Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Mikroelektronik und Datentechnik

in Zusammenarbeit mit dem

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Forschungsbereich Robotik
Bremen

Master-Arbeit

Minimierung und Untersuchung des Simulation-Reality-Gaps anhand eines BRIO®-Labyrinth-Spiels

Constantin Bergatt
Matrikelnummer: 3201353

Betreuer:
Prof. Dr.-Ing. Ralf Salomon (Universität Rostock)
Prof. Dr. rer. nat. Frank Kirchner (DFKI Bremen)
Dipl.-Inf. Jan Hendrik Metzen (DFKI Bremen)

März 2009

Danksagung

An dieser Stelle möchte ich die Gelegenheit ergreifen und mich bei einigen Personen bedanken.

Ich danke Prof. Dr.-Ing. Ralf Salomon und Prof. Dr. rer. nat. Frank Kirchner für die Vergabe dieser Arbeit und richtungsweisenden Hilfestellungen.
Weiterhin bedanke ich mich bei meinem Betreuer, Dipl.-Inf. Jan Hendrik Metzen, für seine stets konstruktive Kritik und seine vielen nützlichen Denkanstöße.
Ebenfalls möchte ich Dipl.-Biol. Elsa Kirchner für ihre organisatorische Betreuung meiner Arbeit danken.
Auch den Mitarbeitern des DFKI Bremen gebührt mein Dank für ihre guten Ratschläge und die praktische Hilfe.
Zu guter Letzt möchte ich mich bei meiner Familie für ihren Rückhalt auch in schwierigen Zeiten bedanken, insbesondere bei meinem Bruder für die vielen Stunden Korrekturarbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis **IV**

Tabellenverzeichnis **VIII**

1 Einleitung **1**

1.1 Motivation 1
1.2 Simulation-Reality-Gap 2
1.3 Struktur der Arbeit 3

2 State of the Art **6**

2.1 Verfahren zum Minimieren des Simulations-Reality-Gaps 6
2.1.1 Modellierung der Realität basierend auf Sensoren und Aktuatoren des Roboters 6
2.1.2 Hinzufügen von Rauschen zu den Sensor- und Aktuatorwerten der Simulation 7
2.1.3 Envelope-of-Noise 8
2.1.4 Estimation-Exploration-Algorithmus 10
2.2 Verfahren zum Überbrücken des Simulations-Reality-Gaps 12
2.2.1 Fortsetzen des Trainings auf dem realen System 12
2.2.2 Evolution des Lernens 12

3 Grundlagen **14**

3.1 BRIO®-Labyrinth-Spiel 14
3.2 BRIO®-Testbed 15
3.2.1 Überblick 15
3.2.2 Bauelemente 17
3.2.3 Verfahren und Methoden des BRIO®-Testbeds 21
3.3 Simulation des BRIO®-Testbeds 26
3.3.1 Machinae Arte Robotum Simulans - MARS 27
3.3.2 Realisierung des BRIO®-Labyrinth-Spiels in MARS 27

4 Konzept der Arbeit **29**

5 Untersuchung der gemessenen Brettwinkel **32**
5.1 Untersuchung des Rauschens der Potentiometerspannungen 33
5.2 Rauschen der Brettwinkel 37

6 Untersuchung der angefahrenen Brettwinkel **40**
6.1 Untersuchung der Positioniergenauigkeit der Servomotoren 40
6.2 Kompensation des Positionierungsfehlers der Servomotoren 41
6.2.1 Einfluss der Servoparameter auf die Positioniergenauigkeit 43
6.2.2 Anpassung der Zielposition unter Berücksichtigung des lokalen Positionierungsfehlers der Servomotoren 49
6.3 Abbildung der Servopositionen auf die Winkel des Spielbretts . . 50
6.3.1 Einfluss des Spiels der Gummiriemen 52
6.3.2 Kompensation des Spiels der Gummiriemen 52
6.4 Fehler und Rauschen der angefahrenen Brettwinkel 54

7 Untersuchung der Bestimmung der Kugelposition **57**
7.1 Rauschen der Pixelkoordinaten der detektierten Kugelposition . . 57
7.2 Abhängigkeit der Bestimmung der Pixelkoordinaten von der Bildhelligkeit 58
7.3 Kompensation der Verzeichnung der Kamera 60
7.4 Koordinatentransformation der Brettkoordinaten 64
7.4.1 Umrechnung der Kugelkoordinaten der Simulation 65
7.4.2 Umrechnung der Kugelkoordinaten des BRIO®-Test-beds . 66
7.5 Untersuchung des Fehlers der Kugelposition 69
7.5.1 Untersuchung des Fehlers der Kugelposition bei unausgelenktem Spielbrett 70
7.5.2 Untersuchung des Einflusses der Brettwinkel auf den Fehler der Kugelposition 72
7.6 Modellieren des Rauschens der Kugelposition 73

8 Optimierung der Motorparameter der BRIO®-Simulation **76**
8.1 Manuelle Optimierung der Motorparameter 78
8.2 Optimierung der Motorparameter mittels Evolutionsstrategie . . . 79

9 Vergleich des Verhaltens beider Systeme **83**
9.1 Epsilon-Tunnel 83
9.2 Aktionsfolge 86
9.3 Untersuchung der Anpassungen der Aktuatoren 87

9.4 Untersuchung der Anpassungen der Bestimmung der Kugelposition 90

10 Ergebnisse **95**

10.1 Zusammenfassung 95

10.2 Ausblick 97

Anhang **99**

A Boxplot **99**

Literaturverzeichnis **100**

Abbildungsverzeichnis

1.1 Simulation-Reality-Gap 3

2.1 Ablauf der drei Phasen des Estimation-Exploration-Algorithmus . 11

3.1 Original BRIO®-Labyrinth-Spiel 14
3.2 Aufbau des Spielbretts 15
3.3 BRIO®-Testbed 16
3.4 Aufbau des BRIO®-Testbeds 16
3.5 Übertragung der Bewegungen der Servomotoren mittels Gummiriemen auf die Achsen des Spielbretts 17
3.6 Drehbereich eines DX-117 Servomotors [21] 18
3.7 RedCOM USB-COMi 19
3.8 Anbringung des MEGATRON MCP50 Potentiometers an der Achse des BRIO®-Spiels 20
3.9 Datenerfassungsmodul (a) und 68-Pin Anschlussblock (b) der Firma National Instruments 20
3.10 PROSILICA GC640C 21
3.11 Intel PRO-1000 GT Desktop Adapter 21
3.12 Berechnung der Brettwinkel ϕ_X und ϕ_Y der *Drehachsen des Spielbretts* basierend auf den gemessenen Potentiometerspannungen U_{AI0} und U_{AI1} der *Achsen des BRIO®-Labyrinth-Spiels* 22
3.13 Grobe Darstellung des Ablaufs der Detektion der Kugelposition (*Erläuterung:* (a) Kamerabild; (b) geglättetes Bild; (c) Bestimmung des hellsten Pixels; (c) Verfeinerung der Kugelposition; (e) detektierte Kugelposition) 24
3.14 Detektion der vier Ecken des Spielbretts mittels eines Harris-Corner-Detectors [6] als Referenzpunkte zur Umrechnung der Pixelkoordinaten des Kamerabildes in Koordinaten bezüglich des Spielbretts . 25
3.15 Alte (a) und neue (b) BRIO®-Simulations-Umgebung 28

5.1 Boxplot der Verteilung der Potentiometerspannungen aufgetragen über die relativen Soll-Positionen der Servomotoren für $AI0$ (a) bzw. $AI1$ (b) 34
5.2 Boxplot der Verteilung der gemessenen Spannungen über den Potentiometern $AI0$ und $AI1$ um den Median der einzelnen Messreihen (a) und (c) bzw. aller Messreihen zusammen (b) und (d) . . . 35
5.3 Histogramme des Rauschens der Potentiometerspannungen für $AI0$ (a) und $AI1$ (b) (*Erklärung:* Die rote Linie repräsentiert dabei die kumulative Verteilungsfunktion des jeweiligen Servomotors. Mit ihrer Hilfe kann die Verteilung der Spannungswerte nachgebildet werden. Dazu wird zunächst eine gleichverteilte Zufallszahl aus dem Intervall [0,1] gezogen und anschließend deren Wert auf den zugehörigen Spannungswert abgebildet (schwarze gestrichelte Linie).) . 36

6.1 Boxplots des Positionierungsfehler der Servomotoren $S2$ (a) und $S3$ (b) aufgetragen über die einzelnen Testpositionen des jeweiligen Servomotors 42
6.2 Darstellung der mittleren Positionierungsfehler des Servomotors $S2$ und $S3$ aufgetragen über die zugehörigen Soll-Positionen . . . 43
6.3 Zusammenwirken der Parameter Compliance Margin, Compliance Slope und Punch [21] 44
6.4 Vergleich des mittleren absoluten Positionierungsfehlers der beiden Servomotoren in Abhängigkeit vom Wert des Parameters Compliance Margin 45
6.5 Vergleich des mittleren absoluten Positionierungsfehlers der beiden Servomotoren in Abhängigkeit vom Wert des Parameters Compliance Slope 47
6.6 Vergleich des mittleren absoluten Positionierungsfehlers der beiden Servomotoren in Abhängigkeit vom Wert des Parameters Punch . 48
6.7 Vergleich des mittleren Positionierungsfehler des Servomotors $S2$ (a) bzw. $S3$ (b) aufgetragen über die zugehörigen Soll-Positionen . 51
6.8 Gegenüberstellung der gemessenen Brettwinkel ϕ_X bzw. ϕ_Y und der angefahrenen Ist-Position des Servomotors $S2$ (a) bzw. $S3$ (b) 53

7.1 Testaufbau zur Bestimmung des Rauschen der Kugelposition . . . 58
7.2 Unterschiedliche Helligkeitsverhältnisse 59
7.3 Standardabweichung der x- und y-Werte der Pixelkoordinaten der detektierten Kugel in Abhängigkeit von der mittleren Helligkeit des Kamerabildes 60

7.4 Verzeichnungsarten [17] 61
7.5 Kalibrierungs-Muster [2] 62
7.6 Vergleich des aufgenommenen Bildes (a) und (c) mit dem Bild nach der Kompensation der Kameraverzeichnung (b) und (d) 63
7.7 Einfluss des Brettwinkels auf die Kugelposition 64
7.8 Grafische Darstellung des gemeinsamen Koordinatensystems anhand des BRIO®-Testbeds 65
7.9 Berechnung der dreidimensionalen Koordinaten der Kugelposition als Schnittpunkt der Geraden durch die detektierte Kugelposition mit der ausgelenkten Spielebene 68
7.10 Testpositionen zur Bestimmung des Fehlers der Kugeldetektion des BRIO®-Testbeds 70
7.11 Zusammenhang zwischen den Drehrichtungen der Servomotoren $S2$ und $S3$ und den Brettwinkeln ϕ_X und ϕ_Y des BRIO®-Testbeds 72

8.1 Darstellung der zeitlichen Verzögerung *delay* 77
8.2 Beispielhafte Darstellung der Messreihe für eine Drehbewegung des Servomotors $S2$ von $0°$ auf $5°$ 78
8.3 Vergleich der zeitlichen Verläufe der Mittelwerte einer Messreihe mit den Brettwinkeln der Simulation bei manuell optimierten Parametern 79
8.4 Darstellung des Integrals E der Abweichungen der simulierten Brettwinkel ϕ^S von den realen Brettwinkeln ϕ^R für Zeitschritte $\triangle t$ von $100ms$ 81
8.5 Vergleich der realen, der manuell und der automatisch optimierten Motorkurven, beispielhaft für den Servomotor $S2$ 82

9.1 Beispielhafte Darstellung des *Epsilon-Tunnels* für $k = 1$ (a) und $k = 3$ (b) (*Erläuterung:* Die blaue Linie repräsentiert den zeitlichen Verlauf des gemittelten Brettwinkels ϕ_X des BRIO®-Testbeds, die beiden roten Linien sind die obere und untere Grenze des Epsilon-Tunnels des Brettwinkels ϕ_X und die grüne Linie stellt den Verlauf des Brettwinkels ϕ_X der Simulation dar.) 85
9.2 Darstellung der Aktionsfolge auf dem gesamten Spielbrett und in der Detailansicht 86

9.3 Positionierungsfehler und zeitlicher Versatz der Brettwinkel zwischen realem und simuliertem System (*Erläuterung:* Die blaue Linie repräsentiert den zeitlichen Verlauf des gemittelten Brettwinkels ϕ_Y des BRIO®-Testbeds, die beiden roten Linien sind die obere und untere Grenze des Epsilon-Tunnels des Brettwinkels ϕ_Y und die grüne Linie stellt den Verlauf des Brettwinkels ϕ_Y der Simulation dar.) 88

9.4 Vergleich der Trajektorien der Koordinaten der Simulation sowie des BRIO®-Testbeds (*Erläuterung:* Zu erkennen ist neben einer generellen Abweichung der Kugelpositionen eine zeitliche Verzögerung des BRIO®-Testbeds gegenüber der Simulation sowie eine Verzerrung des Verlaufs der Trajektorie des realen Systems. Die blaue Linie repräsentiert den zeitlichen Verlauf der jeweiligen Kugelkoordinate des BRIO®-Testbeds, die beiden roten Linien sind die obere und untere Grenze des Epsilon-Tunnels und die grüne Linie stellt den Verlauf der Kugelkoordinate der Simulation dar.) . 91

A.1 Darstellung der einzelnen Komponenten eines Boxplots 99

Tabellenverzeichnis

3.1 Standardparameter der Servomotoren DX-117 18

5.1 Kleinste und größte erlaubte Position sowie Nullposition der Servomotoren 33
5.2 Auftretenswahrscheinlichkeiten der Spannungswerte des Rauschens der Potentiometer $AI0$ und $AI1$ um ihren Median 37
5.3 Gemittelte und gerundete Referenzspannungen der Potentiometer 38

6.1 Kleinste, mittlere und größte erlaubte Position der beiden Servomotoren 52
6.2 Vergleich des Mittelwerts und der Standardabweichung der Differenz zwischen dem vorgegebenen Ziel- und dem angefahrenen Brettwinkel ϕ_X 55
6.3 Vergleich des Mittelwerts und der Standardabweichung der Differenz zwischen dem vorgegebenen Ziel- und dem angefahrenen Brettwinkel ϕ_Y 56

7.1 Mittlere Abweichung der x-Koordinaten der vom BRIO®-Testbed ermittelten Kugelpositionen von der gemessenen Kugelposition für verschiedene Kugelpositionen und bei Spielbrett in Nullposition . 71
7.2 Mittlere Abweichung der y-Koordinaten der vom BRIO®-Testbed ermittelten Kugelpositionen von der gemessenen Kugelposition für verschiedene Kugelpositionen und bei Spielbrett in Nullposition . 71
7.3 Vergleich der Abweichungen der x-Koordinate der mit den verschiedenen Verfahren detektierten Kugelposition für verschiedene Auslenkungen des Spielbretts 74
7.4 Vergleich der Abweichungen der y-Koordinate der mit den verschiedenen Verfahren detektierten Kugelposition für verschiedene Auslenkungen des Spielbretts 75

8.1 Automatisch optimierte Motorparameter der beiden simulierten Motoren 81

9.1 Verwendete Sätze von Motorparametern der Simulation 87
9.2 Vergleich der Überdeckung des Brettwinkels ϕ_Y der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$ 88
9.3 Vergleich der Überdeckung des Brettwinkels ϕ_Y der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$ 89
9.4 Vergleich der Überdeckung der x-Koordinate der Ballposition der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$ 93
9.5 Vergleich der Überdeckung der y-Koordinate der Ballposition der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$ 93

Kapitel 1

Einleitung

1.1 Motivation

Die Methoden und Verfahren der Künstlichen Intelligenz finden immer breitere Anwendung und werden für zunehmend komplexer werdende Aufgaben eingesetzt. Besonders faszinierend ist das Zusammenwirken von Agenten und „echter“ Intelligenz in der realen Welt, der Robotik. Roboter werden ebenfalls in immer komplexer werdenden Szenarien eingesetzt und müssen dabei eine Fülle von äußeren Einflüssen in ihren Planungen berücksichtigen.
Zusätzlich soll der Roboter auf Änderungen der Umgebungsbedingungen selbstständig reagieren und sein Verhalten bzw. seine Vorgehensweise entsprechend anpassen können. Aus diesem Grund sind vom Programmierer fest vorgegebene Verhaltensweisen wie starre Kontrollstrukturen ungeeignet, da diese eine kosten- und zeitaufwendige manuelle Anpassung bei Änderung der Umgebungsbedingungen erforderlich machen. Adaptive Verfahren, wie z.B. maschinelle Lernverfahren, sind daher für den Einsatz zur Steuerung von Robotern besser geeignet. Sie benötigen jedoch im Vorhinein eine Trainingsphase in der die Parameter bzw. die Struktur des eingesetzten Verfahrens für die gegebene Aufgabe und Umweltbedingungen bestimmt werden. Das Training des Roboters stellt somit eine besondere Herausforderung dar.
Oftmals werden Roboter für Umgebungsbedingungen, wie z.B. das Weltall oder die Tiefsee, konstruiert, welche zum Trainieren des Roboters nur schwer bzw. extrem aufwendig nachgebildet werden können.
Ein alternativer Ansatz ist es, den Roboter innerhalb einer digital simulierten Umgebung zu trainieren. Vorteile dieser Herangehensweise sind das Training des Roboters unter immer gleichbleibenden Umgebungsbedingungen, die Möglichkeit der Automatisierung des Trainingsprozess, eine Verkürzung der benötigten Trainingszeit, da die Berechnungen der Simulation unter Umständen schneller als

Echtzeit ausgeführt werden können, sowie kein Verschleiß des Roboters während der Trainingsphase.
Häufig hat sich gezeigt, dass sich ein mittels Simulation trainierter Roboter sich unter realen Bedingungen anders verhält als in der Simulation [14][20]. Dies liegt meist darin begründet, dass in der Simulation ideale Bedingungen vorherrschen. Rauschen, Fehler bzw. zeitliche Verzögerungen in der Positionierung der Aktuatoren bzw. in der Sensorik des realen Systems, ungenaue Berücksichtigung der physikalischen Zusammenhänge innerhalb der Simulation, ..., usw. sind Faktoren, welche die Realität von der Simulation unterscheiden [8][10]. Dabei können diese Faktoren im Zusammenspiel die Umgebungsbedingungen stark beeinflussen. Folglich unterscheiden sich die Bedingungen der Simulation und der Realität, so dass der Roboter mit Hilfe des in der Simulation gelernten Verhaltens eventuell nur eingeschränkt seine Aufgabe in der realen Welt erfüllen kann.

1.2 Simulation-Reality-Gap

Im Rahmen dieser Master-Arbeit wird diese Diskrepanz zwischen dem Verhalten des Roboters innerhalb der Simulation und dem Verhalten des Roboters in der Realität als *Simulation-Reality-Gap* bezeichnet. Hierbei ist der Abfall der Performance des Agenten nach dem Übertragen auf den realen Roboter nicht auf einen Fehler der entwickelten Verhaltensstrategie zurückzuführen, sondern eher auf eine Änderung der Umgebungsdynamik des realen Roboters gegenüber der Simulation.

$$ideal: \; S^{Real} = S^{Sim} \tag{1.1}$$

Im Idealfall sollte sich die Simulation vollkommen identisch zum realen System verhalten (Abbildung 1.1). D.h. befinden sich beide Systeme zum Startzeitpunkt im gleichen Zustand und wird eine Aktion A sowohl auf dem realen System als auch innerhalb der Simulation ausgeführt, so sind die Folgezustände des realen Systems S^{Real} wie auch die Zustände der Simulation S^{Sim} jederzeit vollkommen identisch (1.1).
Dieser Fall ist allerdings nur schwer erreichbar. Zum Einen rauschen reale Bauteile, reagieren mit einer zeitlichen Verzögerung, besitzen eine gewisse Messunsicherheit und beeinflussen die Zustände des realen Systems entsprechend. Zum anderen können Simulationen nur sehr schwer die eigentliche Dynamik der realen Welt nachbilden. Aufgrund des enormen Rechenaufwandes und der hohen Komplexität einer vollständigen Nachbildung der Realität, werden in den meisten Simulationen Vereinfachungen bzw. Approximationen eingesetzt und somit

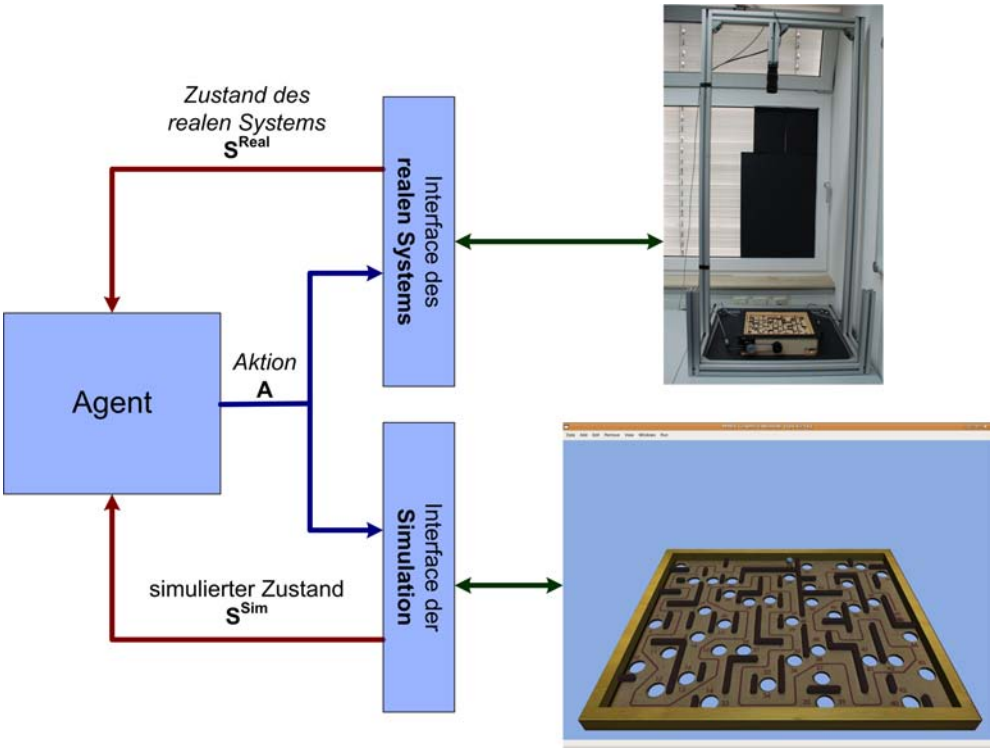


Abbildung 1.1: Simulation-Reality-Gap

der Rechenaufwand sowie die Komplexität der Simulation reduziert.

$$real: \ |S^{Real} - S^{Sim}| = \epsilon \tag{1.2}$$

Folglich ist die Bedingung (1.1) bzw. eine perfekte Nachbildung eines realen Systems innerhalb einer Simulation nur sehr schwer zu erreichen. Die einzelnen Zustände des realen Systems S^{Real} und die Zustände der Simulation S^{Sim} werden voneinander abweichen. Diese Abweichung ϵ wird in dieser Arbeit als Simulations-Reality-Gap bezeichnet und entsprechend (1.2) definiert.

1.3 Struktur der Arbeit

Die Arbeit ist in folgende Abschnitte gegliedert:

- In Kapitel 2 erfolgt zunächst ein kurzer Überblick über den aktuellen State of the Art, wobei verschiedene Verfahren zur Minimierung und Überbrückung des Simulation-Reality-Gaps vorgestellt werden.
- Das reale System (BRIO®-Testbed) sowie die Simulation des BRIO®-Labyrinth-Spiels werden in Kapitel 3 kurz vorgestellt. Dabei wird zuerst das BRIO®-Labyrinth-Spiel selber vorgestellt, bevor dann im Anschluß ein Überblick

über die am BRIO®-Testbed vorgenommenen Erweiterungen bzw. die verwendeten Verfahren gegeben wird. Zu guter letzt erfolgt eine kurze Vorstellung der vom DFKI entwickelte Simulation des BRIO®-Labyrinth-Spiels.

