einen farbigen Kreis) zu halten. Zusätzlich ist eine zweite Variante des Spiels vorgesehen, in der jeder Spieler eine Kugel kontrolliert, die sich ausschließlich entlang der eigenen Achse bewegen lässt. Während in Variante 1 nur koordiniertes Verhalten zur Zielerreichung führt (die Kugel an einer bestimmten Position zu halten), ist bei Variante 2 das Spielverhalten der beiden Akteure unabhängig und das Spiel damit per definitionem nicht-kooperativ. Für Teilexperiment 3 aus Studie 1 planen wir ein 2-Personen Imitationsspiel. Hier ist die Aufgabe der Probanden eine gemeinsame (gespiegelte) Arm/Hand-Bewegung entweder anzuführen, oder dem Interaktionspartner in der Bewegung möglichst zeitnah zu folgen. Variiert wird außerdem die Modalität des verfügbaren Feedbacks: Auditorisch (Sonifikation der Bewegung), haptisch und/oder visuell. Die Teilexperimente in Studie 2 verwenden eine virtuelle Variante des Tablet-Spiels (nur die kooperative Version kommt hier zum Einsatz), in der die Probanden durch Fingerbewegungen eine Achse der Spiel-Fläche kontrollieren und die andere experimentell manipuliert werden kann. Im Verhaltens- wie auch im EEG-Teilexperiment wird den Probanden das Spielfeld inklusive Kugel auf einem Computerbildschirm präsentiert.

Für die Auswertung der EEG-Daten benötigen wir zusätzlich Strukturaufnahmen des Gehirns jeder Versuchsperson, die mit Hilfe der Magnetresonanz-Tomographie (MRT) durchgeführt werden. Diese sind notwendig, um die anatomischen Quellen der EEG-Resultate im Gehirn zu lokalisieren. Die MRT-Messungen werden in einer separaten Sitzung von 0.5 Stunden Dauer durchgeführt.

Die beiden geplanten Studien lassen sich wie folgt zusammenfassen:

Studie 1: Neuronale und sensomotorische Basis von sozialer Kopplung (100 Teilnehmer). Die Studie besteht aus Teilexperimenten 1-3. In Teilexperiment 1 sollen Verhaltens- und EMG- Daten von 25 Versuchspersonen-Paaren (also 50 Probanden) aufgezeichnet werden. In insgesamt 8 Trainingssitzungen à 45 Minuten soll kooperatives wie nicht-kooperatives Verhalten von 2 Personen in einem Tablet-Geschicklichkeitsspiel untersucht werden. Ziel des Spiels ist es, über koordinierte Bewegung der beiden Achsen des Tablets eine auf dem Bildschirm zu sehende Kugel (respektive zwei unabhängig zu kontrollierende Kugeln) an einer bestimmten Position zu halten. SMCs sollen so in einem sozialen Setting erlernt und ihr Zusammenhang zu sozialer Kopplung zwischen den beiden Spielern untersucht werden. Dazu beurteilen beide Akteure nach jeder Spiel-Runde die Qualität der Interaktion sowie ihr subjektiv empfundenes Involviert-Sein in die gemeinsame Aufgabe auf einer visuellen Analogskala. Zusätzlich werden Spielparameter ausgelesen (z.B. die Entfernung der Kugel vom Ziel) und mit den physiologischen (z.B. EMG-) Parametern bzw. den Ratings in Zusammenhang gesetzt. Die zentrale Annahme hierbei lautet, dass engere sensomotorische Kopplung der beiden Spieler (z.B. abgebildet durch höhere Korrelation der EMG- Signaturen) assoziiert ist mit höherer sozialer Kopplung. In Teilexperiment 2 sollen die hirnphysiologischen Korrelate kooperativen Verhaltens untersucht werden. Anhand verschiedener Maße zur Analyse von Phasenbeziehungen neuronaler Signale (in diesem Fall EEG-Signale) soll geprüft werden, inwiefern sensomotorische Kopplung mit neuronaler Kopplung einhergeht. Es wird untersucht, in welchen Frequenzbereichen und Hirnregionen spezifische Veränderungen oszillatorischer

neuronaler Signale mit erfolgreicher Interaktion zusammenhängen. Mindestens 20 der im ersten Teilexperiment trainierten Paare sollen mittels EEG untersucht werden. Teilexperimente 3a und 3b sehen ein anderes Setup vor. Hier imitieren sich zwei einander gegenübersitzende Probanden in ihren Arm/Hand-Bewegungen (Bewegungstracking und mobile EEG-Aufzeichnung), unterstützt durch sensorisches Feedback in verschiedenen Modalitäten. In Teilexperiment 3a nehmen wir

3 mal 25 Datensätze von gesunden Probanden auf. Teilexperiment 3b unterscheidet sich darüber hinaus dadurch, dass einer der Spieler die Diagnose ASS hat. Durch die Analyse sensomotorischer wie neuronaler Kopplungen und den Vergleich zwischen ASS- Patienten und gesunden Kontroll-Probanden soll ein neuer Zugang zur Charakterisierung pathogenetischer Faktoren bei ASS, nämlich jener über socSMCs, erprobt werden. Für Teilexperiment 3b ist die Erhebung von 25 Datensätzen von Patienten mit ASS, sowie 25 parallelisierten Kontrollprobanden vorgesehen.

Studie 2: Soziale Salienz und interpersonale Kopplung bei ASS (75 Teilnehmer). In beiden Teilexperimenten des 2. Studienkomplexes soll eine virtuelle Version des Geschicklichkeitsspiels zur Anwendung kommen. In Teilexperiment 4 sollen 25 Paare gesunder Probanden durch Fingerbewegungen eine Achse des Spiels steuern, wieder mit dem Ziel, die Kugel an bestimmte, sich verändernde Positionen zu bringen. Visuelle Rückmeldung über die Position der Kugel wird diesmal über einen Computer-Monitor gegeben. Aufgabe der Probanden ist es zusätzlich, im Rahmen eines so genannten Turing-Tests Runde für Runde zu beurteilen, ob sie mit einem Menschen oder einer Maschine (bzw. einem Algorithmus) interagiert haben. Das Spielverhalten des (unsichtbaren) Gegenübers ist dabei entweder tatsächlich artifiziell, also auf einem vorab programmierten Algorithmus beruhend (nicht-soziale Kondition), oder kommt von einem zweiten menschlichen Akteur (der in einem anderen Raum sitzend „online“ am Spiel teilnimmt; soziale Kondition). Außerdem soll nach jeder Spiel-Runde das subjektiv empfundene Involviert-Sein in die Aufgabe und die Qualität der Interaktion über eine Rating-Skala beurteilt werden. Angenommen wird, dass die Interaktion mit einem (vermeintlich) menschlichen Partner zu einem beschleunigten sensomotorischen Lernprozess und besserer Performanz führt. In Teilexperiment 5 soll mittels EEG das neuronale Substrat dieser virtuellen sozialen Interaktion untersucht werden, und zwar an einer Stichprobe von 25 Patienten mit Diagnose ASS, sowie einer parallelisierten Kontrollgruppe von 25 gesunden Probanden. Die Vorhersage das Verhalten betreffend ist, dass ASS-Patienten weniger präzise beurteilen können, ob sie mit einem Menschen oder einer Maschine interagieren (also niedrigere Erkennungsraten im Turing Test aufweisen) und keinen Unterschied in den subjektiven Ratings zwischen Spielrunden mit menschlichen Partnern und Maschinen – also zwischen der sozialen und der nicht-sozialen Kondition – zeigen. In der Analyse der EEG-Daten wird untersucht, inwiefern sensomotorisches Lernen und Performanz in dem Geschicklichkeitsspiel assoziiert sind mit neuronaler Kopplung und welche Hirnregionen hierbei besonders relevant sind (so könnten frontozentrale und motorische Netzwerke, die Handlungsplanung und -exekution vermitteln, von besonderem Interesse sein). Wir erwarten, dass das neuronale Entrainment in der sozialen Bedingung stärker ist als in der nicht-sozialen Bedingung und dass dieser Unterschied stärker ausgeprägt ist für die gesunden Kontrollprobaden als für Patienten mit ASS.