- Im Anschluß wird in Kapitel 4 das Konzept der Arbeit präsentiert. An dieser Stelle sollen die einzelnen Untersuchungen und gewählten Ansätze zur Untersuchung und Minimierung des Simulation-Reality-Gaps aufgezeigt werden.

- In Kapitel 5 wird die Bestimmung der Brettwinkel des BRIO®-Testbeds genauer untersucht. Hierbei soll vor allem die Messunsicherheit der Brettwinkel des Spielbretts untersucht werden.

- Die Positioniergenauigkeit des BRIO®-Testbeds wird im 6. Kapitel analysiert. Hierbei wurde zunächst der interne Positionierungsfehler der Servomotoren näher betrachtet und versucht mittels zweier Methoden diesen zu minimieren. Anschließend wurden die Auswirkungen des Positionierungsfehlers als auch des Spiels der Gummiriemen auf den Fehler und das Rauschen der angefahrenen Brettwinkel ergründet.

- Kapitel 7 enthält die durchgeführten Untersuchungen und Anpassungen der Bestimmung der Kugelposition des BRIO®-Testbeds. Neben der Erweiterung der Bildakquise des BRIO®-Testbeds um eine Kompensation der Kameraverzeichnung wurden beide Systeme auf ein einheitliches Koordinatensystem umgestellt und die resultierenden Abweichungen der Kugelpositionen des BRIO®-Testbeds und der Simulation eruiert.

- Im Verlauf der Arbeit wurde eine relativ große Abweichung zwischen dem Verhalten der realen Servomotoren und den simulierten Motoren festgestellt. Aus diesem Grund werden in Kapitel 8 entsprechende Optimierungen der Motorparameter der Simulation vorgenommen. Neben einer manuellen Anpassung wurden die Motorparameter ebenfalls mittels Evolutionsstrategie optimiert.

- In Kapitel 9 wird ein Verfahren zum Vergleichen des Verhaltens des BRIO®-Testbeds mit dem der Simulation vorgestellt. Hierbei werden die zeitlichen Verläufe der Zustände beider Systeme hinsichtlich ihrer Überdeckungen untersucht und ausgewertet. Anschließend werden Auswirkungen der Anpassungen beider Systeme hinsichtlich der Minimierung des Simulation-Reality-Gaps analysiert.

- Im 10. Kapitel werden die Ergebnisse der Arbeit kurz zusammengefasst und ein Ausblick über mögliche weitere Forschungsarbeiten gegeben.

Kapitel 2

State of the Art

In den bisherigen Arbeiten haben sich verschiedene Methoden zum Minimieren und Überbrücken des Simulation-Reality-Gaps herausgebildet, welche in den folgenden Unterabschnitten kurz erläutert werden sollen. Hierbei werden zunächst die Verfahren zur Minimierung des Simulation-Reality-Gaps vorgestellt und anschließend zwei Verfahren zum Überbrücken des Gaps. Es ist jedoch zu beachten, dass die einzelnen Methoden keineswegs nur separat von einander eingesetzt werden können. Oftmals wurden zwei oder mehrere dieser Methoden miteinander kombiniert.

2.1 Verfahren zum Minimieren des Simulations-Reality-Gaps

Bei diesen Verfahren wird versucht eine möglichst akkurate Simulation des realen Systems zu erstellen und den Roboter bereits während der Trainingsphase innerhalb der Simulation auf die realen Bedingungen zu konditionieren.

2.1.1 Modellierung der Realität basierend auf Sensoren und Aktuatoren des Roboters

Die erste Methode verwendet eine auf den Werten der realen Sensoren und Aktuatoren basierende Simulation des Roboters zum Training eines Agenten. Dazu wird die zu simulierende Umgebung mittels der Sensoren und Aktuatoren des realen Systems abgetastet bzw. vermessen und mit den erhaltenen Werten die Simulation als eine Art Zustandstabelle aufgebaut. Im Ergebnis liefert die Simulation ausgehend vom aktuellen Zustand des Roboters, wie z.B. aktueller Position, Orientierung, Bewegungsrichtung und -geschwindigkeit die aufgezeichneten Sensor-

bzw. Aktuatorwerte zurück.
Der Vorteil dieser Vorgehensweise besteht in der relativen Nähe der Simulation zur Realität. Der Agent wird bereits in der Simulation mit realen Werten der Sensoren und Aktuatoren trainiert. Somit besteht die Möglichkeit, dass sich der Agent bereits während der Trainingsphase an etwaige Eigenheiten des Roboters anpasst.
Nachteilhaft an dieser Methode ist jedoch der hohe Aufwand bei der Erstellung der Simulation. Die gesamte Testumgebung sowie das Verhalten des Roboters in dieser Umgebung muss mit sämtlichen Aktuatoren und Sensoren vollständig erfasst werden. Je nach Komplexität des Roboters (Anzahl der Sensoren und Aktuatoren) und der Größe bzw. Umfang der Testumgebung steigt der Aufwand exponentiell an.
Bereits 1995 setzte unter anderem Orazio Miglino diesen Ansatz bereits ein, um eine Simulation eines einfachen Khepera-Roboters zu erstellen [14]. Jedoch zeigte sich nach der Übertragung des Agenten auf das reale System ein Abfallen der Performance des Agenten. Ähnlich entwarfen Stefano Nolfi und Henrik Hautop Lund ihre Simulation eines Khepera-Roboters basierend auf den Abtastwerten der realen Sensoren [16][12]. Hierbei wurde jedoch kein Unterschied zwischen den Fitnesswerten vor und nach der Übertragung auf den realen Roboter festgestellt. Jedoch konnten die Fitnesswerte der evolvierten Agenten durch das Fortsetzen des Trainingsprozess auf dem realen System erhöht werden.

2.1.2 Hinzufügen von Rauschen zu den Sensor- und Aktuatorwerten der Simulation

Eine weitere Methode zur Minimierung des Simulation-Reality-Gap ist das Hinzufügen von Rauschen zu den einzelnen Sensor- und Aktuatorwerten der Simulation, um den Fehler bzw. die Unsicherheiten der realen Bauteile zu simulieren und bereits beim Training des Agenten zu berücksichtigen. Hierbei kann der Abfall der Performance des Agenten nach dem Übertragen auf den Roboter nicht immer auf eine Fehlanpassung in den entwickelten Verhaltensstrategien, sondern eher auf eine noch ungenügende Modellierung der Realität durch die Simulation zurückgeführt werden [16].
Am Beispiel eines einfachen Khepera-Roboters wurde gezeigt, dass durch Hinzufügen eines entsprechend modellierten Rauschens zu den Sensor- und Aktuatorwerten der Simulation während der Trainingsphase der Abfall der Performance des Roboters im Vergleich zur Simulation gemindert werden konnte [14] [10]. Hierbei lernt der Agent anhand der verrauschten Sensordaten und ist somit in

der Lage die Ungenauigkeiten der realen Bauteile in seinem Verhalten zu berücksichtigen. Es hat sich jedoch gezeigt, dass das jeweilige Rauschen mit großer Sorgfalt bestimmt bzw. gewählt werden muss [15] [10]. Neben der Tatsache, dass das Hinzufügen eines geeigneten Rauschens die Performance des Roboters nach der Übertragung auf das reale System verbessert, kann ein zu starkes Verrauschen der Sensor- bzw. Aktuatorwerte die Performanceunterschiede zwischen dem realen und dem simulierten Roboter wieder vergrößern. Folglich erfordert diese Methode eine genaue Untersuchung des Verhaltens jedes einzelnen Sensors und Aktuators des realen Systems.

2.1.3 Envelope-of-Noise

Die *Envelope-of-Noise-Hypothese* oder auch *Minimal Simulation* wurde 1997 von Nick Jakobi vorgestellt [8] [9]. Bei dieser Methode werden der Simulation neben dem Rauschen der einzelnen Bauteile (Kapitel 2.1.2) zusätzliche Parameter oder Komponenten der Simulation verrauscht. Ein Agent, welcher in der Lage ist trotz dieses Rauschmantels (engl. *envelope-of-noise*) die an ihn gestellte Aufgabe zu erfüllen, sollte ohne weiteres in der Lage sein die Unsicherheiten des realen Systems meistern zu können.
Der Ansatz dieser Methode beruht auf der Beobachtung, dass nicht alle Aspekte des realen Systems akkurat modelliert werden müssen, sondern nur eine Untermenge von Aspekten (*Basismengen-Aspekte*) auf denen das zu entwickelnde Verhalten basieren soll. Wenn die einzelnen Elemente dieser Untermenge akkurat modelliert werden können und der resultierende Agent darauf beschränkt werden kann, sein Verhalten ausschließlich beruhend auf dieser Untermenge zu erlernen, wird es vom Standpunkt des Agenten keinen Unterschied zwischen der Simulation und dem Roboter geben. Somit muss anstelle einer vollständigen Simulation aller Eigenschaften und Prozesse der realen Welt lediglich eine kleinere Untermenge davon exakt nachgebildet werden und die Komplexität der Simulation reduziert sich damit entsprechend auf ein Minimum (Minimal Simulation). Im Folgenden soll ein grober Überblick über die Vorgehensweise des Verfahrens gegeben werden [8].

1. Zuerst müssen alle Basismengen-Aspekte des Roboters identifiziert und entsprechend der Realität akkurat modelliert werden. Basierend auf diesen Basismengen-Aspekten wird anschließend eine Simulation erstellt. Um jedoch den simulierten Roboter vollständig implementieren zu können und die einzelnen Basismengen-Aspekte korrekt miteinander zu verbinden, muss die Simulation um weitere Aspekte bzw. Komponenten erweitert werden. Diese

sogenannten *Implementations-Aspekte* haben meist keinen direkten Bezug zur Realität.

2. Während der Trainingsphase wird jeder Implementations-Aspekt der Simulation zufällig variiert, so dass ein Agent, dessen Verhalten von diesen Aspekten abhängt, unzuverlässig wird. Dies kann durch das Hinzufügen eines genügend großen Rauschens sichergestellt werden, so dass der resultierende Agent gezwungen wird, die Implementations-Aspekte als unzuverlässig von seinen Betrachtungen auszuschließen.

3. Jeder Basismengen-Aspekt wird ebenfalls zufällig variiert, so dass der resultierende Agent gegen Ungenauigkeiten dieses Aspektes robust wird. Das Ausmaß und der Charakter dieser zufälligen Streuung muss ausreichend sein, um sicherzustellen, dass die entstehenden Agenten mit dem unvermeidlichen Unterschied zwischen der Simulation und der Realität umgehen können, aber nicht so groß, dass sich kein Agent entwickeln kann.

Nick Jakobi [8] erstellte mittels dieser Methode eine Simulation eines Khepera-Roboters, welcher in Abhängigkeit von einem Lichtsignal an einer T-Kreuzung nach links oder rechts abbiegen sollte. Die so trainierten Agenten konnten die gestellte Aufgabe nach Übertragung auf den realen Roboter erfüllen. Hierzu wurden

- die Reaktion des Roboters auf die entsprechenden Motorsignale
- die Bestimmung des Abstandes des Roboters zu seiner Umgebung mittels seiner Infrarotsensoren
- die Reaktion der Umgebungslichtsensoren des Roboters auf unterschiedliche Helligkeiten

als Basismengen-Aspekte identifiziert und entsprechend des Rauschen der realen Bauteilen variiert. Als Implementationsaspekte wurde

- die Seite des Korridors, von welchem das Lichtsignal ausgeht
- die Breite der Korridore
- die Orientierung des Roboters zum Startzeitpunkt
- die Länge des erleuchteten Bereichs des Korridors
- die Länge des Zufahrtskorridors zur T-Kreuzung

bzw. ihren Werten während des Trainings des Agenten innerhalb der Simulation ein entsprechend starkes, unnatürliches Rauschen hinzugefügt.
Zusätzlich wendete Jakobi diese Methode ebenfalls zum Trainieren eines Agenten zur Phototaxis eines weißen Dreiecks an den Wänden einer Testumgebung an.

2.1.4 Estimation-Exploration-Algorithmus

Ein anderer Ansatz ist der 2004 von Hod Lipson und Josh Bongard vorgestellte *Estimation-Exploration-Algorithmus* [1]. Diese Methode besteht aus der Co-Evolution von drei Populationen. Die erste Population beinhaltet verschiedene Sätze von Simulator-Parametern. Eine zweite Population besteht aus Tests, welche Unstimmigkeiten zwischen den einzelnen Simulationen mit verschiedenen Parametersätzen hervorrufen sollen. Agenten zur Erfüllung einer bestimmten Aufgabe (z.B. Vorwärts-Lokomotion) werden in einer dritten Population verschlüsselt. Der Ablauf der Co-Evolution ist in drei Phasen unterteilt (Abbildung 2.1).

1. In Phase eins, der sogenannten *Explorations-Phase*, wird zunächst eine zufällige Population von Tests erzeugt und unter Verwendung der aktuell besten Simulator-Parametern evolviert. Dabei beruht die Fitness der einzelnen Agenten darauf, dass unter den verschiedenen Sätzen von Simulator-Parametern unterschiedliche Verhaltensweisen herbeigeführt werden. Anschließend wird der beste Test auf dem realen System ausgeführt, wobei alle Daten des Systems aufgezeichnet und zu den vorher bereits aufgenommenen Testdaten hinzugefügt werden.

2. In Phase zwei, der sogenannten *Estimations-Phase*, wird eine Population von Simulator-Parametern evolviert. Hierbei beschreibt die Fitness der einzelnen Individuen wie gut die einzelnen Simulator-Parameter alle bisher aufgezeichneten Daten des realen Systems erklären können. Je weniger die simulierten von den aufgezeichneten Daten des realen Systems bei Ausführung der verschiedenen Tests innerhalb der Simulation abweichen, desto höher ist der Fitnesswert des jeweiligen Satzes von Simulator-Parametern. Erfüllen die besten Simulator-Parameter am Ende der Evolution entsprechende Kriterien, so wird Phase drei mit den ermittelten Parametern gestartet. Andernfalls wird Phase eins unter Verwendung der besten Parameter erneut durchgeführt.

3. In Phase drei erfolgt die eigentliche Evolution bzw. das Training des Agenten mittels der evolvierten Simulator-Parameter. Hierbei bestimmt sich die

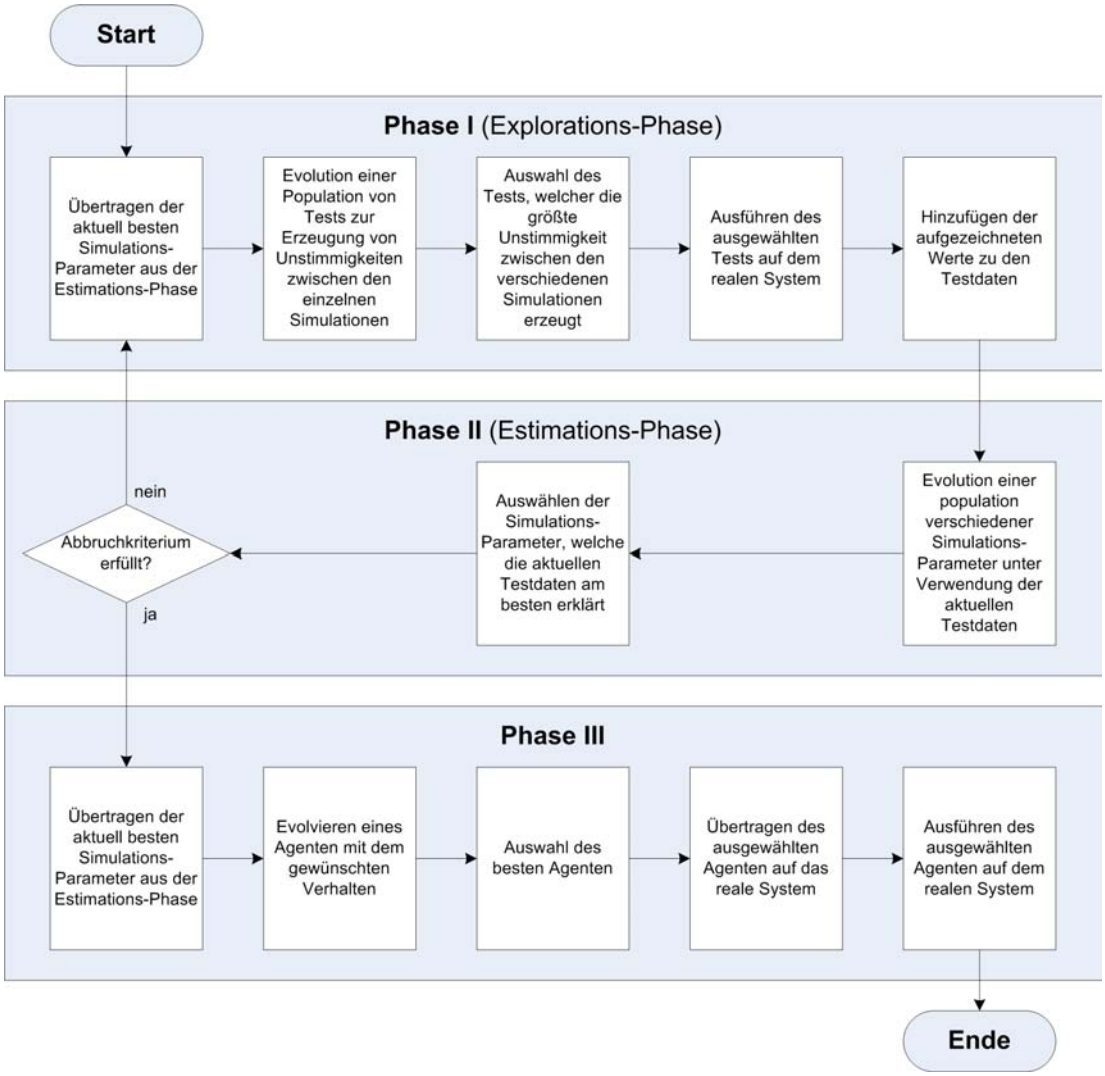


Abbildung 2.1: Ablauf der drei Phasen des Estimation-Exploration-Algorithmus

Fitness der einzelnen Agenten dadurch, dass ein gewünschtes Verhalten ausgelöst bzw. herbeigeführt wird. Der beste Agent dieser Population wird anschließend auf das reale System übertragen und ausgeführt. Sollte die evolvierte Simulation akkurat genug sein, so wird der Roboter das gleiche Verhalten wie in der Simulation zeigen.

Hod Lipson und Josh Bongard setzten diese Methode bereits erfolgreich zur Modellierung eines vierbeinigen Roboters innerhalb einer Simulationsumgebung und bei der Ferndiagnose von Fehlfunktionen eines elektromechanischen Systems mit Hilfe eines simulierten Modells ein [1].

2.2 Verfahren zum Überbrücken des Simulations-Reality-Gaps

Bei diesen Verfahren wird nicht versucht eine möglichst akkurate Nachbildung der realen Bedingungen innerhalb einer Simulation zu erreichen, sondern versucht durch Adaption der erlernten Verhaltensstrategien nach der Übertragung des Agenten auf das reale System den Agenten mit den realen Bedingungen vertraut zu machen.

2.2.1 Fortsetzen des Trainings auf dem realen System

Eine Methode, um die Lücke zwischen der Simulation und der Realität zu überwinden, ist das Fortsetzen der Trainingsphase des Agenten auf dem realen System. Hierbei wird zunächst der Agent mittels einer entsprechenden Simulation des Roboters trainiert. Anschließend wird die erlernte Strategie auf den realen Roboter übertragen, um dann für wenige Generationen das Training auf dem realen System fortzusetzen. Während dieser zweiten Trainingsphase wird die in der Simulation gelernte Strategie entsprechend den in der Realität herrschenden Unzulänglichkeiten angepasst und die Performancelücke zwischen simulierten und realem Verhalten reduziert [14] [16].
Der Vorteil dieser Vorgehensweise liegt darin, dass der Großteil des Lernvorgangs innerhalb der Simulation erfolgen kann und lediglich eine kurze zweite Trainingsphase auf dem realen System notwendig ist, um das erlernte Verhalten an die realen Gegebenheiten anzupassen.

2.2.2 Evolution des Lernens

Die letzte Methode zur Überwindung des Simulation-Reality-Gaps basiert auf der *Evolution des Lernens* anstelle der Evolution von Kontrollparametern. Die Grundidee hierbei ist, Strukturen bzw. Agenten zu trainieren bzw. zu erzeugen, welche in der Lage sind, sich während der Laufzeit an partiell unbekannte Umgebungen anzupassen. Zu diesem Zweck erfolgt keine Evolution der Parameter des Systems selbst, sondern es werden verschiedene Verfahren bzw. Strukturen hinsichtlich ihrer Anpassungsfähigkeit optimiert. Nach der Übertragung auf den Roboter werden die Kontrollparameter des Agenten entsprechend der evolvierten Methode und der jeweiligen Umgebung des Roboters zur Laufzeit angepasst.
Ein Ansatz ist die Nutzung Neuronaler Netze mit adaptiven Synapsen. Hierbei werden die Parameter des Agenten zu Beginn mit kleinen zufälligen Werten initialisiert und während der Laufzeit entsprechend der jeweiligen Lernregel angepasst

[24]. Joseba Urzelai und Dario Floreano implementierten ein entsprechendes Neuronales Netz zur Steuerung eines Khepera-Roboters, welcher sich anhand seiner Abstandssensoren sowie Bilder einer Kamera innerhalb einer Testumgebung bewegen soll. Hierbei differierten die realen von der während der Trainingsphase benutzten Testumgebung. Die Ergebnisse zeigten, dass durch die Verwendung evolutionärer adaptiver Reglers die gestellte Aufgabe im Vergleich zu den evolutionären Reglern mit festen Gewichten, bessere Ergebnisse erreicht wurden.
Ein anderer Ansatz besteht darin, selbstregulierende Neuronale Netze [4] einzusetzen. Hierbei werden die Gewichte des Neuronalen Netzes angepasst, sobald die Aktivierung der einzelnen Neuronen einen vorgegebenen Wertebereich verlassen. Ezequiel A. Di Paolo evolvierte Agenten zur Ausführung einer Langzeit-Phototaxis auf eine Serie von Lichtquellen, während versucht wurde die Werte der Neuronen innerhalb eines vorgegebenen Bereichs zu halten. Anschließend wurde die Fähigkeit des Roboters getestet, die an ihn gestellte mit einem invertierten Sichtfeld oder verrauschten Sensoren bzw. Aktuatoren zu erfüllen. Nach einer längeren Phase der Fehlanpassung konnte in allen Fällen sich der Roboter an die neuen Bedingungen anpassen.

Kapitel 3

Grundlagen

3.1 BRIO®-Labyrinth-Spiel

Das BRIO®-Labyrinth-Spiel wurde 1946 von der Firma BRIO®[1] entwickelt und wird seitdem von Menschen aller Altersgruppen weltweit gespielt.

Abbildung 3.1: Original BRIO®-Labyrinth-Spiel

Es besteht aus einer stabilen hölzernen Kiste, einer Kugel, dem Spielbrett und zwei Knöpfen mit deren Hilfe man die Neigung des Spielbretts verändern kann (Abbildung 3.1). Das Spielbrett (Abbildung 3.2) selbst ist ein Labyrinth aus Trennwänden und Löchern. Des Weiteren sind ein Start- und ein Endpunkt sowie der vorgeschriebene Weg zwischen diesen beiden Punkten auf dem Spielbrett eingezeichnet. Den einzelnen Löchern im Spielbrett sind nach ihrer Reihenfolge entlang des markierten Weges Nummern bzw. Punkte zu geordnet.