Durchführung der EEG-Untersuchungen

In allen Untersuchungen wird den Versuchspersonen ein Aufklärungsbogen vorgelegt und es wird der Versuchsablauf besprochen. Gleichzeitig werden die Probanden darauf hingewiesen, dass die Teilnahme an diesen Untersuchungen freiwillig ist und die

Zustimmung jederzeit widerrufen werden kann. Bezüglich des Ablaufs und Inhaltes der Aufklärungsgespräche sei auf die beiliegenden Aufklärungsbögen verwiesen.

Jede Einzeluntersuchung zur EEG-Messung wird etwa 3,5 Stunden dauern. Für die EEG-Teilexperimente soll die neuronale Aktivität zweier Probanden mittels EEG hyperscanning (in Studie 1 kommen zwei 64-Kanal EEG-Systeme zur Anwendung, in Studie 2 kommen zwei 128-Kanal EEG-Systeme zur Anwendung) aufgezeichnet werden. Zusätzlich werden zur Aufzeichnung der Augenbewegungen mittels Elektrooculogramm (EOG) 2 weitere Elektroden im Gesicht der Probanden angelegt. Vor der Anbringung der Elektroden muss die Haut mit einem Alkoholtupfer gereinigt werden. Insgesamt ist für Aufklärung, Vorbereitung der Probanden und die eigentliche Messung mit einer Dauer von ca. 3,5 Stunden zu rechnen.

Risiken der Untersuchung
Mit Hilfe des EEG lassen sich elektromagnetische Aktivierungen des menschlichen Gehirns mit hoher zeitlicher Auflösung (Millisekunden-Bereich) erforschen. Nach dem aktuellen wissenschaftlichen Kenntnisstand ist dieses Verfahren nicht mit Risiken für die Versuchsteilnehmer verbunden. Die hier zur Anwendung kommende strukturelle Kernspintomographie (MRT) ermöglicht es, Unterschiede in der Gehirnstruktur bei verschiedenen Probanden mit sehr hoher räumlicher Auflösung zu untersuchen. Untersuchungen mittels MRT sind sicher und ohne bekannte Nebenwirkungen. Sie beinhalten keine Strahlenbelastung oder andere gesundheitliche Risiken und können deshalb wiederholt durchgeführt werden.

Stichprobengröße
Wir veranschlagen eine Gruppengröße von jeweils 25 Probanden (-Paaren) für jedes der Teilexperimente der beiden Studien. Dies ist notwendig, um aussagekräftige Daten zu erhalten und die Vergleichbarkeit zwischen verschiedenen Bedingungen sicherzustellen. Diese Stichprobengröße von n=25 Teilnehmern pro Teilexperiment wurde unter Zugrundelegung eines Typ 1-Fehlers von 5% und einer Power von 0.8 unter Berücksichtigung der Erfahrung aus vorherigen Studien berechnet. Wir gehen von einer Drop-Out-Rate von maximal 20% (n=5) aus.

Einschlusskriterien für gesunde Probanden in Teilstudien 1, 2, 3a und 4:
- Gesunde Probanden ohne vorherige psychiatrische Diagnose
- Alter zwischen 18 und 35 Jahren
- Deutsch als Muttersprache
- Rechtshändigkeit (überprüft mit Hilfe des Edinburgh Handedness Inventory)
- Normale Sehschärfe (evtl. mit Korrektur), normale Farbsichtigkeit
- Unterschriebene Einwilligung in die Untersuchung

Ausschlusskriterien für gesunde Probanden in Teilstudien 1, 2 und 4:
- Neurologische und schwere internistische Begleiterkrankungen
- Dauerhafte Einnahme von Medikamenten
- Alkohol oder Koffein am Tag des Experiments
- Bei Probandinnen: Schwangerschaft bzw. mögliche Schwangerschaft
- Platzangst
- Herzschrittmacher oder andere implantierte biomedizinische Geräte
- Nicht MEG- oder MRT-kompatible metallische Implantate oder Fremdkörper im oder am Körper