[1] http://www.brio.net/

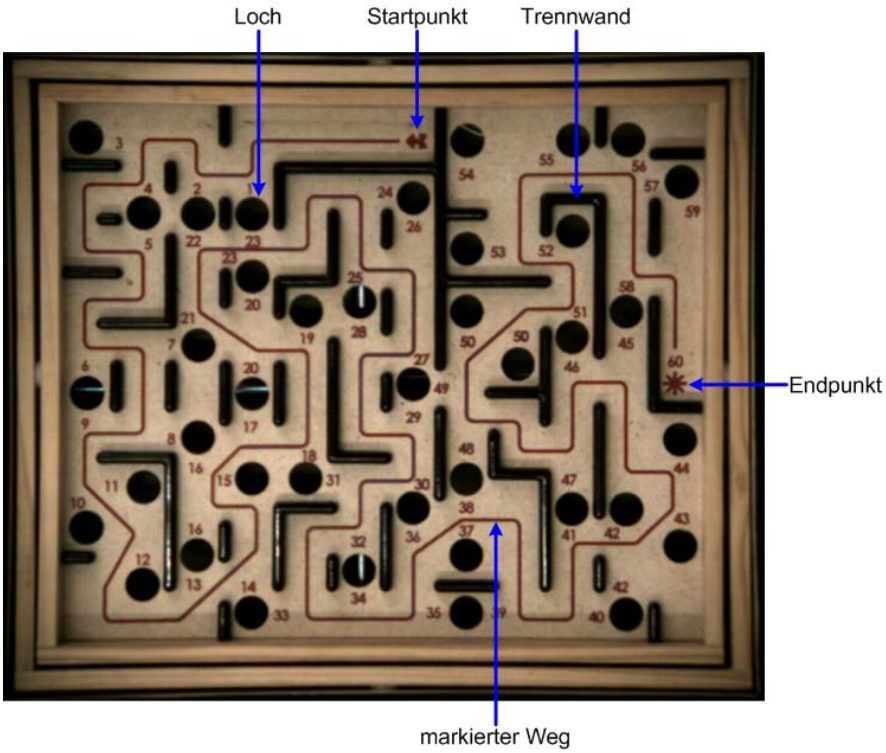


Abbildung 3.2: Aufbau des Spielbretts

Ziel des Spiels ist es, die Kugel mittels Neigen und Kippen des Spielbretts vom Startpunkt aus entlang des markierten Weges an den Löchern vorbei durch das Labyrinth zum Endpunkt zu bewegen. Je weiter die Kugel durch das Labyrinth manövriert wird, desto mehr Punkte erhält der Spieler. Für das Erreichen des Endpunktes erhält man 60 Punkte. Fällt die Kugel vorher durch eines der Löcher, so erhält der Spieler nur die Nummer des Loches als Punktzahl.

3.2 BRIO®-Testbed

In dem nun folgenden Kapitel wird das BRIO®-Testbed (Abbildung 3.3) vorgestellt werden. Dabei wird zunächst ein Überblick über den Aufbau des BRIO®-Testbeds gegeben und anschließend werden die einzelnen Bauteile sowie eingesetzten Methoden kurz vorgestellt.

3.2.1 Überblick

Abbildung 3.4 stellt den Aufbau des BRIO®-Testbeds grafisch dar. Um das automatisierte Spielen des BRIO®-Labyrinth-Spiels zu ermöglichen, wurde das eigentliche Spiel um mehrere Sensoren und Aktuatoren erweitert. Im Einzelnen handelt es sich dabei um zwei Servomotoren $S2$ und $S3$ (ROBOTIS Dynamixel

Abbildung 3.3: BRIO®-Testbed

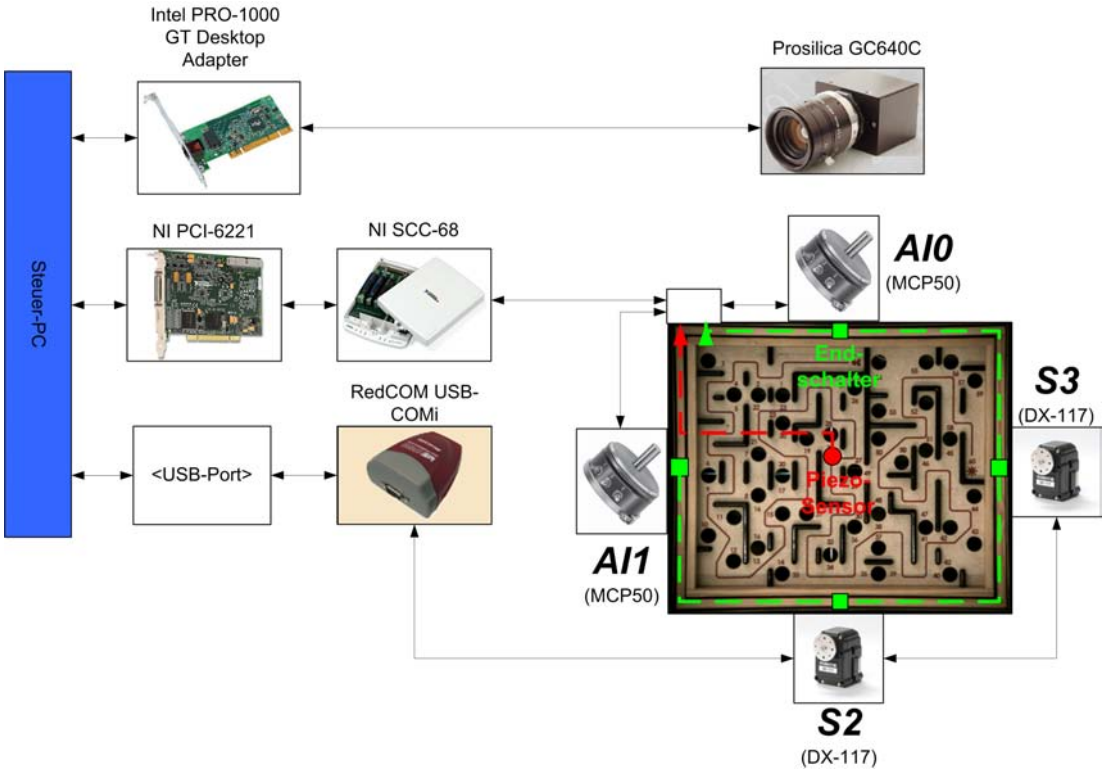


Abbildung 3.4: Aufbau des BRIO®-Testbeds

DX-117) zur Kontrolle der Brettneigung, eine digitale Videokamera (PROSILICA GC640C) zur Bestimmung der Kugelposition, zwei Potentiometer $AI0$ und $AI1$ (MEGATRON MCP50) zur Bestimmung der Winkel des Spielbretts, vier Endschalter als Brettanschläge zur Kalibrierung des Testbeds sowie einen Piezo-Sensor, der das Herunterfallen des Balles detektiert. Hinzu kommen noch weitere Peripheriegeräte, die die Daten der verwendeten Sensoren auslesen (NI PCI-6221, NI SCC-68) und mit einzelnen Bauteilen kommunizieren (Intel PRO-1000 GT Desktop Adapter, RedCOM USB-COMi), sowie ein Steuer-PC,der die erhaltenen Sensorwerte verarbeitet und die beiden Servomotoren ansteuert.

3.2.2 Bauelemente

In diesem Abschnitt werden die einzelnen verwendeten Bauelemente kurz vorgestellt.

3.2.2.1 ROBOTIS Dynamixel DX-117

Zum Verändern der Winkel des Spielbretts werden zwei Dynamixel DX-117 Servomotoren der Firma ROBOTIS[2] eingesetzt. Diese sind über Gummiriemen und den Steuerknöpfen mit den jeweiligen Achsen des Spiels verbunden (Abbildung 3.5). Die Achsen des Spiels wiederum sind mittels eines robusten Bindfaden, welcher einseitig durch einen O-Ring aus Gummi unter Spannung gehalten wird, mit dem eigentlichen Spielbrett verbunden.

Abbildung 3.5: Übertragung der Bewegungen der Servomotoren mittels Gummiriemen auf die Achsen des Spielbretts

Die Servomotoren selbst haben einen Drehbereich von 300° und können dabei 1024 verschiedene Winkelstellungen einnehmen (Abbildung 3.6). Somit entspricht ein einzelner Schritt des Servomotors einer Drehung der Achse des Servomotors von rund $300° : 1023 = 0,2933°$ bzw. $0,0051 rad$.

[2] http://www.robotis.com/

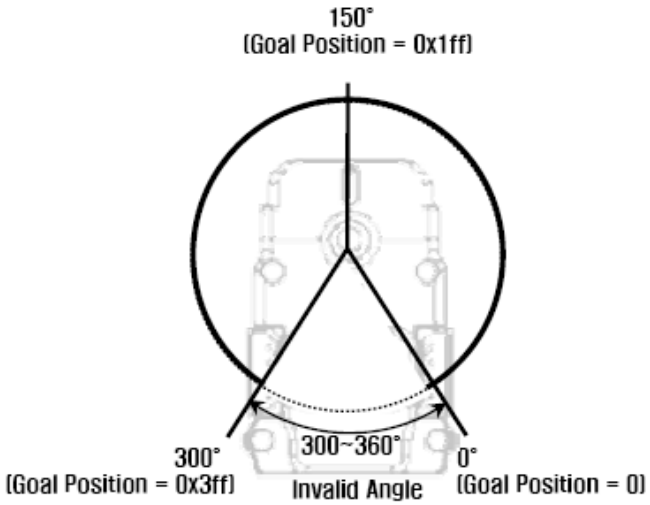


Abbildung 3.6: Drehbereich eines DX-117 Servomotors [21]

Das Verhalten der Servomotoren kann mittels mehrerer Parameter beeinflusst werden. Zu diesen gehören unter anderem der minimale Strom (*Punch*), der dem Motor während einer Bewegung zur Verfügung steht, die maximale Bewegungsgeschwindigkeit (*Moving Speed*), das maximale Ausgangsdrehmoment (*Output-Torque-Limit*) und die Parameter *Compliance Slope* und *Compliance Margin* für Bewegungen mit (engl. *clockwise*, Abk. *CW*) und entgegen (engl. *counter-clockwise*, Abk. *CCW*) dem Uhrzeigersinn. Eine umfassende Beschreibung der einzelnen Parameter und der Handhabung der Servomotoren kann dem entsprechenden Handbuch [21] entnommen werden. Die standardmäßig verwendeten Werte der einzelnen Parameter sind in Tabelle 3.1 angegeben.

Parameter	Hexadezimalwert	Dezimalwert
Punch	$0x0020$	32 (default)
Moving Speed	$0x03FF$	1023 (default)
Output-Torque-Limit	$0x03FF$	1023 (default)
CCW Compliance Slope	$0x3C$	60
CCW Compliance Margin	$0x00$	0
CW Compliance Margin	$0x00$	0
CW Compliance Slope	$0x46$	70

Tabelle 3.1: Standardparameter der Servomotoren DX-117

3.2.2.2 RedCOM USB-COMi

Die Kommunikation mit den Servomotoren erfolgt über eine RS485-Schnittstelle. Da der verwendete Steuer-PC BMM3 über keine entsprechende serielle Schnittstelle verfügt, wurde ein Schnittstellenumsetzer RedCOM USB-COMi (Abbildung

3.7) der Firma MEILHAUS ELECTRONIC[3] verwendet, um die Kommunikation des Steuer-PCs mit den Servomotoren zu realisieren.

Abbildung 3.7: RedCOM USB-COMi

3.2.2.3 Endschalter

Des Weiteren wurde das BRIO®-Labyrinth-Spiel um vier Endschalter, je einer in den vier Anschlägen der Spielebene, erweitert. Die Endschalter werden für eine anfängliche Kalibrierung des Testbeds (Kapitel 3.2.3.1) genutzt, um den maximalen Bewegungsspielraum beider Achsen zu bestimmen.

3.2.2.4 Piezo-Sensor

Um das Herunterfallen der Kugel zu detektieren wurde ein Piezo-Sensor am Boden des BRIO®-Labyrinth-Spiels angebracht. Mit Hilfe dieses Piezo-Sensors können die Vibrationen der Bodenplatte des Spiels aufgezeichnet und ausgewertet werden.

3.2.2.5 MEGATRON MCP50

Da die in den Servomotoren integrierten Potentiometer lediglich ein maximales Auflösungsvermögen von $0,2933°$ bieten, wurden zusätzlich an beide Achsen des BRIO®-Testbeds zwei Leitplastik-Potentiometer MCP50 der Firma MEGATRON Elektronik AG & Co.[4] montiert (Abbildung 3.8). Diese Potentiometer zeichnen sich durch eine sehr geringe Linearitätstoleranz von $0,1\%$ aus und sind mit einer Lebensdauer von bis zu 50 Millionen Achsumdrehungen sehr zuverlässig. Mit Hilfe dieser Potentiometer ist es möglich die Brettwinkel des BRIO®-Labyrinth-Spiels mit einer Auflösung von $0,02304°$ zu bestimmen.

[3] http://www.meilhaus.de/
[4] http://megatron.de/

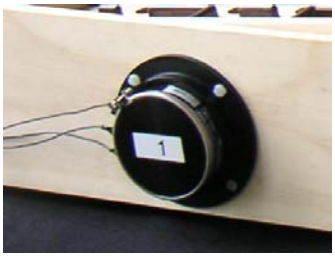


Abbildung 3.8: Anbringung des MEGATRON MCP50 Potentiometers an der Achse des BRIO®-Spiels

3.2.2.6 NI PCI-6221 und NI SCC-68

Das Auslesen des Piezo-Sensors, der Endschalter sowie der beiden Potentiometer wird mittels Geräten der Firma National Instruments[5] realisiert. Zu diesem Zweck wird das Datenerfassungsmodul PCI-6221, welches in den Steuer-PC eingebaut wurde, und der 68-Pin Anschlussblock mit SCC Erweiterungs-Slot SCC-68 verwendet (Abbildung 3.9).

(a) NI PCI-6221 (b) NI SCC-68

Abbildung 3.9: Datenerfassungsmodul (a) und 68-Pin Anschlussblock (b) der Firma National Instruments

Mit Hilfe der beiden Module können die beiden Potentiometer sowie der Piezo-Sensor mit einer Auflösung von 16 Bit und einer maximal möglichen Abtastrate von bis zu $250kHz$ analog abgetastet werden. Der Bereich der messbaren Spannungen an den analogen Eingängen liegt hierbei zwischen $-10V$ und $+10V$. Laut Herstellerangaben beträgt somit die kleinste aufzulösende Spannung der analogen Eingänge 320 μV. Des Weiteren werden die vier Endschalter digital ausgelesen.

3.2.2.7 PROSILICA GC640C

Zur Bestimmung der Kugelposition wird eine Digitalkamera vom Typ GC640C der Firma PROSILICA[6] (Abbildung 3.10) eingesetzt. Diese Kamera bietet eine maximale Auflösung von 659 mal 493 Pixel und unterstützt dabei eine Reihe von Bildformaten (Mono8, Bayer8, RGB24, YUV411, ... usw.).

[5] http://www.national-instruments.com/
[6] http://www.prosilica.com/

Abbildung 3.10: PROSILICA GC640C

Die Kamera kommuniziert über Gigabit-Ethernet und ist somit in der Lage auch große Datenmengen schnell zu übertragen. In Abhängigkeit von Farbformat, Auslastung des Gigabit-Netzwerkes und der gewählten Auflösung sind Bildwiederholraten von bis zu 197 Frames pro Sekunde möglich.

3.2.2.8 Intel PRO-1000 GT Desktop Adapter

Um eine möglichst optimale Kommunikation zwischen der digitalen Videokamera und dem Steuer-PC zu ermöglichen, wurde ein zur Kamera kompatibler Gigabit-Ethernet-Adapter in den Steuer-PC eingebaut und mit der Kamera direkt verbunden.

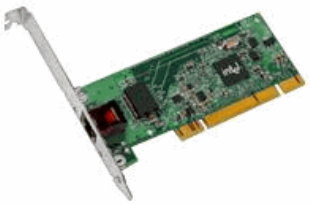


Abbildung 3.11: Intel PRO-1000 GT Desktop Adapter

Der Intel PRO-1000 GT Desktop Adapter (Abbildung 3.11) wird ausdrücklich vom Hersteller der digitalen Videokamera empfohlen und soll einen optimalen Datenstrom von der Kamera zum Steuer-PC ermöglichen.

3.2.3 Verfahren und Methoden des BRIO®-Testbeds

Im Anschluß werden einzelne vom BRIO®-Testbed verwendete Methoden bzw. Verfahren vorgestellt.

3.2.3.1 Berechnung der Brettwinkel des BRIO®-Testbeds basierend auf den gemessenen Potentiometerspannungen

Zu Beginn soll die Berechnungsvorschrift der Brettwinkel ϕ_X bzw. ϕ_Y aus den Potentiometerspannungen U_{AI0} und U_{AI1} erläutert werden. Bei dieser Umrechnung wird von einem linearen Zusammenhang zwischen den Winkeln der Achsen

des BRIO®-Labyrinth-Spiels und der Neigung des Spielbretts ausgegangen (Abbildung 3.12).

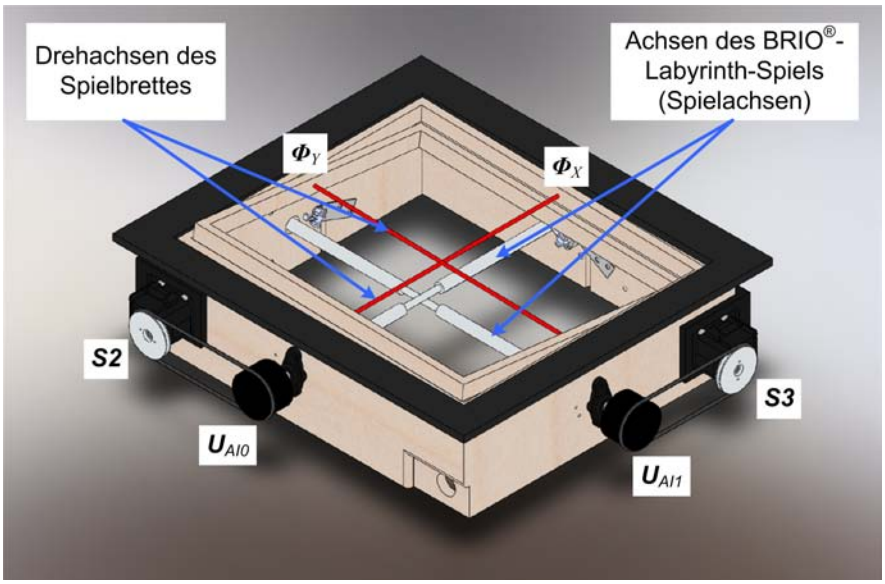


Abbildung 3.12: Berechnung der Brettwinkel ϕ_X und ϕ_Y der *Drehachsen des Spielbretts* basierend auf den gemessenen Potentiometerspannungen U_{AI0} und U_{AI1} der *Achsen des BRIO®-Labyrinth-Spiels*

Hierzu werden während der Kalibrierung des Systems pro Achse jeweils drei Referenzspannungen bestimmt; je eine Referenzspannung, wenn sich das Spielbrett im linken bzw. unteren Endanschlag U_{min}, in der Nullposition U_{center} und im rechten bzw. oberen Endanschlag U_{max} befindet. Mit Hilfe einer digitalen Wasserwaage wurden die jeweiligen Winkel des Spielbretts in den einzelnen Positionen gemessen und als Referenzwinkel zur Berechnung der Auslenkung des Spielbretts eingesetzt. Es ergab sich für den Brettwinkel ϕ_X eine mittlere Auslenkung des Spielbretts von $9,11°$ für das Spielbrett im linken Endanschlag und von $-8,33°$ für das Spielbrett im rechten Endanschlag. Für den Brettwinkel ϕ_Y ergab sich ein Referenzwinkel für das Spielbrett im unteren Brettanschlag von $-5,43°$ und für das Spielbrett im oberen Endanschlag von $5,13°$. Unter Berücksichtigung der jeweiligen Referenzwerte können die Brettwinkel ϕ_X und ϕ_Y aus den zugehörigen Spannungswerten U_{AI0} bzw. U_{AI1} mit Hilfe der Formeln (3.1) und (3.2) berechnet werden.

$$\phi_X(U_{AI0}) = \begin{cases} +9,11° \cdot \frac{U_{center}-U_{AI0}}{U_{center}-U_{max}} & \text{für } U_{AI0} < U_{center} \\ -8,33° \cdot \frac{U_{center}-U_{AI0}}{U_{center}-U_{min}} & \text{für } U_{center} \leq U_{AI0} \end{cases} \tag{3.1}$$

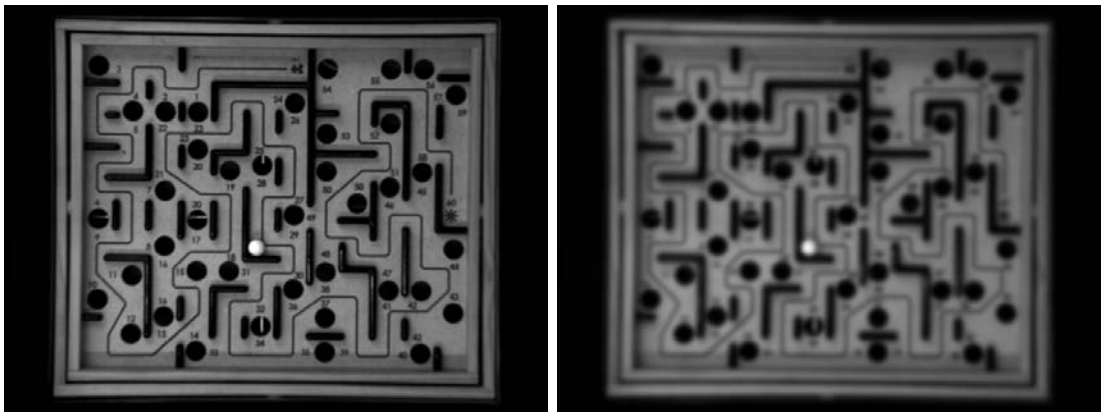
$$\phi_Y(U_{AI1}) = \begin{cases} -5,43° \cdot \frac{U_{center}-U_{AI1}}{U_{center}-U_{max}} & \text{für } U_{AI1} < U_{center} \\ +5,13° \cdot \frac{U_{center}-U_{AI1}}{U_{center}-U_{min}} & \text{für } U_{center} \leq U_{AI1} \end{cases} \tag{3.2}$$

3.2.3.2 Algorithmus zur Bestimmung der Position der Kugel im Kamerabild

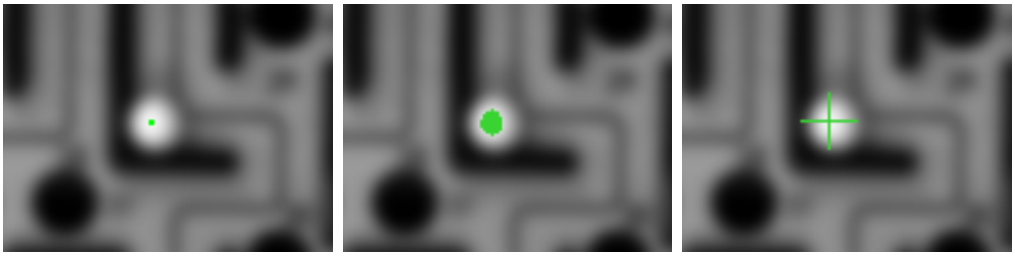
In diesem Abschnitt wird das zur Kugelerkennung eingesetzte Verfahren kurz vorgestellt. Die Ansteuerung der Kamera des BRIO®-Testbeds und die Übertragung der Bilder von dieser zum Steuer-PC basiert auf der von PROSILICA zur Verfügung gestellten Programmierschnittstelle *PvAPI* [18]. Die Implementierung des Algorithmus zur Kugeldetektion erfolgte unter Verwendung von *OpenCV* [7]. Diese quelloffene Programmbibliothek wurde im September 2006 in der Version 1.0 von Intel[7] herausgegeben und seitdem beständig weiterentwickelt. Die OpenCV-Bibliothek beinhaltet unter anderem Algorithmen zur Gesichtserkennung, 3D-Funktionalität, Haar-Klassifikatoren, verschiedene sehr schnelle Filter sowie Funktionen für die Kamerakalibrierung. Des Weiteren ist eine große Anzahl von Algorithmen basierend auf neuesten Forschungsergebnissen bereits in OpenCV implementiert.

Bei der Detektion der Position der weißen Kugel des BRIO®-Testbeds wird davon ausgegangen, dass sich im Bild der Kamera neben der Kugel selbst kein helles Objekt befindet. Aufgrund dieser Tatsache konnte der Aufwand bzw. die Komplexität der Kugeldetektion verringert werden.
Nachdem mit Hilfe der PvAPI die von der PROSILICA GC640C aufgenommenen Graustufenbilder (Abbildung 3.13 (a)) zum Steuer-PC übertragen wurden, werden die einzelnen Bilder auf das Vorhandensein der weißen Spielkugel und deren Position hin untersucht. Um die Auswirkungen des Bildrauschens zu reduzieren und somit die Kugelerkennung stabiler zu machen, wird das aufgenommene Bild zunächst einmal geglättet (Abbildung 3.13 (b)). Im Anschluss wird der Pixel des Bildes mit der größten Intensität bestimmt und mit einem konstanten Schwellwert verglichen. Liegt die Intensität des bestimmten Pixels unterhalb dieses Schwellwertes, so wird davon ausgegangen, dass sich die Kugel nicht im Bild befindet und die weitere Verarbeitung des Bildes abgebrochen.
Überschreitet die Intensität des Pixels diesen Schwellwert, werden die Koordinaten des Pixels als erste Schätzung der Kugelposition verwendet (Abbildung 3.13 (c)). Diese erste Schätzung wird daraufhin mittels angrenzender Pixel ähnlicher Intensität verfeinert (Abbildung 3.13 (d)). Dazu werden alle benachbarten Pixel mit einem maximalen Intensitätsunterschied von 20 ausgewählt und die Kugelposition aus den Mittelwerten der Koordinaten aller dieser Pixel bestimmt (Abbildung 3.13 (e)). Zur Verringerung des benötigten Rechenaufwandes wird, wenn die Kugel im vorherigen Durchgang erfolgreich erkannt wurde, der zu durchsuchen-

[7] http://www.intel.com



(a) Kamerabild (b) nach der Glättung



(c) hellster Pixel (d) ähnliche Pixel (e) detektierte Kugelposition

Abbildung 3.13: Grobe Darstellung des Ablaufs der Detektion der Kugelposition (*Erläuterung:* (a) Kamerabild; (b) geglättetes Bild; (c) Bestimmung des hellsten Pixels; (c) Verfeinerung der Kugelposition; (e) detektierte Kugelposition)

de Bereich des Bildes auf einen Bereich von 101 mal 101 Pixel um die erkannte Kugelposition beschränkt.

3.2.3.3 Umrechnung der Pixelkoordinaten der Kugel in Koordinaten bezüglich des Spielbretts des BRIO®-Testbeds

Im Anschluß an die Bestimmung der Position der Kugel im Kamerabild müssen die erhaltenen Pixelkoordinaten in Koordinaten bezüglich des Spielbretts umgerechnet werden. Zu diesem Zweck werden während der Kalibrierung des BRIO®-Testbeds die Positionen der vier Ecken des Spielbretts im Kamerabild mittels eines *Harris-Corner-Detectors* [6] bestimmt (Abbildung 3.14). Hierbei wird davon ausgegangen, dass die vier Ecken des Spielbretts die stärksten und vom Bildmittelpunkt am weitesten entfernten Ecken oben links, oben rechts, unten links sowie unten rechts im Bild sind.

Aus den Pixelkoordinaten der vier Referenzpunkte und den bekannten Abmaßen des Spielbretts kann anschließend aufgrund der Überbestimmtheit des Systems die Abbildungsmatrix M zur Umrechnung der Pixelkoordinaten (x_{Pixel}, y_{Pixel}) der Kugel im Kamerabild auf die korrespondierenden Brettkoordinaten (x_{Brett}, y_{Brett})

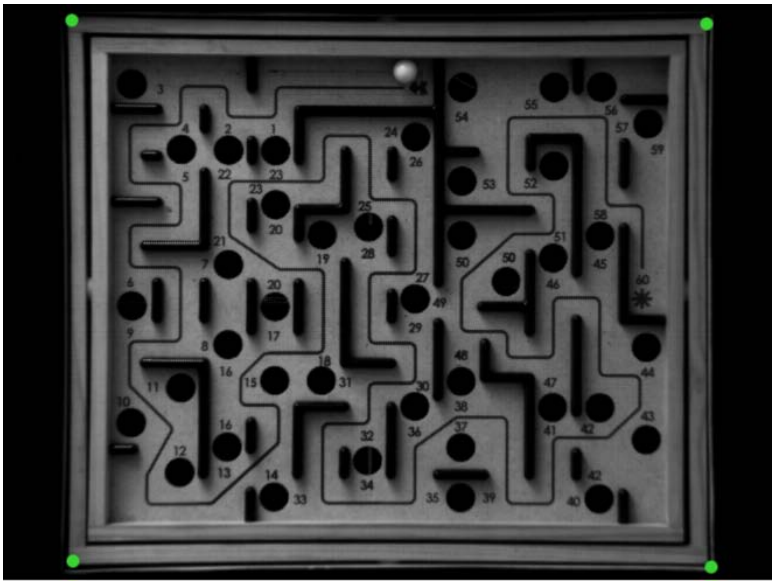


Abbildung 3.14: Detektion der vier Ecken des Spielbretts mittels eines Harris-Corner-Detectors [6] als Referenzpunkte zur Umrechnung der Pixelkoordinaten des Kamerabildes in Koordinaten bezüglich des Spielbretts

ermittelt werden. Hierbei dient die linke obere Ecke des Spielbretts als Koordinatenursprung beider Koordinatensysteme.

$$\begin{pmatrix} x_{Brett} \\ y_{Brett} \end{pmatrix} = \begin{pmatrix} x_{Pixel} \\ y_{Pixel} \end{pmatrix} \cdot M \tag{3.3}$$

Die Umrechnung der detektierten Pixelkoordinaten in die entsprechenden Brettkoordinaten kann anschließend durch einfache Multiplikation des Spaltenvektors der Pixelkoordinaten mit der Abbildungsmatrix M erfolgen (Formel (3.3)). In (3.4) ist eine beispielhafte Umrechnung der Pixelkoordinaten $(311, 354)$ auf die entsprechenden Brettkoordinaten $(0,18m, -0,20m)$ gezeigt.

$$\begin{pmatrix} x_{Brett} \\ y_{Brett} \end{pmatrix} = \begin{pmatrix} 311 \\ 345 \end{pmatrix} \cdot \begin{pmatrix} 0,000584 & -0,000002 \\ -0,000007 & -0,000586 \end{pmatrix} = \begin{pmatrix} 0,18m \\ -0,20m \end{pmatrix} \tag{3.4}$$

3.2.3.4 Schätzen der Kugelposition sowie der -geschwindigkeit

Da das direkte Messen der Geschwindigkeit der Kugel des BRIO®-Testbeds nur sehr aufwendig zu realisieren wäre, wird ein Zustandsschätzer zur Bestimmung der Kugelgeschwindigkeit eingesetzt. Neben der Kugelgeschwindigkeit wird dieser ebenfalls zum Schätzen der Position der Kugel verwendet. Der im BRIO®-Testbed genutzte Zustandsschätzer ist ein modifizierter *Kalman-Filter*.

Der Kalman-Filter wurde 1960 von Rudolf Emil Kálmán in [11] vorgestellt. Mit

Hilfe des Filters ist es möglich den vergangenen, aktuellen oder zukünftigen Zustand eines linearen Prozesses effizient zu schätzen, sogar wenn der zu schätzende Prozess nur unvollständig bekannt ist.

Der Kalman-Filter verwendet zum Schätzen des Zustandes eines Prozesses eine Art Rückkopplungssteuerung. Dabei schätzt der Filter zunächst den Zustand des jeweiligen Prozesses (*Prediktion*) und erhält anschließend eine Rückkopplung in Form von verrauschten Messwerten der geschätzten Größen. Unter Berücksichtigung der Abweichung des gemessenen vom geschätzten Zustand des Prozesses passt der Kalman-Filter sein internes Modell des zu schätzenden Prozesses entsprechend an (*Korrektur*). Eine genauere Einführung und praktische Beispiele zum Thema Kalman-Filter bietet [25] von Greg Welch und Gary Bishop.

Vor Probleme wird der Kalman-Filter jedoch durch das Abprallen bzw. Ablenken der Kugel an einer Wand des Spielbretts gestellt. Hierbei verändert sich die Bewegungsrichtung und -geschwindigkeit der Kugel sehr stark. Trifft z.B. die Kugel im rechten Winkel auf eine Wand, so bewegt sich die Kugel nach dem Zusammenprall mit verminderter Geschwindigkeit in die entgegengesetzte Richtung weiter. Der Kalman-Filter jedoch wird einige *Korrektur*-Phasen benötigen, bevor er sich an die neue Bewegungsrichtung der Kugel angepasst hat.
Um diese Anpassung des internen Modells zu beschleunigen, wurde der Kalman-Filter leicht modifiziert.
Nunmehr werden die gemessenen Kugelpositionen auf ein abruptes Umkehren der Bewegungsrichtung der Kugel jeweils in x- als auch in y-Richtung untersucht. Wird ein solches Abprallen der Kugel von einer Wand detektiert, wird der Betrag ihrer Geschwindigkeit halbiert und ihre Bewegungsrichtung im internen Modell des Filters invertiert.

3.3 Simulation des BRIO®-Testbeds

Nachdem nun in Kapitel 3.2 das BRIO®-Testbed mit seinen einzelnen Komponenten und Verfahren vorgestellt wurde, soll nun die Simulation des Systems kurz präsentiert werden.

3.3.1 Machinae Arte Robotum Simulans - MARS

Die Simulation des BRIO®-Testbeds wird durch eine am DFKI-Labor Bremen[8] entwickelte Simulationsumgebung realisiert. MARS (*Machinae Arte Robotum Simulans*) ist eine physikbasierte Robotersimulation. Sie kann zur Simulation des Verhaltens von Robotern und als Visualisierungstool genutzt werden. Aufgrund seines modularen Aufbaus ist MARS in der Lage, die verschiedensten Roboter und Umgebungsbedingungen nachzubilden.

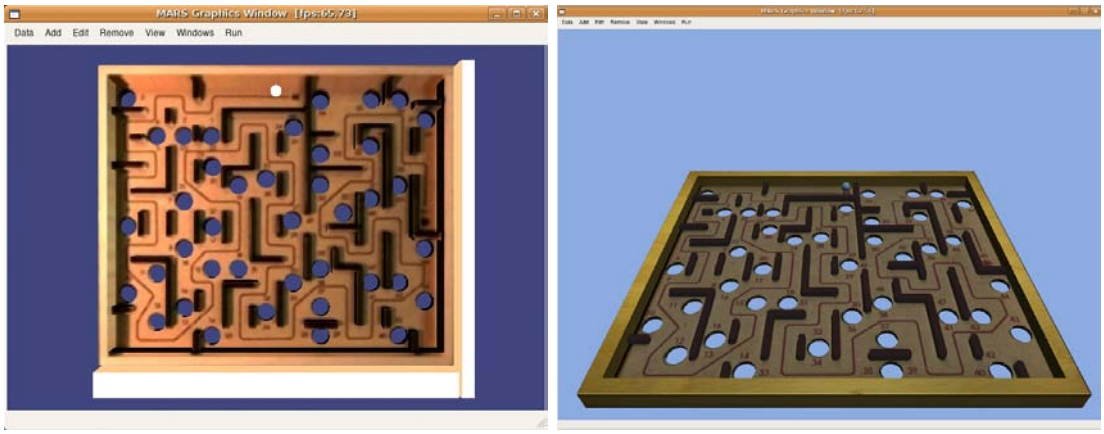
Aktuell wird die *Open Dynamics Engine*[9] (*ODE*) als Physiksimulation eingesetzt. Aber auch der Einsatz anderer Physikbibliotheken ist möglich. ODE ist eine freie C/C++ Bibliothek, die für das Simulieren der Dynamik von starren Körpern in der virtuellen Realität eingesetzt wird [22]. Laut Herstellerangaben ist die Physik-Engine schnell, leistungsstark, robust, flexibel und hat eine integrierte Kollisionsabfrage. Sie wurde hauptsächlich von Russell Smith entwickelt und wird aktuell unter Mitwirkung anderer weiterentwickelt.

3.3.2 Realisierung des BRIO®-Labyrinth-Spiels in MARS

Zeitgleich mit dieser Master-Arbeit wurde von Mitarbeitern des DFKI Bremens kontinuierlich an der MARS-Simulation und der Umsetzung des BRIO®-Labyrinth-Spiels gearbeitet. In dem nun folgenden Abschnitt wird ein grober Überblick über die wichtigsten Anpassungen und Veränderungen bezüglich der Simulation des BRIO®-Spiels gegeben.

Zur Simulation des BRIO®-Labyrinth-Spiels wurde zunächst das Spielbrett innerhalb der MARS-Simulation nachgebildet (Abbildung 3.15 (b)). Dazu wurde ein 3D-Modell des Spielbretts anhand eines Fotos der Spielfläche erstellt. Zum Kippen und Neigen des Labyrinths wurden jeweils zwei virtuelle Motoren mit dem Spielbrett verbunden. Zusätzlich wurden zwei quaderförmige Objekte hinzugefügt, um die beiden Drehachsen des Spielbretts zu realisieren. Diese beiden virtuellen „Holzlatten“ verbinden die beiden Motoren der Simulation.
Die erste Version dieser BRIO®-Umgebung war eine relativ ungenaue Simulation des BRIO®-Testbeds. Das gesamte System war um den Faktor 10 größer skaliert. Zudem wurde bei den Physikberechnungen nicht das reale Gewicht der einzelnen Elemente berücksichtigt. Somit war es zunächst notwendig die Abmaße der einzelnen Elemente der Simulationsumgebung an die realen Werte anzupassen. Des

[8] http://robotik.dfki-bremen.de/
[9] http://www.ode.org/



(a) alt (b) neu

Abbildung 3.15: Alte (a) und neue (b) BRIO®-Simulations-Umgebung

Weiteren wurde das Gewicht der Kugel und des Spielbretts bestimmt und die entsprechenden Werte in die Simulationsumgebung übernommen.
Ein weiteres Problem stellte die grobe Nachbildung der Labyrinthstruktur durch das erste 3D-Modell dar. Aufgrund der relativ geringen Auflösung des zugrundeliegenden Fotos der Spielfläche wies das Modell einige Ungenauigkeiten auf. Aus diesem Grund wurde ein zweites, detailgetreueres 3D-Modell des Spielbretts erstellt. Dazu wurde das reale Spielbrett hochauflösend eingescannt und unter Berücksichtigung der Abmaße und Form des realen Labyrinths nachgebildet (Abbildung 3.15 (b)). Zusätzlich wurde die Positionierung der Motoren überdacht, so dass diese jetzt gemeinsam im Schnittpunkt der beiden Drehachsen des Spielbretts angebracht sind und somit die beiden „Holzlatten“ nicht länger benötigt werden.

Kapitel 4

Konzept der Arbeit

Im Rahmen dieser Master-Arbeit soll das Simulation-Reality-Gap (Kapitel 1.2) zwischen dem BRIO®-Testbed (Kapitel 3.2) und der Simulation des BRIO®-Labyrinth-Spiels zum einen (Kapitel 3.3) untersucht und zum anderen durch geeignete Anpassungen ausgewählter Komponenten eines bzw. beider Systeme minimiert werden. Da eine vollständige Untersuchung aller Komponenten und deren Einfluss auf das Simulation-Reality-Gap nicht mit dem zeitlichen Rahmen einer Master-Arbeit vereinbar ist, wird der Umfang der untersuchten Komponenten eingegrenzt. Dementsprechend sollen nur die Winkelmessung, die Ansteuerung der Servomotoren, einige Komponenten der Bestimmung der Kugelposition des BRIO®-Testbeds sowie die Motorparameter der Simulation untersucht werden.
Parallel zur Untersuchung der einzelnen Komponenten des BRIO®-Testbeds wird versucht ein entsprechendes Rauschen der einzelnen Sensoren und Aktuatoren zu modellieren. Eine Methode zur Minimierung des Simulation-Reality-Gaps besteht in der Berücksichtigung des Rauschens der realen Sensoren und Aktuatoren bereits während des Trainings des Agenten innerhalb der Simulation (Kapitel 2.1.2). Aus diesem Grund wird versucht das Rauschen der einzelnen Bauteile empirisch zu bestimmen und mit geeigneten Methoden nachzubilden, um in anschließenden Untersuchungen die Unsicherheit der realen Bauteile während des Trainings des Agenten zu berücksichtigen.

Als erstes soll die Messung der jeweiligen Auslenkung des Spielbretts des BRIO®-Testbeds (Kapitel 5) näher betrachtet werden. Aufgrund des Aufbaus des Spiels können die Brettwinkel des Spielbretts nicht direkt gemessen werden. Stattdessen werden die Winkel der beiden Achsen mit Hilfe von Potentiometern gemessen und anschließend die entsprechenden Brettwinkel daraus berechnet. In diesem Zusammenhang ist beabsichtigt, zunächst das Rauschen der Potentiometerspannungen für verschiedene Auslenkungen des Spielbretts aufzuzeichnen und zu analysieren.

Anschließend soll anhand des Rauschens der gemessenen Spannungen der Potentiometer ein entsprechendes Rauschen der Brettwinkel des BRIO®-Testbeds modelliert werden.

Im zweiten Schritt ist die Positioniergenauigkeit des BRIO®-Testbeds (Kapitel 6) näher zu untersuchen. Die Positioniergenauigkeit beschreibt hierbei die Fähigkeit der Servomotoren das Spielbrett wiederholbar entsprechend vorgegebener Brettwinkel auszulenken. Entgegen den idealen Bedingungen innerhalb der Simulation weichen die erreichten Brettwinkel des BRIO®-Testbeds zumeist von den vorgegebenen Brettwinkeln ab. Im Zuge dieser Master-Arbeit soll der Positionierungsfehler des BRIO®-Testbeds analysiert, eventuelle Fehlerquellen identifiziert und entsprechende Methoden zur Reduzierung des Fehlers implementiert werden.

Als nächstes wird die Bestimmung der Kugelposition des BRIO®-Testbeds (Kapitel 7) genauer begutachtet. Aktuell erfolgt die Bestimmung der Kugelposition in zwei Schritten: Zuerst wird die Position der Kugel im Kamerabild bestimmt und anschließend werden die Pixelkoordinaten der detektierten Kugelposition in Spielbrettkoordinaten umgerechnet. Im Rahmen dieser Master-Arbeit soll das aktuelle Verfahren der Kugeldetektion hinsichtlich ihrer Messunsicherheit untersucht und mögliche Anpassungen zur Verringerung dieser realisiert werden.

Außerdem zeigte sich eine relativ große Abweichung zwischen dem Verhalten der Servomotoren und den Motoren der Simulation. Das Verhalten der simulierten Motoren kann mittels entsprechender Parameter beeinflusst werden. Anhand einer Optimierung dieser Motorparameter soll versucht werden, die bestehende Abweichung zwischen den Motoren beider Systeme zu reduzieren (Kapitel 8).

Abschließend ist das Verhalten der beiden Systeme miteinander zu vergleichen (Kapitel 9). Von den im State of the Art (Kapitel 2) vorgestellten Verfahren wurde meist die Abweichung zwischen dem simulierten und dem realen System anhand der Performance des Agenten innerhalb der Simulation und nach der Übertragung auf das reale System bestimmt. Vorherrschend wurden hierzu die erreichten Fitnesswerte des Agenten innerhalb der simulierten Umgebung mit den in der Realität erreichten Werten verglichen.
In dieser Arbeit wird jedoch ein anderer Ansatz verfolgt. Anstatt das Simulation-Reality-Gap anhand der Fitness der Agenten in Simulation und Realität zu bestimmen, soll das Verhalten beider Systeme direkt miteinander verglichen werden. Zu diesem Zweck ist angestrebt im Verlauf dieser Master-Arbeit ein Ver-

fahren zu entwickelt, mit dessen Hilfe sich quantitative Aussagen bezüglich des Simulation-Reality-Gap zwischen dem realen und simulierten System treffen lassen. Abschließend sollen mit der entwickelten Methode die Auswirkungen der einzelnen vorgenommenen Anpassungen an Simulation und Testbed hinsichtlich ihrer Auswirkungen auf das Simulation-Reality-Gap untersucht werden.

Kapitel 5

Untersuchung der gemessenen Brettwinkel

Als erstes wurden die gemessenen Brettwinkel des BRIO®-Testbeds im Hinblick auf ihr Rauschen untersucht. Bedingt durch den Aufbau des BRIO®-Labyrinth-Spiels können die Winkel des Spielbretts nicht direkt gemessen werden. Stattdessen werden die Winkel der beiden Achsen des BRIO®-Labyrinth-Spiels mit Hilfe von Potentiometern bestimmt und anschließend die Winkel des Spielbretts daraus berechnet (Kapitel 3.2.3.1). Die Verbindung der beiden Achsen des Spiels mit dem Spielbrett, welche jeweils durch einen robusten Bindfaden hergestellt wird, führt bei einer Drehung einer der beiden Achsen dazu, dass sich die entsprechenden Winkel des Spielbretts verändern (Abbildung 3.12). Bei der Umrechnung des Winkels der Spielachse in den zugehörigen Brettwinkel wird davon ausgegangen, dass ein linearer Zusammenhang zwischen den Winkeln der Achsen des BRIO®-Labyrinth-Spiels und der Neigung des Spielbretts besteht.

Das nun folgende Kapitel beschreibt die einzelnen Schritte bei der Bestimmung der Brettwinkel des BRIO®-Labyrinth-Spiels genauer. Dazu wird zunächst das Rauschen der Potentiometerspannungen untersucht und anschließend aus dem Rauschen der gemessenen Spannungen ein entsprechendes Rauschen der beiden Brettwinkel modelliert. Dieses Rauschmodell könnte später beim Trainieren eines Agenten innerhalb der Simulation des BRIO®-Labyrinth-Spiels genutzt werden, um die Unsicherheit der Brettwinkel des BRIO®-Testbeds nachzubilden.