Einschlusskriterien für Patienten in Teilstudien 3b und 5:

Für Probanden mit ASS gültige Einschlusskriterien:
- Diagnose einer ASS nach ICD-10 oder DSM IV/ V
- Alter zwischen 18 und 50 Jahren
- Rechtshändigkeit
- Einwilligungs- und Studienfähigkeit nach Rücksprache mit dem/der diagnostizierenden Arzt/Ärztin bzw. psychologischen Psychotherapeut/in

Einschlusskriterien für gesunde Kontrollprobanden in Teilstudien 3b und 5:
- in Bezug auf Probanden mit ASS parallelisiert nach Alter, Geschlecht und Bildungsstand
- Alter zwischen 18 und 50 Jahren
- Rechtshändigkeit
- Einwilligung in die Untersuchung

Ausschlusskriterien für Patienten in Teilstudien 3b und 5:
Für Probanden mit ASS gültige Ausschlusskriterien:
- Alter < 18 oder > 50 Jahre
- neurologische und internistische Begleiterkrankungen
- schwere psychiatrische Begleiterkrankungen (z.B. schwere Depression, Suchtmittel-Abhängigkeit)
- starke Hornhautverkrümmungen (bei nicht vorhandenen Kontaktlinsen)
- Platzangst
- Schwangerschaft
- Herzschrittmacher oder andere implantierte biomedizinische Geräte
- Nicht MEG- oder MRT-kompatible metallische Implantate oder Fremdkörper im oder am Körper

Ausschlusskriterien für gesunde Kontrollprobanden in Teilstudien 3b und 5:
- Alter < 18 oder > 50 Jahre
- neurologische und internistische Begleiterkrankungen
- schwere psychiatrische Begleiterkrankungen (z.B. schwere Depression, Suchtmittel-Abhängigkeit)
- Hornhautverkrümmungen (bei nicht vorhandenen Kontaktlinsen)
- Platzangst
- Schwangerschaft
- Herzschrittmacher oder andere implantierte biomedizinische Geräte
- Nicht MEG- oder MRT-kompatible metallische Implantate oder Fremdkörper im oder am Körper
- Signifikantes Vorliegen von autistischen Zügen (erfasst mit dem Autism Spectrum Quotient)

Verwendete Fragebögen
- Probandenfragebögen zur Erfassung der Händigkeit;
- Interpersonal Reactivity Index (Dt. Version: "Saarbrücker Persönlichkeitsfragebogen SPF");
- Persönlichkeitsfragebogen (NEO-FFI);
- Fragebögen zur Erfassung von relevanten Vorerfahrungen (Musik, PC-Spiele) und Spielstrategien;
- kurze ‚Rating'-Fragen (Antwort erfolgt mittels einer Skala) während des Experiments, sowie ein offenes Interview nach dem Experiment, zur Erfassung des subjektiven

Erlebens von Zusammenarbeit, Performanz, Involviert-Sein im Spiel und Vorhersehbarkeit der Handlungen des Spielpartners;

- für MRT-Untersuchung Fragebogen zur Erfassung von Ausschlusskriterien;
- Tests und Fragebögen zur Charakterisierung der Psychopathologie bei ASS im Vergleich zu gesunden Probanden (*Auswahl*: Wortschatztest (Schmidt und Metzler); Becks Depression Inventar II; Autismus Quotient & Empathie-Quotient (AQ, EQ; Freitag et al., 2006; Baron-Cohen & Weelwright, 2004)).

Literaturangaben

Bergström N, Björkman M, Kragic D (2011) Generating Object Hypotheses in Natural Scenes through Human-Robot Interaction, In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA.

Carruthers P., Smith P.K. (1996) Theories of theories of mind. Cambridge University Press.