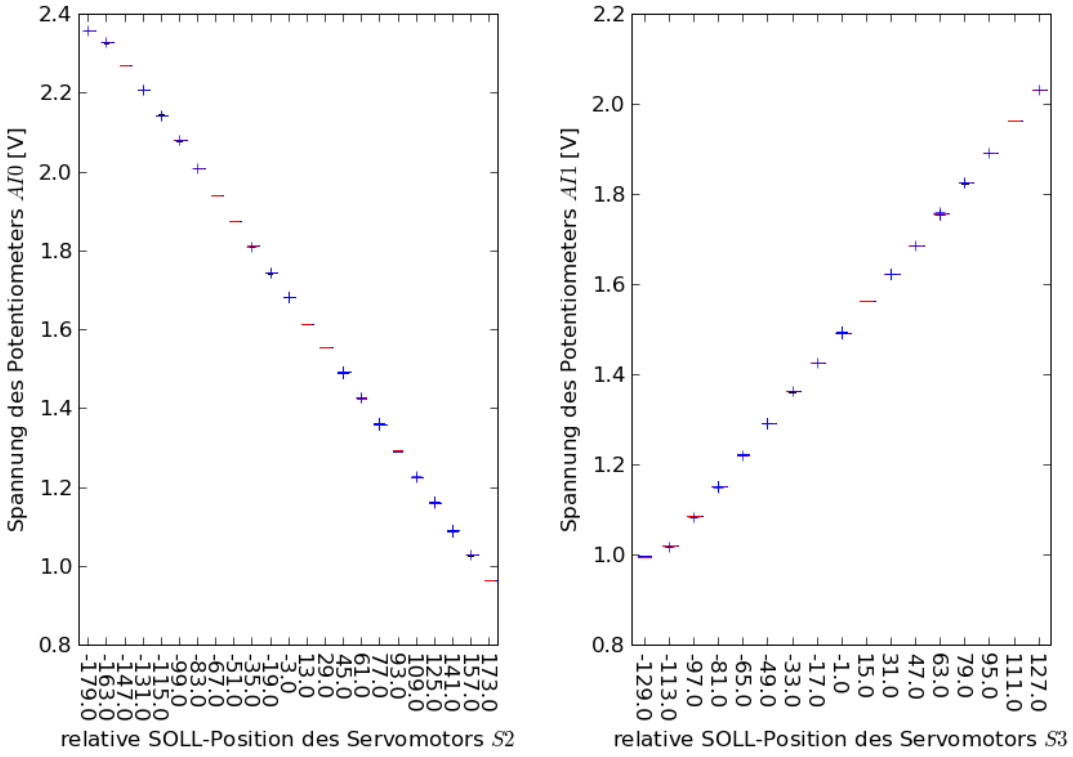
5.1 Untersuchung des Rauschens der Potentiometerspannungen

Zur Modellierung des Rauschens der Brettwinkel wurde das Spielbrett zunächst mittels der beiden Servomotoren $S2$ und $S3$ verschieden ausgelenkt und das Rauschen der gemessenen Spannungswerte in den einzelnen Auslenkungen bestimmt. Mit Hilfe der Messung der Spannungswerte an verschiedenen Servopositionen sollte eine vorhandene Abhängigkeit zwischen der Auslenkung des Spielbretts und dem Rauschen der Potentiometerspannungen erkannt bzw. ausgeschlossen werden können.

	kleinste erlaubte Servoposition	Nullposition	größte erlaubte Servoposition
$S2$	330	509	688
$S3$	353	482	611

Tabelle 5.1: Kleinste und größte erlaubte Position sowie Nullposition der Servomotoren

Als erstes wurde eine Kalibrierung des BRIO®-Testbeds (Kapitel 3.2.3.1) durchgeführt, um die Nullposition sowie den Bereich erlaubter Servopositionen (Tabelle 5.1) zu bestimmen. Aus den erlaubten Positionen des jeweiligen Servomotors wurde eine Menge von Testpositionen, beginnend mit der kleinsten Position mit einer Schrittweite von 16 Positionen, ausgewählt. Es ergaben sich für den Servomotor $S2$ 23 und für den Servomotor $S3$ 17 Testpositionen. Anschließend wurde jede dieser Testpositionen angefahren und jeweils 100 Mal die zugehörige Potentiometerspannung gemessen und aufgezeichnet. In Abbildung 5.1 sind die Verteilungen der gemessenen Spannungen an den einzelnen Testpositionen beider Servomotoren mittels Boxplots (Anhang A) grafisch dargestellt.
In Folge einer zum Potentiometer $AI0$ spiegelverkehrten Beschaltung des Potentiometers $AI1$ verhalten sich die gemessenen Potentiometerspannungen in Abhängigkeit von der relativen Soll-Position des jeweiligen Servomotors invers. Während sich die Werte der gemessenen Spannungen des Potentiometers $AI0$ mit zunehmend größeren Soll-Positionen des Servomotors $S2$ verkleinern, steigen die Spannungswerte des Potentiometers $AI1$ mit größer werdenden Soll-Positionen des Servomotors $S3$ an.
Dadurch, dass die Spannungswerte der einzelnen Messreihen, verglichen mit dem Wertebereich aller gemessenen Spannungen, nur sehr gering rauschen, sind in Abbildung 5.1 die Boxplots der einzelnen Verteilungen kaum zu erkennen. Aus diesem Grund und zur Verbesserung der Vergleichbarkeit der einzelnen Messreihen untereinander wurde eine Normierung der Daten durchgeführt. Dazu wurde



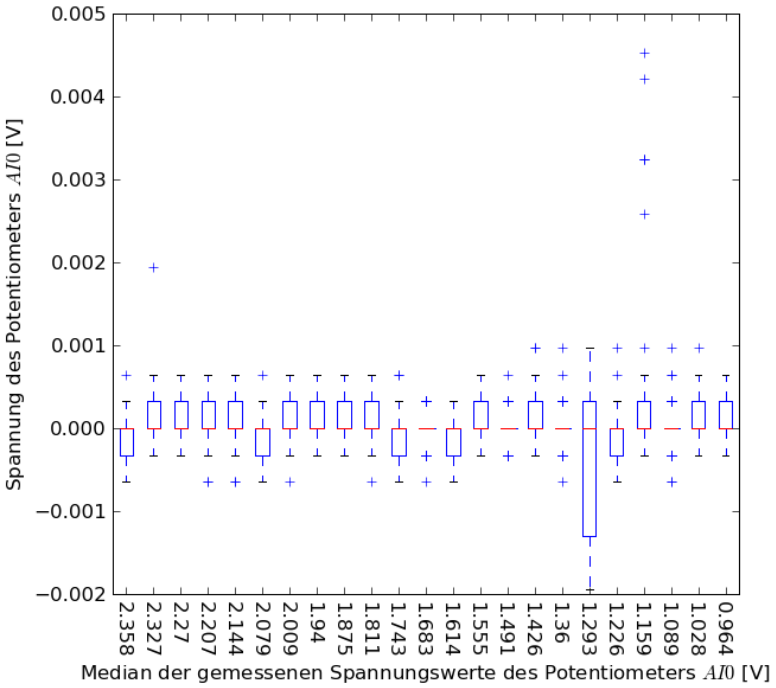
(a) $AI0$ (b) $AI1$

Abbildung 5.1: Boxplot der Verteilung der Potentiometerspannungen aufgetragen über die relativen Soll-Positionen der Servomotoren für $AI0$ (a) bzw. $AI1$ (b)

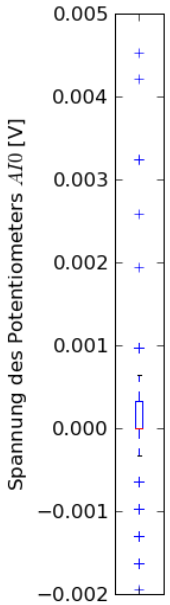
der jeweilige Median aller gemessenen Spannungen einer Testposition ermittelt und anschließend dieser von allen Werten der Messreihe abgezogen. In Abbildung 5.2 (a) sind die normierten Verteilungen der verrauschten Spannungswerte der einzelnen Messreihen des Potentiometers $AI0$ grafisch dargestellt. Analog sind die Spannungswerte des Potentiometer $AI1$ in Abbildung 5.2 (c) dargestellt. In beiden Abbildungen konnte keine Winkelabhängigkeit der normierten Spannungswerte festgestellt werden.

Die meisten normierten Verteilungen der gemessenen Spannungen schwanken mit Ausnahme einiger weniger Ausreißer in einem sehr kleinen Intervall von $-640\mu V$ bis $640\mu V$ um den Wert $0V$. Die Werte der normierten Spannungen betragen hierbei ein Vielfaches des maximalen Auflösungsvermögens des eingesetzten Analog-Digital-Wandlers (Kapitel 3.2.2.5) von $320\mu V$.

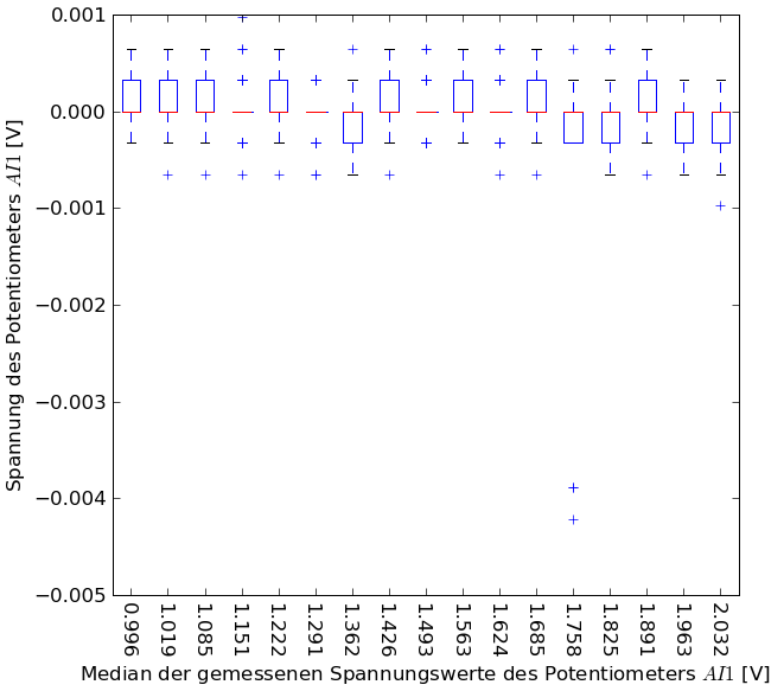
Aufgrund der Unabhängikeit des Rauschens der Potentiometerspannungen von den Brettwinkel wurden zur Modellierung des Rauschens alle Verteilungen der normierten Messreihen des entsprechenden Potentiometers jeweils zu einer Gesamtverteilung akkumuliert (Abbildung 5.2 (b) und (d)). In Abbildung 5.3 sind die beiden entstehenden Verteilungen der normierten Spannungswerte mittels



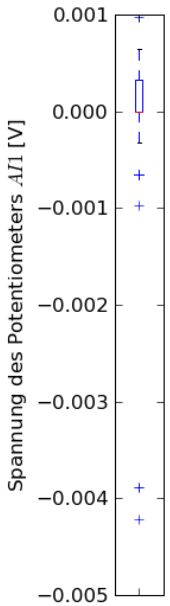
(a) $AI0$ (einzeln)



(b) $AI0$ (gesamt)

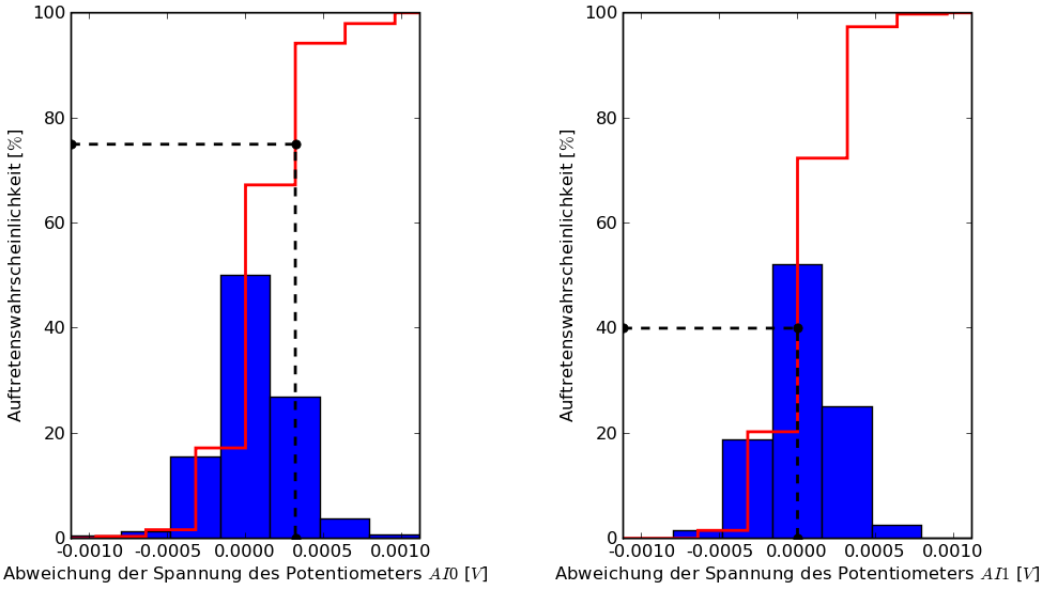


(c) $AI1$ (einzeln)



(d) $AI1$ (gesamt)

Abbildung 5.2: Boxplot der Verteilung der gemessenen Spannungen über den Potentiometern $AI0$ und $AI1$ um den Median der einzelnen Messreihen (a) und (c) bzw. aller Messreihen zusammen (b) und (d)



(a) $AI0$ (b) $AI1$

Abbildung 5.3: Histogramme des Rauschens der Potentiometerspannungen für $AI0$ (a) und $AI1$ (b) (*Erklärung:* Die rote Linie repräsentiert dabei die kumulative Verteilungsfunktion des jeweiligen Servomotors. Mit ihrer Hilfe kann die Verteilung der Spannungswerte nachgebildet werden. Dazu wird zunächst eine gleichverteilte Zufallszahl aus dem Intervall [0,1] gezogen und anschließend deren Wert auf den zugehörigen Spannungswert abgebildet (schwarze gestrichelte Linie).)

zweier Histogramme dargestellt. Hierbei wurden alle normierten Spannungswerte ausserhalb des Intervalls von $[-1,12mV, +1,12mV]$ als Ausreisser gewertet und bei der Erstellung der Histogramme nicht berücksichtigt. Beide so erhaltenen Histogramme zeigen eine ähnliche Verteilung der Auftretenswahrscheinlichkeiten der einzelnen normierten Spannungswerte (Tabelle 5.2). Wie bereits die einzelnen Messreihen, schwanken die Potentiometerwerte hauptsächlich zwischen den Werten $-320\mu V$, $0\mu V$ und $+320\mu V$ um ihren Median.

Mit den so ermittelten Auftretenswahrscheinlichkeiten kann das Rauschen der gemessenen Spannungen entsprechend modelliert werden. Zu diesem Zweck wurde das in [23] vorgestellte Verfahren zum Ziehen von Zufallszahlen aus verschiedenen diskreten Verteilungen eingesetzt. Hierbei wird zufällig eine Zahl x aus dem Einheitsintervall $[0,1]$ ausgewählt und anschließend mittels einer kumulativen Verteilungsfunktion (rote Linie in Abbildung 5.3) auf den entsprechenden Spannungswert (schwarze gestrichelte Linie in Abbildung 5.3) abgebildet. Für den Potentiometer $AI0$ kann somit das Rauschen der gemessenen Spannungen mathematisch mittels (5.1) unter Berücksichtigung der Eigenheiten des Potentio-

Abweichung des Potentiometerwertes $[\mu V]$	Auftretenswahrscheinlichkeit $AI0$ [%]	Auftretenswahrscheinlichkeit $AI1$ [%]
-960	0,391	0,059
-640	1,261	1,471
-320	15,565	18,765
0	50,043	52,059
320	26,957	25,000
640	3,739	2,412
960	0,696	0,059

Tabelle 5.2: Auftretenswahrscheinlichkeiten der Spannungswerte des Rauschens der Potentiometer $AI0$ und $AI1$ um ihren Median

meters modelliert werden. Entsprechend ergibt sich für den Potentiometer $AI1$ das Modell des Rauschens der Potentiometerspannung zu (5.2).

$$n_{AI0}(x \sim U(0,1)) = \begin{cases} -960\mu V & \text{für } x \leq 0,00391 \\ -640\mu V & \text{für } 0,00391 < x \leq 0,01652 \\ -320\mu V & \text{für } 0,01652 < x \leq 0,17217 \\ 0\mu V & \text{für } 0,17217 < x \leq 0,67260 \\ 320\mu V & \text{für } 0,67260 < x \leq 0,94217 \\ 640\mu V & \text{für } 0,94217 < x \leq 0,97956 \\ 960\mu V & \text{für } 0,97956 < x \end{cases} \tag{5.1}$$

$$n_{AI1}(x \sim U(0,1)) = \begin{cases} -960\mu V & \text{für } x \leq 0,00059 \\ -640\mu V & \text{für } 0,00059 < x \leq 0,01530 \\ -320\mu V & \text{für } 0,01530 < x \leq 0,20295 \\ 0\mu V & \text{für } 0,20295 < x \leq 0,72354 \\ 320\mu V & \text{für } 0,72354 < x \leq 0,97354 \\ 640\mu V & \text{für } 0,97354 < x \leq 0,99766 \\ 960\mu V & \text{für } 0,99766 < x \end{cases} \tag{5.2}$$

5.2 Rauschen der Brettwinkel

Im Anschluß muss das Rauschen der gemessenen Spannungswerte (5.1) und (5.2) unter Verwendung der beiden Formeln (3.1) und (3.2) aus Kapitel 3.2.3.1 in das Rauschen der zugehörigen Brettwinkel überführt werden.

Während die Referenzwinkel fest vorgegeben sind, werden die Referenzspannungen U_{min}, U_{center} und U_{max} erst während der Kalibrierung des Systems ermittelt. Prinzipiell sind zwar auch diese drei Werte durch die feste Verbindung 6der Potentiometer mit den Achsen des BRIO®-Labyrinth-Spiels fest vorgegeben. Um aber eine aufwendige Neukalibrierung nach dem Austauschen eines Potentiometers bzw. dem Ab- und Anbau des alten Potentiometers zu verhindern, werden die Werte erst zur Laufzeit des BRIO®-Testbeds bestimmt. Da jedoch die Bestimmung der Referenzspannungen ebenfalls dem Rauschen der Potentiometerspannungen unterliegt sowie abhängig vom Auslösen des jeweiligen Endschalters ist, können die Spannungswerte von Kalibrierung zu Kalibrierung leicht variieren. Um die Auswirkungen dieser beiden Effekte zu minimieren, wurde das BRIO®-Testbed jeweils 10 Mal kalibriert und anschließend der Mittelwert der Referenzspannungen für beide Potentiometer berechnet. Die berechneten Mittelwerte der einzelnen Referenzspannungen wurden anschließend auf den nächsten, von den A/D-Wandlern aufzulösenden Spannungswert gerundet. Es ergaben sich folgende gemittelte und gerundete Referenzspannungen beider Potentiometer (Tabelle 5.3).

	U_{min}	U_{center}	U_{max}
$AI0$	1,17216	2,18720	3,15488
$AI1$	1,39936	2,14400	2,89472

Tabelle 5.3: Gemittelte und gerundete Referenzspannungen der Potentiometer

Mit Hilfe der so erhaltenen Referenzspannungen und der beiden Berechnungsvorschriften (3.1) und (3.2) kann das Rauschen der gemessenen Spannungswerte U_{AI0} und U_{AI1} in das Rauschen des jeweiligen Brettwinkels ϕ_X und ϕ_Y des BRIO®-Testbeds überführt werden. Dazu wurden die einzelnen Spannungspegel des Rauschens der Potentiometer ($-960\mu V$, $-640\mu V$, $-320\mu V$, $0\mu V$, $320\mu V$, $640\mu V$, $960\mu V$) in den entsprechenden Winkel umgerechnet. Aufgrund der zweigeteilten Berechnung der Winkel einer Drehachse für Spannungen größer und kleiner der mittleren Referenzspannung, ergeben sich somit vier verschiedene Rauschfunktionen. Das Rauschen des Brettwinkels ϕ_X kann für Winkel kleiner 0° mittels der Formel (5.3) und für Winkel größer gleich 0° mittels der Formel (5.4) modelliert werden. Dementsprechend modellieren die Formeln (5.5) und (5.6) das Rauschen

des Brettwinkels ϕ_Y.

$$n_{\phi_X<0^\circ}(x \sim U(0,1)) = \begin{cases} -0,008265^\circ & \text{für } x \leq 0,00391 \\ -0,005510^\circ & \text{für } 0,00391 < x \leq 0,01652 \\ -0,002755^\circ & \text{für } 0,01652 < x \leq 0,17217 \\ 0,000000^\circ & \text{für } 0,17217 < x \leq 0,67260 \\ 0,002755^\circ & \text{für } 0,67260 < x \leq 0,94217 \\ 0,005510^\circ & \text{für } 0,94217 < x \leq 0,97956 \\ 0,008265^\circ & \text{für } 0,97956 < x \end{cases} \tag{5.3}$$

$$n_{\phi_X\geq0^\circ}(x \sim U(0,1)) = \begin{cases} -0,008616^\circ & \text{für } x \leq 0,00391 \\ -0,005744^\circ & \text{für } 0,00391 < x \leq 0,01652 \\ -0,002872^\circ & \text{für } 0,01652 < x \leq 0,17217 \\ 0,000000^\circ & \text{für } 0,17217 < x \leq 0,67260 \\ 0,002872^\circ & \text{für } 0,67260 < x \leq 0,94217 \\ 0,005744^\circ & \text{für } 0,94217 < x \leq 0,97956 \\ 0,008616^\circ & \text{für } 0,97956 < x \end{cases} \tag{5.4}$$

$$n_{\phi_Y<0^\circ}(x \sim U(0,1)) = \begin{cases} -0,006999^\circ & \text{für } x \leq 0,00059 \\ -0,004666^\circ & \text{für } 0,00059 < x \leq 0,01530 \\ -0,002333^\circ & \text{für } 0,01530 < x \leq 0,20295 \\ 0,000000^\circ & \text{für } 0,20295 < x \leq 0,72354 \\ 0,002333^\circ & \text{für } 0,72354 < x \leq 0,97354 \\ 0,004666^\circ & \text{für } 0,97354 < x \leq 0,99766 \\ 0,006999^\circ & \text{für } 0,99766 < x \end{cases} \tag{5.5}$$

$$n_{\phi_Y\geq0^\circ}(x \sim U(0,1)) = \begin{cases} -0,006561^\circ & \text{für } x \leq 0,00059 \\ -0,004374^\circ & \text{für } 0,00059 < x \leq 0,01530 \\ -0,002187^\circ & \text{für } 0,01530 < x \leq 0,20295 \\ 0,000000^\circ & \text{für } 0,20295 < x \leq 0,72354 \\ 0,002187^\circ & \text{für } 0,72354 < x \leq 0,97354 \\ 0,004374^\circ & \text{für } 0,97354 < x \leq 0,99766 \\ 0,006561^\circ & \text{für } 0,99766 < x \end{cases} \tag{5.6}$$

Kapitel 6

Untersuchung der angefahrenen Brettwinkel

In den nun folgenden Abschnitten soll die Positioniergenauigkeit des Testbeds näher betrachtet werden.
Ziel der durchgeführten Untersuchungen war es, die Fähigkeit des BRIO®-Testbeds, wiederholbar den gleichen Brettwinkel anzufahren, zu untersuchen, wenn notwendig zu verbessern und abschließend ein Rauschmodell zu erstellen.

6.1 Untersuchung der Positioniergenauigkeit der Servomotoren

In einem ersten Schritt wurde dafür zunächst die Positioniergenauigkeit der Servomotoren des BRIO®-Testbeds separat, d.h. unabhängig vom Spiel, untersucht.