Chukoskie, L., Townsend, J., & Westerfield, M. (2013). Motor skill in autism spectrum disorders: a subcortical view. International Review of Neurobiology, 113, 207–249.

De Jaegher, H., Di Paolo, E., & Gallagher, S. (2010). Can social interaction constitute social cognition? Trends in Cognitive Sciences, 14(10), 441–447.

Di Paolo, E., & De Jaegher, H. (2012). The interactive brain hypothesis. Frontiers in Human Neuroscience, 6, 163.

Engel, A.K., Fries, P., and Singer, W. (2001). Dynamic predictions: oscillations and synchrony in top-down processing. Nature Reviews Neuroscience 2, 704–716.

Engel, A. K., Maye, A., Kurthen, M., & König, P. (2013). Where's the action? The pragmatic turn in cognitive science. Trends in Cognitive Sciences, 17, 202–209.

Hahamy, A., Behrmann, M., & Malach, R. (2015). The idiosyncratic brain: distortion of spontaneous connectivity patterns in autism spectrum disorder. Nature Neuroscience, 18, 302–309.

Hasson, U., Ghazanfar, A. A., Galantucci, B., Garrod, S., & Keysers, C. (2012). Brain-to-brain coupling: a mechanism for creating and sharing a social world. Trends in Cognitive Sciences, 16, 114–121.

Hipp, J.F., Engel, A.K., and Siegel, M. (2011). Oscillatory synchronization in large-scale cortical networks predicts perception. Neuron 69, 387–396.

Knoblich, G., & Sebanz, N. (2008). Evolving intentions for social interaction: from entrainment to joint action. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, 363, 2021–2031.

Krampen, G. (1981). Persönlichkeitsfragebogen zur Messung des Konstrukts *locus of control* (generalisierte *Kontrollüberzeugung*). Handanweisung. Göttingen: Hogrefe.

Maye A., Engel, A.K. (2013) Extending sensorimotor contingency theory: Prediction, planning, and action generation. Adaptive Behavior 21, 6.

Müller, R.-A., Shih, P., Keehn, B., Deyoe, J. R., Leyden, K. M., & Shukla, D. K. (2011). Underconnected, but how? A survey of functional connectivity MRI studies in autism spectrum disorders. Cerebral Cortex, 21, 2233–2243.

O'Regan, J. K., & Noë, A. (2001). A sensorimotor account of vision and visual consciousness. The Behavioral and Brain Sciences, 24, 939–973; discussion 973–1031.

Rinaldi, T., Perrodin, C., & Markram, H. (2008). Hyper-connectivity and hyper-plasticity in the medial prefrontal cortex in the valproic Acid animal model of autism. Frontiers in Neural Circuits, 2, 4.

Rizzolatti, G., & Fabbri-Destro, M. (2008). The mirror system and its role in social cognition. Current Opinion in Neurobiology, 18, 179–184.

Sánchez-Fibla M, Duff A, Verschure PFMJ (2013) A sensorimotor account of visual and tactile integration for object categorization and grasping IEEE/RSJ International Conference on Robotics and Automation (ICRA): 107-112.

Sänger, J., Müller, V., & Lindenberger, U. (2013). Directionality in hyperbrain networks discriminates between leaders and followers in guitar duets. Frontiers in Human Neuroscience, 7, 234.

Stanley, D. A., & Adolphs, R. (2013). Toward a neural basis for social behavior. Neuron, 80, 816–826.

Tanabe, H. C., Kosaka, H., Saito, D. N., Koike, T., Hayashi, M. J., Izuma, K., … Sadato, N. (2012). Hard to "tune in": neural mechanisms of live face-to-face interaction with high-functioning autistic spectrum disorder. Frontiers in Human Neuroscience, 6, 268.

Timmermans, B., & Schilbach, L. (2014). Investigating alterations of social interaction in psychiatric disorders with dual interactive eye tracking and virtual faces. Frontiers in Human Neuroscience, 8, 758.

_____

Prof. Dr. Andreas K. Engel, Institutsdirektor

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.
Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den _____ Unterschrift: _____