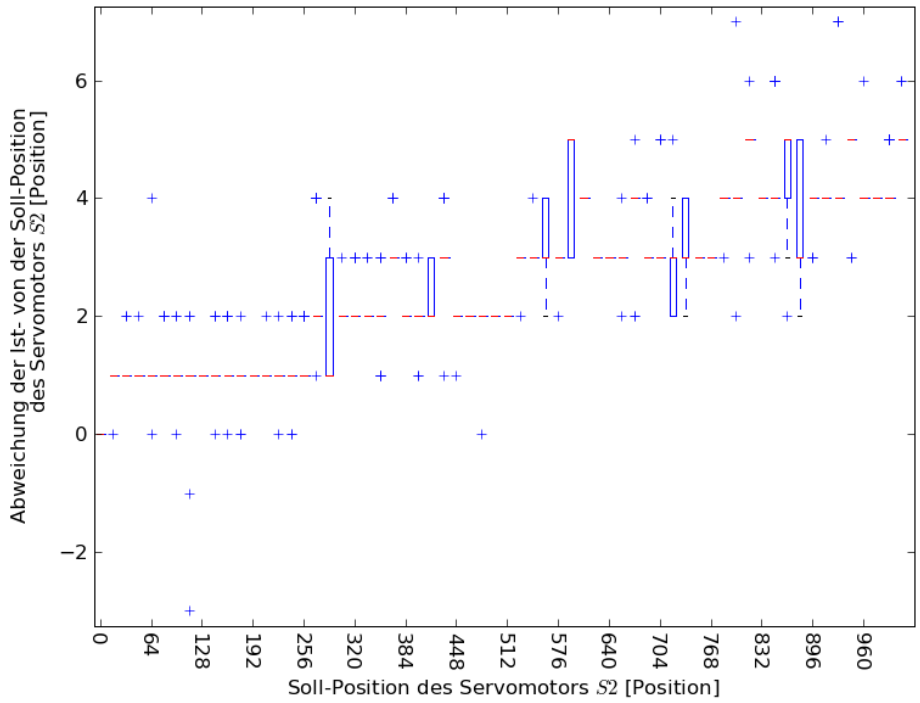
Für beide Servomotoren wurde jeweils folgendes Experiment durchgeführt: Aus allen 1024 möglichen Servopositionen wurden alle Vielfache der Zahl 16 als Testpositionen ausgewählt. Von jeder dieser 64 Testpositionen ausgehend, wurden alle 63 anderen Testpositionen jeweils einmal angefahren. Dadurch wurden die einzelnen Testpositionen aus verschiedenen Richtungen und mit verschieden langen Schrittweiten angefahren. Die Reihenfolge des Anfahrens der einzelnen Testpositionen wurde dabei zufällig gewählt. Für jede dieser 4032 Bewegungen wurden die Soll- und die Ist-Position des jeweiligen Servomotors aufgezeichnet. Die Soll-Position des Servomotors war dabei die vorgegebene Zielposition des Servomotors, wohingegen die Ist-Position aus dem jeweiligen von den internen Potentiometern der Servomotoren zurückgelieferten Wert bestand. Nach dem Aufzeichnen der Soll- und Ist-Position wurde aus allen von der aktuellen Position aus noch nicht

angefahrenen Testpositionen zufällig eine ausgewählt und als nächste angefahren. Korrekturbewegungen waren somit nur notwendig, wenn von der aktuellen Position des Servomotors ausgehend bereits alle anderen Testpositionen angefahren wurden. Somit konnte die Anzahl der notwendigen Korrekturbewegungen auf ein Minimum reduziert werden.

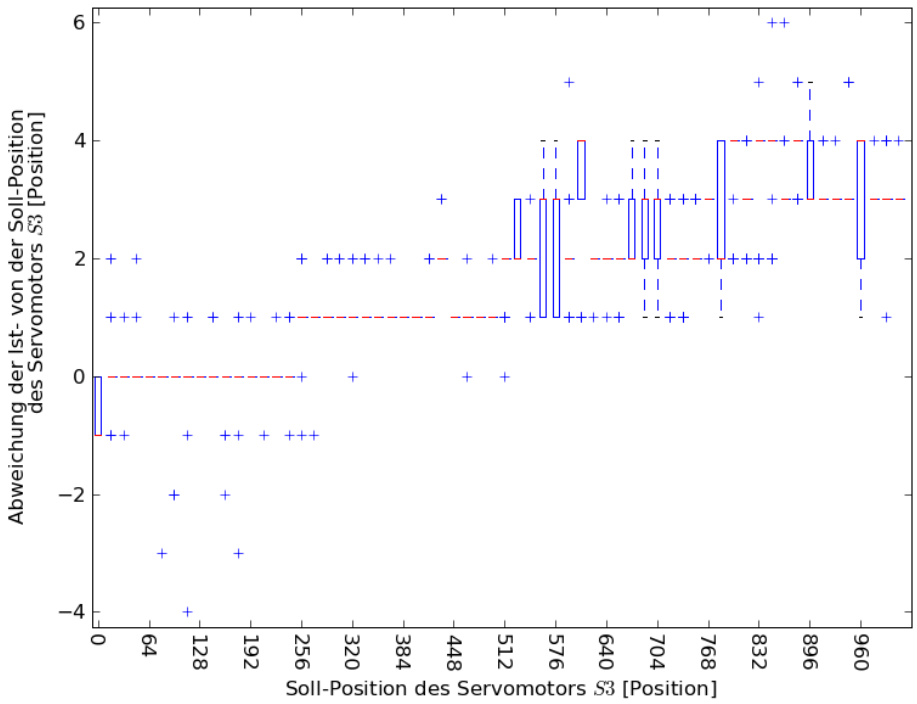
Im nächsten Schritt wurde die Abweichung der einzelnen Ist- von ihrer Soll-Position berechnet. Die Verteilung dieses Positionierungsfehlers der einzelnen Testpositionen ist für beide Servomotoren in Abbildung 6.1 mittels Boxplots (Anhang A) dargestellt.
In beiden Boxplots (Abbildung 6.1) ist eine Abweichung der vorgegebenen Zielposition von der angefahrenen Position der Servomotoren deutlich zu erkennen. Gemittelt über alle Daten eines Servomotors weicht die Ist-Position von der Soll-Position für $S2$ um 2,542 und für $S3$ um 1,580 Servopositionen ab. Zusätzlich steigt der Positionierungsfehler der Servomotoren in Abhängigkeit von der vorgegebenen Zielposition an. Zur besseren Veranschaulichung dieses Sachverhaltes wurde der mittlere Positionierungsfehler der einzelnen Testpositionen berechnet und in Abbildung 6.2 visualisiert. Für beide Servomotoren ist ein Zuwachs des Positionierungsfehlers mit zunehmend größer werdenden Soll-Positionen zu erkennen. Der mittlere Positionierungsfehler des Servomotors $S2$ wächst von einem anfänglichen Wert von 0 Positionen mit Ausnahme einiger Schwankungen stufenweise auf einen Wert von 5 Positionen an. Entsprechend ähnliches Verhalten zeigt der mittlere Positionierungsfehler des Servomotors $S3$, welcher sich von einem Wert von -1 Position zu Beginn auf einen Wert von 3 Positionen erhöht. Dieser Positionierungsfehler beider Servomotoren könnte entweder durch ungeeignete Servoparameter oder aufgrund eines systematischen Fehlers der Servomotoren selbst hervorgerufen werden.

6.2 Kompensation des Positionierungsfehlers der Servomotoren

Aufgrund der festgestellten Abweichung zwischen der vorgegebenen Zielposition und der angefahrenen Position der Servomotoren und der daraus resultierenden Positionierungenauigkeit des BRIO®-Testbeds wurden Untersuchungen zur Reduzierung des Positionierungsfehler der Servomotoren durchgeführt. Im Rahmen dieser Arbeit wurden dazu zwei Ansätze verfolgt. Während der erste Ansatz die Untersuchung bestimmter Servoparameter und deren Einfluss auf die Positionier-



(a) $S2$



(b) $S3$

Abbildung 6.1: Boxplots des Positionierungsfehler der Servomotoren $S2$ (a) und $S3$ (b) aufgetragen über die einzelnen Testpositionen des jeweiligen Servomotors

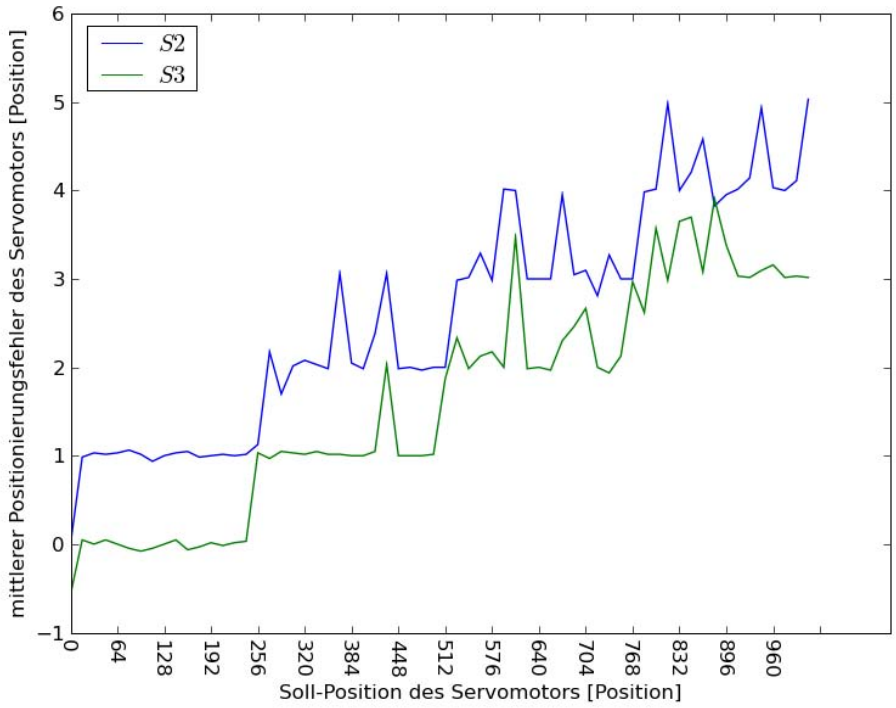


Abbildung 6.2: Darstellung der mittleren Positionierungsfehler des Servomotors $S2$ und $S3$ aufgetragen über die zugehörigen Soll-Positionen

genauigkeit der Servomotoren umfasste, wurde beim zweiten Ansatz die Möglichkeit einer „Korrektur“ der Zielposition vor der Ansteuerung der Servomotoren geprüft.

6.2.1 Einfluss der Servoparameter auf die Positioniergenauigkeit

Wie bereits erwähnt, sollten bei diesem ersten Ansatz zur Erhöhung der Positioniergenauigkeit der Servomotoren ausgewählte Servoparameter näher untersucht werden. Entsprechend dem Handbuch der Servomotoren [21] ist deren Positioniergenauigkeit vor allem von den Parametern *Compliance Slope*, *Compliance Margin* und *Punch* abhängig.

6.2.1.1 Compliance Margin

Zuerst wurden die Auswirkungen des Parameters Compliance Margin auf die Positioniergenauigkeit der Servomotoren untersucht. Der Parameter Compliance Margin beeinflusst die maximal erlaubte Abweichung der Ist-Position des Servomotors von der vorgegebenen Zielposition. Mit Hilfe dieses Parameters ist es möglich, einen gewissen Toleranzbereich um die Zielposition zu definieren und

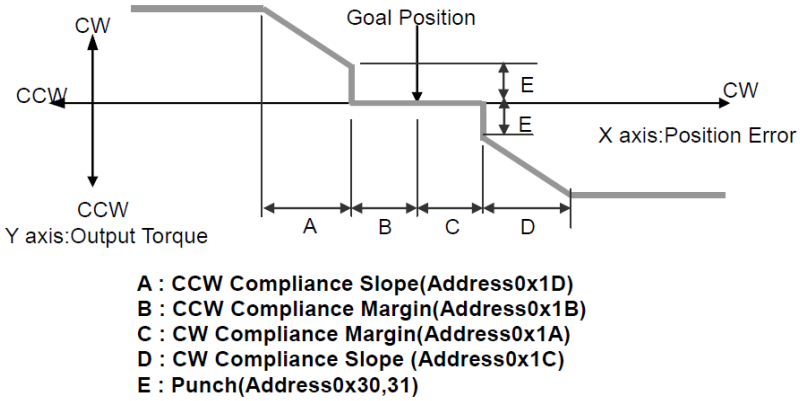


Abbildung 6.3: Zusammenwirken der Parameter Compliance Margin, Compliance Slope und Punch [21]

somit die Positionierungsgenauigkeit des Servomotors direkt zu beeinflussen (Bereiche **B** und **C** in Abbildung 6.3)). Weiter erlauben die verwendeten Servomotoren verschiedene Parameterwerte für Bewegungen *im* (*CW*) und *entgegen* dem Uhrzeigersinn (*CCW*) zu definieren.

Bei der Untersuchung der Auswirkungen des Parameters wurden die Servomotoren wieder unabhängig vom BRIO®-Labyrinth-Spiel untersucht. Dadurch war es möglich einen größeren Drehbereich des Servomotors zu untersuchen. Für beide Servomotoren wurden aus den 1024 möglichen Servopositionen die Positionen 255, 383, 511, 639 und 767 als Testpositionen ausgewählt und jeweils 100 Mal von zufällig gewählten Servopositionen aus angefahren. Dabei wurden sowohl die Soll- als auch die Ist-Positionen des jeweiligen Servomotors aufgezeichnet. Anschließend wurden die erhaltenen Daten ausgewertet und verglichen. Dazu wurde der mittlere absolute Positionierungsfehler aus allen Paaren von Ist- und Soll-Positionen eines Experiments berechnet. Diese Vorgehensweise wurde für die Parameterwerte 0, 2, 4, 8, 16, 32, 64 und 128 des Parameters Compliance Margin durchgeführt. Dabei wurden jeweils beide Parameterwerte für Bewegungen im und entgegen dem Uhrzeigersinn auf den gleichen Wert gesetzt. Während der Experimente wurden beide Werte des Compliance Slope auf 1 und der Punch auf den Standardwert 32 gesetzt.
Die sich ergebenden mittleren absoluten Positionierungsfehler in Abhängigkeit vom Wert des Parameters sind in Abbildung 6.4 für beide Servomotoren grafisch dargestellt. Wie nicht anders zu erwarten, nimmt der Positionierungsfehler der Servomotoren mit zunehmend größeren Werten des Parameters Compliance Margin zu. Während der mittlere absolute Fehler für kleine Werte des Parameters Compliance Margin (0, 2, 4 und 8) in einem Bereich von 1,254 bis 3,470

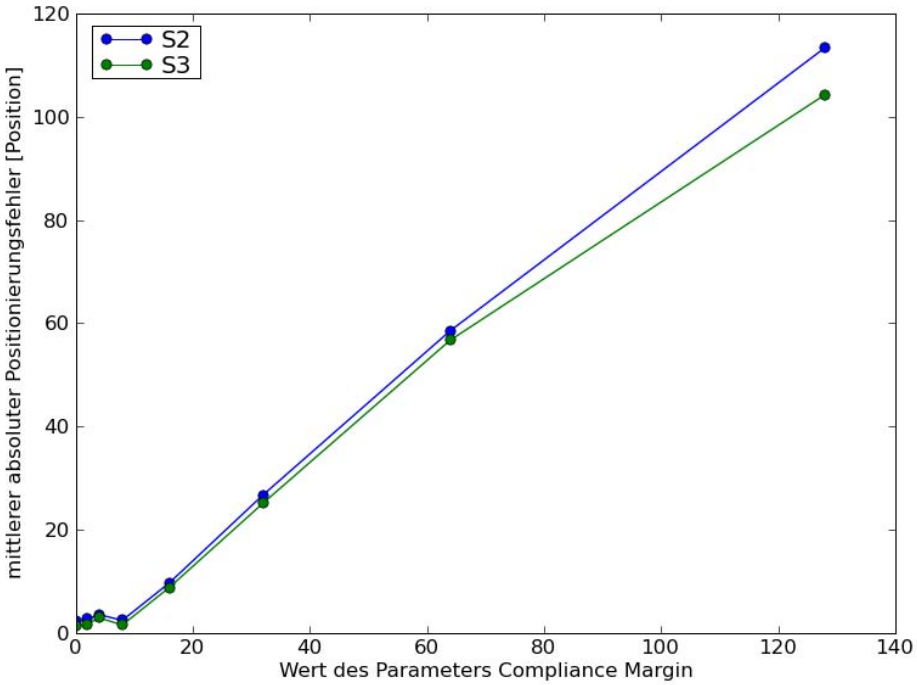


Abbildung 6.4: Vergleich des mittleren absoluten Positionierungsfehlers der beiden Servomotoren in Abhängigkeit vom Wert des Parameters Compliance Margin

Servopositionen liegen, wächst der Fehler für einen Compliance Margin Wert von 128 für $S2$ auf einen Wert von 113,4 Positionen und für $S3$ auf 104,288 Positionen.

Folglich erreichten die beiden Servomotoren den geringsten Fehler für einen Parameterwert von 0, wobei der Positionierungsfehler des Servomotors $S3$ mit 1,254 Positionen deutlich kleiner ausfällt als der von $S2$ mit 2,190 Positionen.
Beide Servomotoren begannen allerdings bei diesem Parameterwert stark zu schwingen. Die Servomotoren versuchten bereits bei einer minimalen Abweichung die Position des Servos durch eine Gegenbewegung zu korrigieren. Diese Gegenbewegung führte zu einer weiteren Abweichung, diesmal jedoch in die andere Richtung. Folglich versuchte der Servomotor fortlaufend mit entsprechenden Gegenbewegungen die entstehenden Abweichungen auszugleichen.

6.2.1.2 Compliance Slope

Als zweites wurde der Parameter Compliance Slope näher untersucht. Der Parameter Compliance Slope bestimmt einen Bereich um die Zielposition des Servomotors, in dem das Ausgangsdrehmoment und damit die Geschwindigkeit des Servomotors beim Annähern an die Zielposition reduziert wird (Bereiche **A** und **D**

in Abbildung 6.3). Innerhalb dieses Bereichs wird das Ausgangsdrehmoment mit kleiner werdendem Abstand zur Zielposition von seinem maximal erlaubten Wert linear auf ein Minimum reduziert. In Abhägigkeit vom Wert dieses Parameters beginnt der Servomotor entsprechend frühzeitig vor Erreichen der Zielposition seine Geschwindigkeit und somit das Risiko eines Überschwingens zu reduzieren. Auch für diesen Parameter können verschiedene Werte für Bewegungen im und entgegen dem Uhrzeigersinn vorgegeben werden.

Um den Einfluss dieses Parameters zu untersuchen, wurde das gleiche Experiment wie im vorherigen Abschnitt (Kapitel 6.2.1.1) durchgeführt. Dabei wurden beide Werte des Parameters Compliance Margin, aufgrund des bei einem Parameterwert von 0 auftretenden Schwingens der Servomotoren, auf einen Wert von 2 und der Parameter Punch auf einen Wert von 32 gesetzt. Anschließend wurde das Experiment für die Werte 1, 2, 4, 8, 16, 32, 64 und 128 des Parameters Compliance Slope durchgeführt. Auch hier wurden beide Parameterwerte gleichgesetzt. Zusätzlich wurde die aktuell verwendete Konfiguration der Servomotoren des BRIO®-Testbeds (CCW Compliance Slope: 60; CW Compliance Slope 70) untersucht. Auch für diese Daten wurden die mittleren absoluten Positionierungsfehler in Abhängigkeit der einzelnen Parameterwerte berechnet und miteinander verglichen (Abbildung 6.5).
Der Parameter Compliance Slope scheint nur geringfügige Auswirkungen auf die Positioniergenauigkeit der Servomotoren zu besitzen. Tendenziell ist ein leichter Anstieg des Positionierungsfehlers in Abhängigkeit vom Wert des Parameters zu erkennen. Jedoch schwanken die mittleren absoluten Positionierungsfehler der Servomotoren sehr stark. Aufgrund des sehr geringen Positionierungsfehler der Standardkonfiguration des Systems wurden im weiteren Verlauf der Arbeit die Standardwerte (CCW: 60; CW: 70; Tabelle 3.1) für diesen Parameter verwandt.

6.2.1.3 Punch

Beim dritten getesteten Parameter der Servomotoren handelt es sich um den Parameter Punch. Dieser bestimmt den minimalen Strom, welcher dem Servomotor während einer Bewegung zur Verfügung steht, und beeinflusst somit das minimale Ausgangsdrehmoment der Servomotoren (Bereich **E** in Abbildung (6.3)). Mit Hilfe dieses Parameters wird das Ausgangsdrehmoment der Servomotoren beim Erreichen der Zielposition definiert. Je höher der Wert des Parameters, desto mehr Drehmoment steht dem Servomotor zum Anfahren und Halten der Zielposition zur Verfügung.

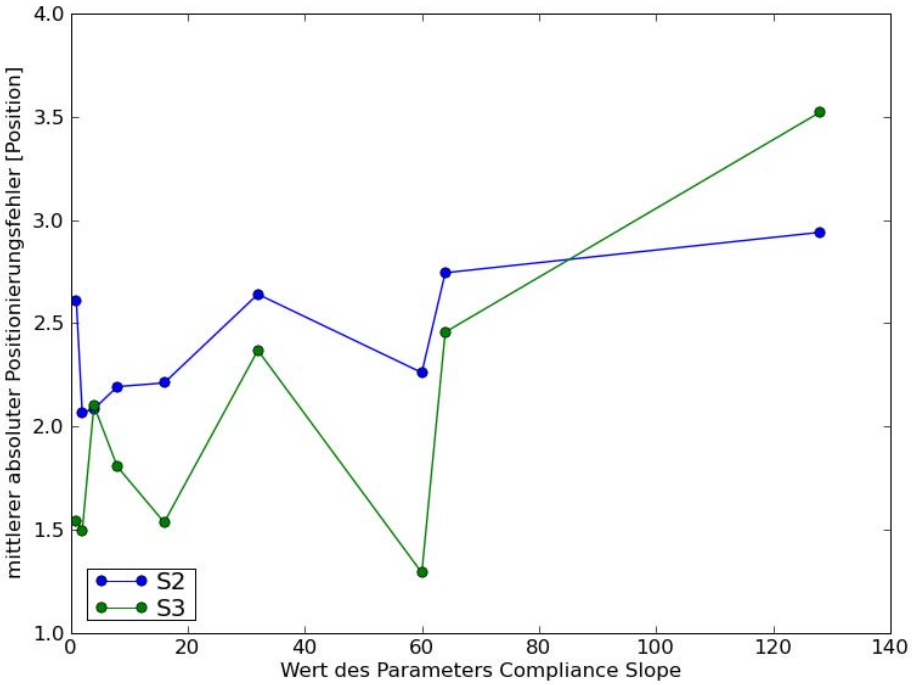


Abbildung 6.5: Vergleich des mittleren absoluten Positionierungsfehlers der beiden Servomotoren in Abhängigkeit vom Wert des Parameters Compliance Slope

Auch der Einfluß dieses Parameters wurde wieder mit der Herangehensweise aus Kapitel 6.2.1.1 untersucht. Hierbei wurden beide Werte des Parameters Compliance Margin auf einen festen Wert von 2 und beide Werte des Parameter Compliance Slope auf einen Wert von 1 gesetzt. Anschließend erfolgte die Bestimmung der mittleren absoluten Positionierungsfehler beider Servomotoren für die Werte 0, 2, 4, 8, 16, 32, 64, 128, 256 und 512 des Parameters Punch. In Abbildung 6.6 sind die resultierenden mittleren absoluten Fehler grafisch dargestellt. Untersuchungen der Parameterwerte 768 und 1023 wurden ebenfalls durchgeführt. Deren Ergebnisse wurden jedoch von der Auswertung ausgeschlossen, da die Servomotoren bei dieser Konfiguration bereits nach kurzer Zeit aufgrund übermäßiger Erwärmung stoppten.

Im Ergebnis scheint der Parameter Punch keinen nennenswerten Einfluss auf die Genauigkeit der Servomotoren zu besitzen. Selbst das drastische Erhöhen des Parameters auf einen Wert von 512 führt zu einem mittleren absoluten Fehler des Servomotors $S2$ von 2,478 Positionen bzw. des Servomotors $S3$ von 1,602 Positionen. Allgemein schwanken die Positionierungsfehler beider Servomotoren in einem relativ kleinen Bereich. Während die Werte des Servomotors $S2$ zwischen den Werten 2,478 und 2,638 variieren, liegen die Werte des Servomotors $S3$ zwischen 1,502 und 1,640 Positionen. Deshalb wurde der Wert des Parameters

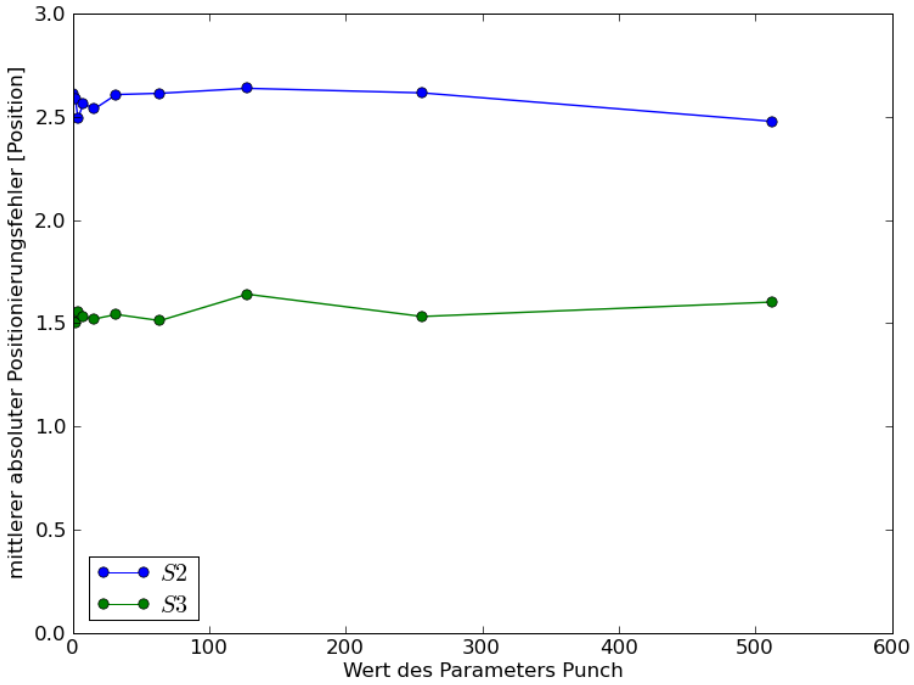


Abbildung 6.6: Vergleich des mittleren absoluten Positionierungsfehlers der beiden Servomotoren in Abhängigkeit vom Wert des Parameters Punch

Punch für alle folgenden Untersuchungen auf den vom Hersteller vorgegebenen Standardwert von 32 gesetzt und damit das Risiko einer übermäßigen Erwärmung selbst bei andauernden Betrieb der Servomotoren minimiert.

6.2.1.4 Diskussion des Einfluss der Servoparameter auf die Positioniergenauigkeit

Für einen Wert des Parameters Compliance Margin von 0 wiesen die Servomotoren die höchste Positionierungsgenauigkeit auf. Allerdings begannen die Servomotoren oftmals zu schwingen. Während der Untersuchung der weiteren Parameter hat sich jedoch gezeigt, dass das Schwingen der Servomotoren in Kombination mit größeren Werten des Parameters Compliance Slope nicht mehr auftrat.
Entsprechend wurden als Werte des Parameters Compliance Slope die beiden Standardwerte (Tabelle 3.1) gewählt, da hiermit zum einen das Schwingen der Servomotoren vermieden wird und zum anderen die Servomotoren verglichen mit den anderen untersuchten Werten des Parameters einen relativ geringen Positionierungsfehler erreichen.
Der Wert des Parameters Punch hat scheinbar keinerlei Einfluss auf die Genauigkeit der Servomtoren und somit wurde hier der vom Hersteller standardmäßig gewählte Wert übernommen.

Insgesamt konnte die Positioniergenauigkeit der Servomotoren mit Hilfe dieses Ansatzes nicht verbessert werden, da die in Kapitel 6.1 bereits verwendete Servokonfiguration nach den durchgeführten Untersuchungen die besten Ergebnisse lieferte. Infolgedessen werden die Standardwerte der einzelnen Parameter in den folgenden Untersuchungen verwendet.

6.2.2 Anpassung der Zielposition unter Berücksichtigung des lokalen Positionierungsfehlers der Servomotoren

Da der mittlere Fehler mit Hilfe einer Anpassung der Parameterwerte der Servomotoren nicht entsprechend kompensiert werden konnte, wurde ein zweiter Ansatz zur Erhöhung der Positioniergenauigkeit der Servomotoren untersucht. Hierbei soll der lokale Positionierungsfehler des jeweiligen Servomotors bereits bei dessen Ansteuerung berücksichtigt werden. Dem Servomotor wird dabei statt der gewünschten Zielposition eine „korrigierte“ Zielposition übergeben.

Um die Funktionsweise dieser Methode zu veranschaulichen, soll folgendes Beispiel dienen. Ausgangspunkt ist dabei die Annahme, dass ein Servomotor die vorgegebene Position 245 anfahren soll. Der Servomotor erreicht jedoch die vorgegebene Position nicht und beendet im Mittel seine Bewegung an Position 243. Die angefahrene Servoposition weicht somit um zwei Positionen von der vorgegebenen Zielposition ab.
Die Idee ist nun, dem Servomotor anstelle der gewünschten Zielposition die um zwei Positionen verschobene Servoposition 247 als Ziel vorzugeben. Aufgrund des lokalen Positionierungsfehlers würde der Servomotor zwei Positionen vor dieser verschobenen Zielposition stoppen und damit die gewünschte Zielposition erreichen.

6.2.2.1 Bestimmung der Abbildung der gewünschten auf die vorzugebende Zielposition

Zur Umsetzung dieser Methode ist die Kenntnis des lokalen Positionierungsfehlers des jeweiligen Servomotors unabdingbar. Als Grundlage der Bestimmung dieses Fehlers dienten die in Kapitel 6.1 aufgezeichneten Soll- und Ist-Positionen. Jeweils für beide Servomotoren wurde mittels der *Methode der kleinsten Fehlerquadrate* [3] eine lineare Abbildung der gewünschten Servoposition auf die vorzugebende Zielposition bestimmt. Mit Hilfe dieser Funktion kann zu einer gegebenen Ist-Position P_{S2} bzw. P_{S3} unter Berücksichtigung des lokalen Positionierungsfehlers

die vorzugebende Soll-Position P^*_{S2} bzw. P^*_{S3} des jeweiligen Servomotors ermittelt werden. Da die Servomotoren nur ganze Zahlen als Zielpositionen akzeptieren, werden die berechneten Soll-Position entsprechend zur nächsten ganzen Zahl gerundet. Die sich ergebende Berechnungsvorschrift der einzelnen Servomotoren ist für $S2$ in (6.1) und für $S3$ in (6.2) dargestellt.

$$P^*_{S2} = round(1,00405316 \cdot P_{S2} + 0,56237193) \tag{6.1}$$

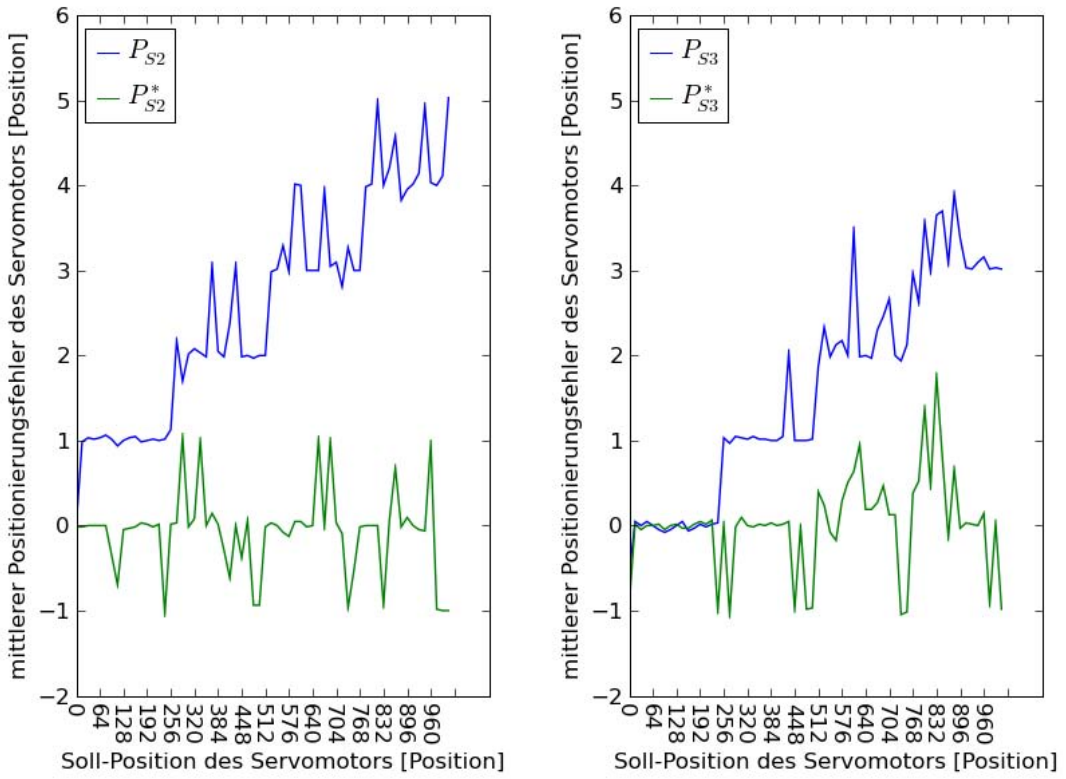
$$P^*_{S3} = round(1,00400549 \cdot P_{S3} - 0,39893924) \tag{6.2}$$

6.2.2.2 Auswirkungen der Abbildung der gewünschten auf die vorzugebende Zielposition

Zur Untersuchung der Auswirkungen dieses Verfahrens wurde dasselbe Experiment wie in Kapitel 6.1 durchgeführt. Dieses Mal wurden allerdings den Servomotoren die mittels (6.1) und (6.2) berechneten Zielpositionen übergeben. Aus den erhaltenen Daten wurde für beide Servomotoren der mittlere Positionierungsfehler der einzelnen Soll-Positionen berechnet und mit den in Kapitel 6.1 berechneten Fehlern verglichen (Abbildung 6.7).
Dabei zeigte sich bei beiden Servomotoren eine deutliche Verbesserung der Positioniergenauigkeit. Während bei den in Kapitel 6.1 aufgezeichneten Daten P_{S2} und P_{S3} der mittlere Positionierungsfehler beider Servomotoren mit ansteigenden Soll-Positionen stufenweise zunimmt, schwanken die Werte des mittleren Positionierungsfehlers bei Berücksichtigung der lokalen Positionierungsungegnauigkeit P^*_{S2} und P^*_{S3} der Servomotoren um den Wert 0. Mit Hilfe des hier entwickelten Verfahrens konnte der mittlere Positionierungsfehler über alle Daten des Servomotors $S2$ von 2,542 Positionen auf -0,076 Positionen verringert werden. Entsprechend reduzierte sich dieser Wert für den Servomotor $S3$ von 1,580 Positionen auf 0,008 Positionen.

6.3 Abbildung der Servopositionen auf die Winkel des Spielbretts

Während die Motoren der Simulation ihre Zielposition in Form von Winkeln übergeben bekommen, akzeptieren die Servomotoren des BRIO®-Testbeds nur Servopositionen im Bereich von null bis 1023. Zur einheitlichen Ansteuerung beider Systeme ist somit eine Umrechnung von Brettwinkeln in Servopositionen bzw. umgekehrt notwendig. Im Rahmen dieser Arbeit wurde eine Ansteuerung der Mo-



(a) $S2$ (b) $S3$

Abbildung 6.7: Vergleich des mittleren Positionierungsfehler des Servomotors $S2$ (a) bzw. $S3$ (b) aufgetragen über die zugehörigen Soll-Positionen

toren mittels Brettwinkeln gewählt, da bereits die Zustände der beiden Systeme Winkel zur Beschreibung der Auslenkung der Spielebenen verwenden. Folglich war es notwendig zu einem vorgegebenen Brettwinkel α bzw. β die jeweilig anzufahrende Position des entsprechenden Servomotors P_{S2} bzw. P_{S3} des BRIO®-Testbeds bestimmen zu können.

Zu diesem Zweck wurde ein ähnliches Experiment wie in Kapitel 6.1 durchgeführt. Die Servomotoren waren hierbei mit den Achsen des BRIO®-Labyrinth-Spiels verbunden, um den Zusammenhang zwischen angefahrener Position der Servomotoren und dem zugehörigen Brettwinkel untersuchen zu können. Auch dieses Mal wurde das BRIO®-Testbed zu Beginn der Experimente kalibriert, um den Bereich der erlaubten Positionen der beiden Servomotoren zu bestimmen (Tabelle 6.1). Aus diesen wurden, beginnend mit dem kleinsten Wert, in Schritten von 8 Positionen verschiedene Testpositionen ausgewählt. Für den Servomotor $S2$ ergaben sich somit 44 Testpositionen und 32 für $S3$. Anschließend wurde jede Testposition von allen anderen Testpositionen des jeweiligen Servomotors aus

angefahren und dabei die Ist-Position des Servomotors sowie der erreichte Brettwinkel ermittelt und aufgezeichnet.

	kleinste erlaubte Servoposition	Nullposition	größte erlaubte Servoposition
$S2$	371	550	729
$S3$	352	488	624

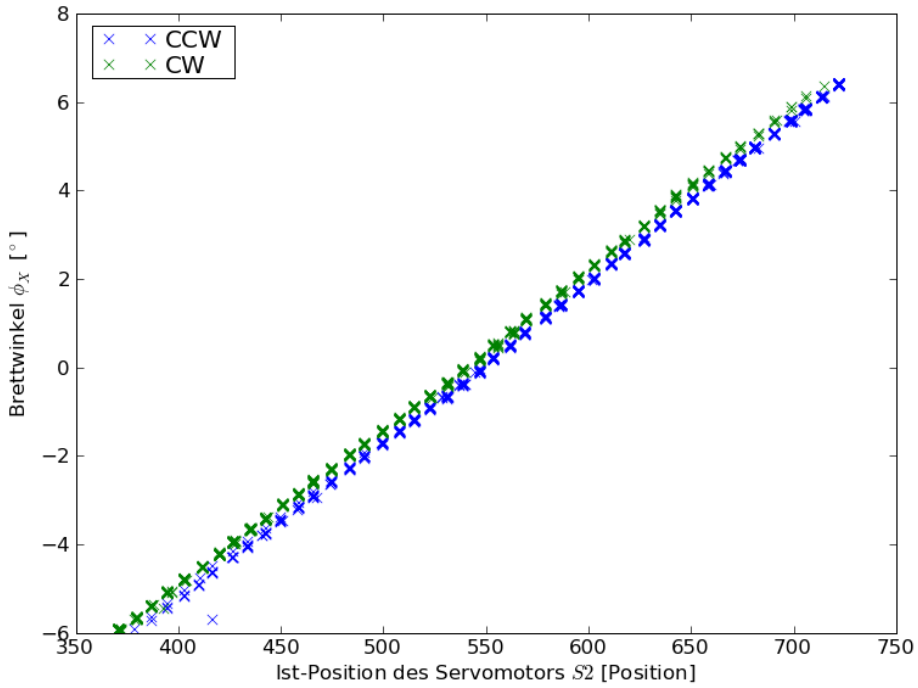
Tabelle 6.1: Kleinste, mittlere und größte erlaubte Position der beiden Servomotoren

6.3.1 Einfluss des Spiels der Gummiriemen

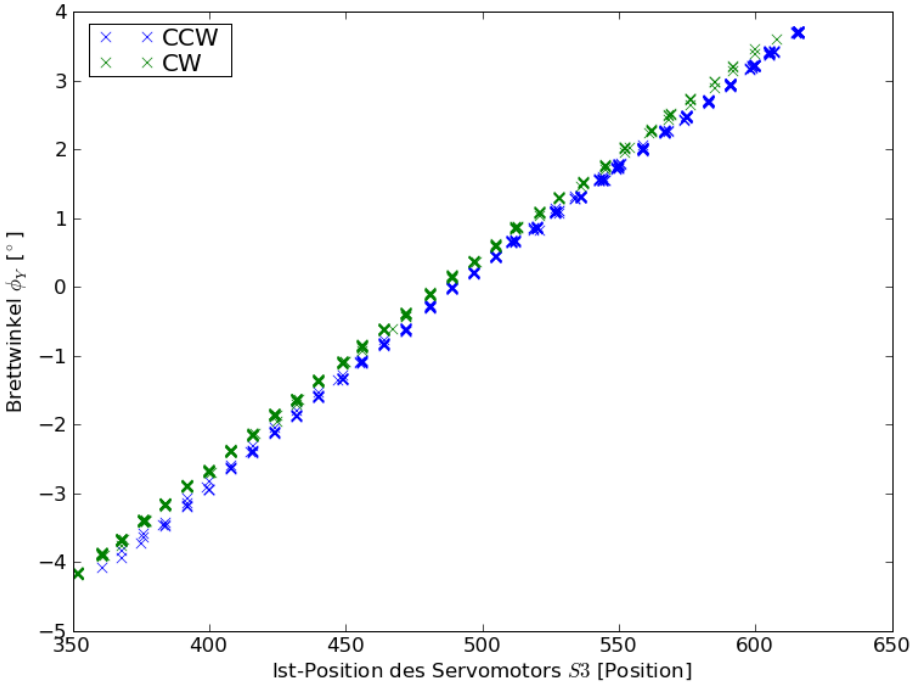
Die Untersuchung der so aufgezeichneten Daten zeigte eine Abhängigkeit der angefahrenen Brettwinkel von der Drehrichtung der Servomotoren. Je nachdem, ob die gewünschte Zielposition durch eine Drehung des Servomotors *im* (blaue Kreuze in Abbildung 6.8) bzw. *entgegen* dem Uhrzeigersinn (grüne Kreuze in Abbildung 6.8) angefahren wurde, unterscheiden sich die angefahrenen Brettwinkel voneinander. Eine Ursache dieser Verschiebung könnte der Einsatz elastischer Gummiriemen zur Übertragung der Drehbewegung der Servomotoren auf die Achsen des BRIO®-Labyrinth-Spiels sein. Denn aufgrund einer zu geringen Spannung der Gummiriemen, verfügen diese über ein leichtes Spiel. Je nach Drehrichtung der Bewegung steht der obere oder untere Strang des Gummiriemens unter Last, während sich der jeweils andere geringfügig entspannt. Folglich werden Drehbewegungen des Servomotors im und entgegen dem Uhrzeigersinn mit einem leichten Versatz zueinander übertragen, da ein Teil der Drehung des Servomotors im Spiel der Gummiriemen verloren geht.

6.3.2 Kompensation des Spiels der Gummiriemen

Um diese Abhängigkeit bei der Ansteuerung der Servomotoren zu berücksichtigen, sollen jeweils zwei Abbildungsfunktionen pro Servomotor, basierend auf den zuvor aufgezeichneten Daten (Kapitel 6.3), ermittelt werden: Je eine für Drehbewegungen im und eine für Drehbewegungen entgegen dem Uhrzeigersinn. Hierdurch soll versucht werden, die Auswirkungen des Spiels der Gummiriemen zu reduzieren und die Positioniergenauigkeit des BRIO®-Testbeds zu erhöhen. Aus diesem Grund wurden alle Paare, bestehend aus angefahrener relativer Ist-Position und erreichtem Brettwinkel, aufgeteilt nach der jeweiligen Drehrichtung des Servomotors separat untersucht. Mit Hilfe der Methode der kleinsten Fehlerquadrate wurden für alle vier Datensätze die Abbildungen des angefahrenen Brettwinkels



(a) $S2$



(b) $S3$

Abbildung 6.8: Gegenüberstellung der gemessenen Brettwinkel ϕ_X bzw. ϕ_Y und der angefahrenen Ist-Position des Servomotors $S2$ (a) bzw. $S3$ (b)

auf die relative Ist-Position des zugehörigen Servomotors ermittelt.

$$P_{S2}^{CCW}(\alpha) = round(27,30680908 \cdot \alpha - 2,77337729) \tag{6.3}$$

$$P_{S2}^{CW}(\alpha) = round(27,96830639 \cdot \alpha - 10,99802313) \tag{6.4}$$

$$P_{S3}^{CCW}(\beta) = round(33,04556877 \cdot \beta \ + 4,7838177) \tag{6.5}$$

$$P_{S3}^{CW}(\beta) = round(32,3511729 \cdot \beta - 1,8994012) \tag{6.6}$$

Die Formeln (6.3), (6.4), (6.5) und (6.6) stellen die erhaltenen Abbildungsfunktionen des gewünschten Zielwinkels α bzw. β auf die entsprechenden Positionen des Servomotors P_{S2} bzw. P_{S3} relativ zur Nullposition für Bewegungen entgegen (CCW) und im Uhrzeigersinn (CW) dar.

6.4 Fehler und Rauschen der angefahrenen Brettwinkel

Nachdem der Positionierungsfehler der Servomotoren und die Abbildung der vorgegegebenen Brettwinkel auf die relativen Servopositionen untersucht sowie entsprechende Verfahren bzw. Abbildungen entwickelt und implementiert wurden, soll nun der Fehler und das Rauschen der angefahrenen Brettwinkel näher betrachtet werden.

Zu diesem Zweck wurden verschiedene Bewegungen des BRIO®-Testbeds zum einen unter Berücksichtigung der entwickelten Methoden und zum anderen ohne die vorgenommenen Anpassungen des BRIO®-Testbeds durchgeführt. Bei dem Verfahren ohne die Berücksichtigung des lokalen Positionierungsfehlers der Servomotoren und dem Spiel des Gummiriemens wurde die relative Position des Servomotors durch Division des Zielwinkels durch den Faktor $0,03°$ berechnet. Diese Umrechnung beruht auf Erfahrungswerten und empirischen Untersuchungen des BRIO®-Testbeds. Für den Winkel des Spielbretts ϕ_X wurden 10 Testwinkel im Bereich zwischen $-5,0°$ und $+5,0°$ ausgewählt. Aufgrund des kleineren Bewegungsspielraumes des Brettwinkels ϕ_Y wurden nur 8 Testwinkel aus dem Bereich von $-4,0°$ und $+4,0°$ ausgewählt.
Anschließend wurden die entsprechenden Bewegungen jeweils 100 Mal für jeden Testwinkel durchgeführt und dabei der erreichte und vorgegebene Brettwinkel

aufgezeichnet. Zur Auswertung und zum Vergleich der beiden Verfahren wurde der Mittelwert sowie die Standardabweichung der Differenz zwischen angefahrenem und vorgegebenem Brettwinkel der einzelnen Messreihen jeweils für das alte und das neue, optimierte Verfahren berechnet. Die erhaltenen Mittelwerte und Standardabweichungen für den Brettwinkel ϕ_X sind in der Tabelle 6.2 und für den Brettwinkel ϕ_Y in der Tabelle 6.3 dargestellt.

vorgegebener Winkel ϕ_X [°]	Abweichung zwischen vorgegebenen Winkel und gemessenem Winkel (altes Verfahren) $\triangle\phi_X$ [°]	Abweichung zwischen vorgegebenen Winkel und gemessenem Winkel (neues Verfahren) $\triangle\phi_X$ [°]
$-5,0$	$0,4940 \pm 0,0033$	$-0,1690 \pm 0,0039$
$-4,5$	$0,3754 \pm 0,0050$	$-0,1587 \pm 0,0050$
$-4,0$	$0,2808 \pm 0,0061$	$-0,1333 \pm 0,0041$
$-3,5$	$0,2291 \pm 0,0042$	$-0,0959 \pm 0,0057$
$-3,0$	$0,1300 \pm 0,0047$	$-0,0525 \pm 0,0024$
$-2,5$	$0,0475 \pm 0,0039$	$-0,1054 \pm 0,0031$
$-2,0$	$0,0055 \pm 0,0039$	$-0,0699 \pm 0,0037$
$-1,5$	$-0,0284 \pm 0,0040$	$-0,0482 \pm 0,0049$
$-1,0$	$-0,1387 \pm 0,0042$	$-0,0294 \pm 0,0049$
$-0,5$	$-0,2330 \pm 0,0049$	$-0,0011 \pm 0,0046$
$0,5$	$-0,1876 \pm 0,0041$	$-0,0090 \pm 0,0045$
$1,0$	$-0,2532 \pm 0,0043$	$-0,0193 \pm 0,0053$
$1,5$	$-0,4221 \pm 0,0093$	$-0,0378 \pm 0,0042$
$2,0$	$-0,5251 \pm 0,0034$	$-0,0487 \pm 0,0037$
$2,5$	$-0,6549 \pm 0,0031$	$-0,0089 \pm 0,0034$
$3,0$	$-0,6886 \pm 0,0038$	$-0,1921 \pm 0,0039$
$3,5$	$-0,8287 \pm 0,0050$	$-0,0614 \pm 0,0036$
$4,0$	$-0,8676 \pm 0,0065$	$-0,0109 \pm 0,0030$
$4,5$	$-0,9727 \pm 0,0065$	$-0,0134 \pm 0,0084$
$5,0$	$-1,1505 \pm 0,0056$	$-0,0676 \pm 0,0059$

Tabelle 6.2: Vergleich des Mittelwerts und der Standardabweichung der Differenz zwischen dem vorgegebenen Ziel- und dem angefahrenen Brettwinkel ϕ_X

In beiden Tabellen zeigt sich, dass die Positioniergenauigkeit des BRIO®-Testbeds durch die Berücksichtigung des lokalen Positionierungsfehlers der Servomotoren sowie des Spiels der Gummiriemen erhöht werden konnte. Abgesehen von wenigen Ausnahmen, wurde mit Hilfe des entwickelten Verfahrens die mittlere Abweichung zwischen den angefahrenen und den vorgegebenen Brettwinkel für beide Brettwinkel im Vergleich zur alten Methode verkleinert. Am deutlichsten zeigt sich die Verbesserung der Positioniergenauigkeit in den Messreihen für Brettwinkel

vorgegebener Winkel ϕ_Y [°]	Abweichung zwischen vorgegebenen Winkel und gemessenem Winkel (altes Verfahren) $\triangle\phi_Y$ [°]	Abweichung zwischen vorgegebenen Winkel und gemessenem Winkel (neues Verfahren) $\triangle\phi_Y$ [°]
$-4,0$	$0,0910 \pm 0,0064$	$0,0316 \pm 0,0036$
$-3,5$	$0,1210 \pm 0,0035$	$0,0637 \pm 0,0054$
$-3,0$	$0,0666 \pm 0,0046$	$0,0622 \pm 0,0037$
$-2,5$	$0,0556 \pm 0,0034$	$0,0209 \pm 0,0029$
$-2,0$	$0,0495 \pm 0,0026$	$0,0525 \pm 0,0040$
$-1,5$	$0,0425 \pm 0,0020$	$0,0415 \pm 0,0036$
$-1,0$	$0,0482 \pm 0,0041$	$0,0799 \pm 0,0030$
$-0,5$	$0,1010 \pm 0,0029$	$0,1290 \pm 0,0044$
$0,5$	$0,2034 \pm 0,0064$	$0,0302 \pm 0,0057$
$1,0$	$0,2616 \pm 0,0055$	$0,0361 \pm 0,0152$
$1,5$	$0,2878 \pm 0,0061$	$0,0498 \pm 0,0050$
$2,0$	$0,2992 \pm 0,0199$	$0,0657 \pm 0,0080$
$2,5$	$0,2964 \pm 0,0104$	$0,0370 \pm 0,0197$
$3,0$	$0,2733 \pm 0,0170$	$0,0390 \pm 0,0090$
$3,5$	$0,3010 \pm 0,0036$	$0,0750 \pm 0,0123$
$4,0$	$0,2726 \pm 0,0034$	$-0,0055 \pm 0,0128$

Tabelle 6.3: Vergleich des Mittelwerts und der Standardabweichung der Differenz zwischen dem vorgegebenen Ziel- und dem angefahrenen Brettwinkel ϕ_Y

größer gleich null. Während hier der mittlere Positionierungsfehler des alten Verfahrens sich zunehmend vergrößert, bleiben die Mittelwerte des neuen Verfahrens vergleichsweise stabil.

Zum Abschluß sollte noch das Rauschen der angefahrenen Winkel des Spielbretts des BRIO®-Testbeds nachgebildet werden, um später beim Training eines Agenten den Positionierungsfehler des realen Systems innerhalb der BRIO®-Simulation berücksichtigen zu können. Hierbei soll dem Rauschen der angefahrenen Brettwinkel ein normalverteiltes Rauschen (*Normalverteilung* siehe [3]) um den Mittelwert 0 zugrunde gelegt werden. Zur Bestimmung der Standardabweichung wurden alle gemessenen Werte jeweils eines Brettwinkels ϕ_X oder ϕ_Y zusammengefasst und die Standardabweichung der erhaltenen Verteilung der gemessenen Winkel berechnet. Es ergab sich eine Standardabweichung von $0,0299°$ für den Brettwinkel ϕ_X sowie von $0,0083°$ für ϕ_Y.

Kapitel 7

Untersuchung der Bestimmung der Kugelposition

Nachdem das Rauschen der gemessenen Brettwinkel und die Positioniergenauigkeit der Servomotoren betrachtet wurden, soll im folgenden Abschnitt die Bestimmung der Kugelposition hinsichtlich ihres Rauschens und möglicher Verbesserungen des Verfahrens untersucht werden.

Zu diesem Zweck wurde zunächst der Algorithmus zur Detektion der Kugel (Kapitel 3.2.3.2) in den von der Kamera aufgezeichneten Bildern hinsichtlich seines Rauschens und seiner Abhängigkeit von der Bildhelligkeit untersucht und versucht die Genauigkeit durch Kompensation der Kameraverzeichnung zu erhöhen. Anschließend müssen die detektierten Pixelkoordinaten in Spielbrettkoordinaten umgerechnet werden. Hierzu wurde das bestehende Verfahren (Kapitel 3.2.3.3) aufgegriffen und erweitert.
Zu guter letzt wurden die Auswirkungen der neuen Verfahren hinsichtlich des Fehlers zwischen der tatsächlichen und der detektierten Kugelposition untersucht und ein entsprechendes Rauschen der Kugelposition des BRIO®-Testbeds modelliert.

7.1 Rauschen der Pixelkoordinaten der detektierten Kugelposition

Zunächst wurde das Rauschen der Pixelkoordinaten der durch den Algorithmus (Kapitel 3.2.3.2) bestimmten Position der Kugel bei ruhender Kugel untersucht. Dazu wurde die weiße Kugel mittig auf dem Spielbrett positioniert (Abbildung 7.1) und 1000 Mal die Pixelposition der Kugel bestimmt.

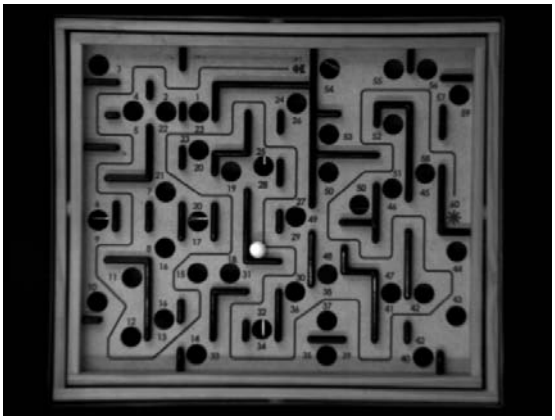


Abbildung 7.1: Testaufbau zur Bestimmung des Rauschen der Kugelposition

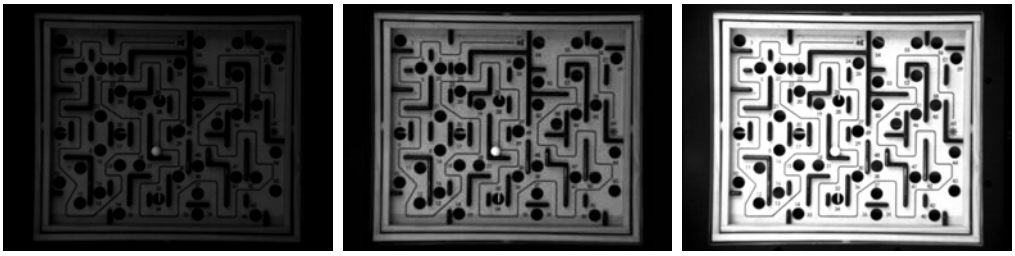
Es zeigte sich, dass die Pixelkoordinaten der erkannten Kugel kaum bzw. nur sehr geringfügig schwanken. Bei allen 1000 Messungen traten immer nur drei verschiedene Paare von (x,y)-Koordinaten auf. Dabei wurde in 97,8% aller Messdaten die Kugelposition mit den Koordinaten (303,292) angegeben. In 1,9% aller Fälle lag die Kugelposition bei (302,291) und in den restlichen 0,3% bei (302,292).
Die Ursache des nur geringen Rauschens ist auf ein leichtes Schwanken der Ausleuchtung des BRIO®-Testbeds zurückzuführen. Dies konnte während der Experimente nicht vollkommen ausgeschlossen werden. Aufgrund der minimalen Auswirkungen wurden im weiteren Verlauf der Arbeit die Pixelkoordinaten als rauschfrei betrachtet.

7.2 Abhängigkeit der Bestimmung der Pixelkoordinaten von der Bildhelligkeit

Im Verlauf der Untersuchung des Algorithmus zur Bestimmung der Kugelposition zeigte dieser eine starke Abhängigkeit von den herrschenden Helligkeitsverhältnissen im Labor.

Je geringer die Helligkeit des Bildes (Abbildung 7.2 (a)), desto häufiger wurde die Kugel im Bild nicht erkannt. Dies ist vor allem auf den im Algorithmus verwandten konstanten Schwellwert zurückzuführen. Aufgrund der reduzierten Helligkeit des Bildes ist die Intensität des stärksten Pixels geringer als der konstante Schwellwert und somit wird vom Algorithmus angenommen, dass sich die Kugel nicht im Bild befindet. Demgegenüber wird bei großer Helligkeit die korrekte Kugelposition oft nicht erkannt (Abbildung 7.2 (c)). Grund hierfür sind der abnehmende Intensitätsunterschied zwischen der weißen Kugel und dem hellen

Spielbrett. Je kleiner dieser Unterschied ist, desto größer wird die Wahrscheinlichkeit einer fehlerhaften Positionsbestimmung.



(a) geringe Helligkeit (b) mittlere Helligkeit (c) hohe Helligkeit

Abbildung 7.2: Unterschiedliche Helligkeitsverhältnisse

Als Grundlage der Untersuchung dieser Abhängigkeit von den Helligkeitsverhältnissen diente das Experiment aus Kapitel 7.1. Auch hier wurde die Kugel an eine feste Position in der Mitte des Spielbretts gelegt und anschließend 1000 Mal die Kugelkoordinaten bestimmt. Zwischen den einzelnen Experimenten wurden die Helligkeitsverhältnisse mit Hilfe der Belichtungszeit der Kamera verändert. Bei $10ms$ beginnend, wurde die Belichtungszeit der Kamera schrittweise um $5ms$ auf eine maximale Belichtungsdauer von $50ms$ erhöht. Für die einzelnen Experimente wurde jeweils die mittlere Helligkeit über alle 1000 Bilder einer Messreihe bestimmt und aufgezeichnet.
Anschließend wurden die Standardabweichungen der x- bzw. y-Werte der Koordinaten der einzelnen Messreihen gegenüber der mittleren Helligkeit der Messreihe aufgetragen (Abbildung 7.3) und verglichen. Es wird davon ausgegangen, dass, wenn die Position der Kugel im Bild korrekt und stabil erkannt wird, die Koordinaten der Kugel nur sehr leicht um maximal einen Pixel in jede Richtung schwanken. Je kleiner die Standardabweichung der Kugelkoordinaten ist, desto besser wird die ruhende Kugel erkannt. Variieren die Werte der Pixelkoordinaten der ruhenden Kugel stärker, kann angenommen werden, dass es sich bei dem detektierten Punkt nicht um die weiße Kugel handelt, da sich diese während des Versuchs nicht bewegt. Eine hohe Standardabweichung deutet somit auf eine fehlerhafte Erkennung der Kugelposition hin.
Die Auswertung der so erhaltenen Daten zeigt, dass der Algorithmus für Bilder mit einem mittleren Helligkeitswert zwischen 35 und 122 gut funktioniert. Bei einer mittleren Helligkeit der Bilder von weniger als 19,307 konnte der Algorithmus die Kugel im Bild über die gesamte Messreihe hinweg nicht erfassen. Dem gegenüber stieg die Standardabweichung für Bilder ab einer mittleren Helligkeit von 137,747 auf einen Wert von über 20 an. Bei den durchgeführten Messreihen, deren mittlere Helligkeit sich in dem Bereich zwischen 35 und 122 befand, wiesen

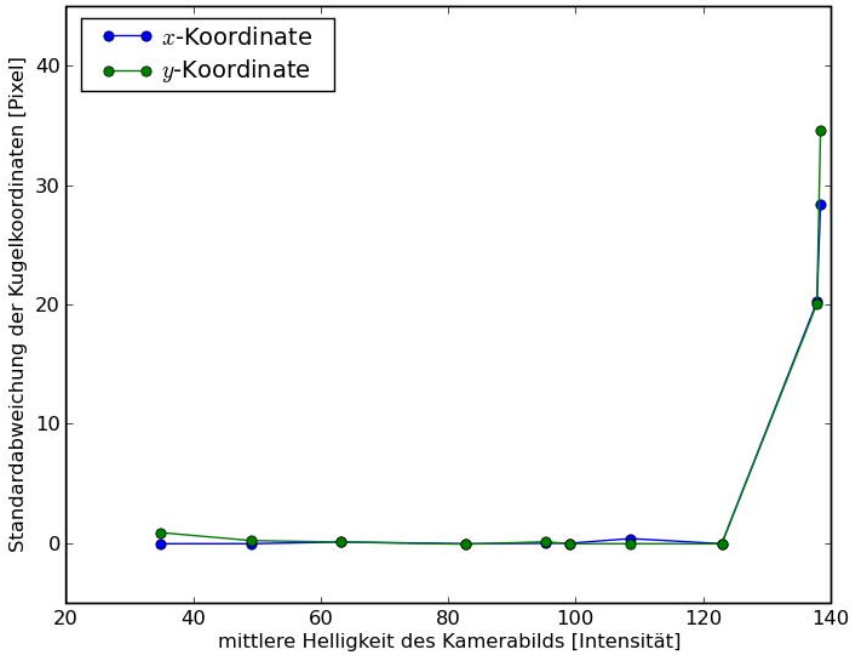


Abbildung 7.3: Standardabweichung der x- und y-Werte der Pixelkoordinaten der detektierten Kugel in Abhängigkeit von der mittleren Helligkeit des Kamerabildes

die Standardabweichungen beider Kugelkoordinaten Werte zwischen 0,0 bis maximal 0,93 auf. Die Messreihe mit der größten Standardabweichung von 0,93 war zugleich die Messreihe mit der kleinsten mittleren Helligkeit von 34,758 und liegt somit an der unteren Grenze des erfassbaren Helligkeitsbereichs.

Zusammenfassend lässt sich feststellen, dass der Algorithmus für Graustufenbilder mit einer mittleren Helligkeit im Bereich von 35 bis 122 gut funktioniert und bei der Bestimmung der Kugelposition, sofern sich kein zweites helles Objekt im Bild befindet, ein sehr geringes Rauschen aufweist.

7.3 Kompensation der Verzeichnung der Kamera

Neben der Abhängigkeit der Kugeldetektion von der Helligkeit des aufgezeichneten Kamerabildes wurde eine Verzerrung der Geometrie des Spielbretts des BRIO®-Testbeds in den Kamerabildern festgestellt.

Dieser Abbildungsfehler ist auf die Verzeichnung der eingesetzten Kamera (Kapitel 3.2.2.7) und deren Optik zurückzuführen. Diese Verzeichnung kann die Detektion der Kugelposition negativ beeinflussen. Zum einen könnte die detektierte

Kugelposition verfälscht werden, da die Position der Kugel im Bild aufgrund der Verzerrung leicht verschoben ist und zum anderen kann es zu Fehlern bei der Umrechnung der Pixelkoordinaten der Kugel in die Spielbrettkoordinaten kommen, da die Landmarkendetektion (Kapitel 3.2.3.3) ebenfalls mit verzerrten Bildern durchgeführt wird.

Verzeichnung (engl. *distortion*) ist ein geometrischer Abbildungsfehler eines optischen Systems. Aufgrund von Ungenauigkeiten im Design und dem Zusammenbau der Linsen des Kameraobjektivs werden verschiedene Bildpunkte verzerrt, so dass die abgebildete Geometrie eines Objektes von dessen tatsächlicher Geometrie abweichen kann.

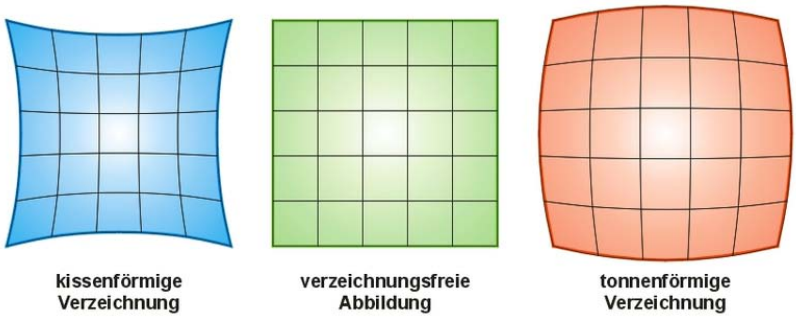


Abbildung 7.4: Verzeichnungsarten [17]

Als Beispiel sind in Abbildung 7.4 zwei mögliche Verzerrungen der Objektgeometrie dargestellt. Als Prüfobjekt dient hierbei ein quadratisches Muster. Das Bild in der Mitte ist verzeichnungsfrei, d.h. das quadratische Muster wird korrekt wiedergegeben. Im linken Bild spricht man von einer kissenförmigen Verzeichnung. Hierbei werden die Ecken des Musters ausgehend vom Bildmittelpunkt weiter nach außen gestreckt. Im rechten Bild hingegen werden die äußeren Ecken zur Bildmitte hingezogen: Man spricht hierbei von einer tonnenförmigen Verzeichnung.

Die entstehenden Verzerrungen sind von den intrinsischen Parametern des verwendeten optischen Systems abhängig. Die intrinsischen oder auch inneren Parameter beschreiben den internen geometrischen Aufbau und die optischen Eigenschaften der Kamera. Zu den inneren Parametern gehören die Brennweite f_C, die Pixelkoordinaten der Bildmitte c_C sowie die Verzeichnungskoeffizienten k_C. Mit Hilfe dieser Parameter können die Auswirkungen der Verzeichnung kompensiert werden.

Um die intrinsischen Parameter optischer Systeme zu bestimmen, existieren ver-

schiedene Programme. Im Rahmen dieser Master-Arbeit wurde die 2008 von Jean-Yves Bouguet veröffentlichte *Camera Calibration Toolbox* für *MATLAB* [2] eingesetzt.

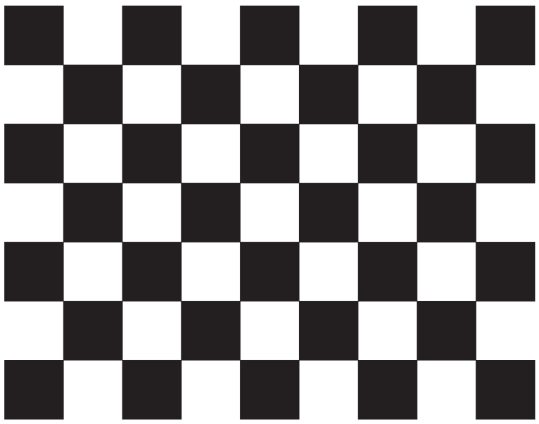


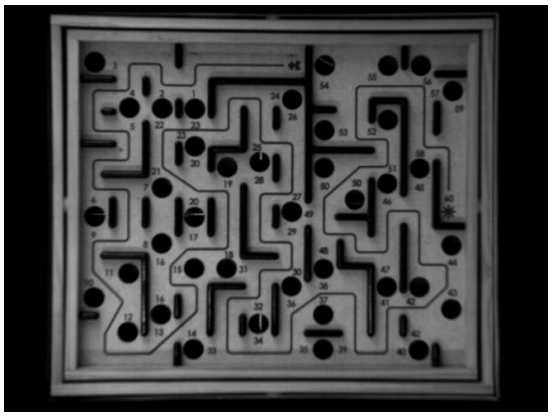
Abbildung 7.5: Kalibrierungs-Muster [2]

Hierbei wird ein Kalibrierungs-Muster mit vorgegebenen Abmaßen verwendet (Abbildung 7.5), um die intrinsischen Parameter der Kamera zu bestimmen. Von diesem Muster wurden mittels der verwendeten digitalen Kamera (Kapitel 3.2.2.7) und deren Objektiv 30 Bilder in verschiedenen Positionen aufgenommen. Aus den aufgenommenen Bildern und den auftretenden Verzerrungen des Testmusters wurden anschließend die intrinsischen Parameter der Kameraoptik mit Hilfe der MATLAB-Toolbox ermittelt. Die Werte der einzelnen Parameter sind in (7.1) aufgeführt.

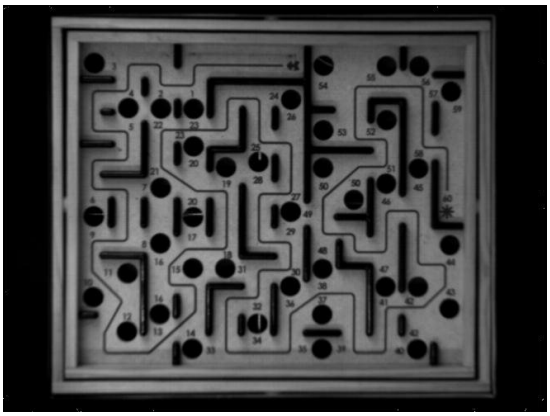
$$f_C = \begin{pmatrix} 2099,32 \\ 2100,08 \end{pmatrix}; \quad c_C = \begin{pmatrix} 352,49 \\ 228,37 \end{pmatrix}; \quad k_C = \begin{pmatrix} 0,42 \\ 7,48 \\ 0,00 \\ 0,00 \\ 0,00 \end{pmatrix} \tag{7.1}$$

Unter Berücksichtigung der erhaltenen intrinsischen Parameter können die auftretenden Verzerrungen des Kamerabildes reduziert werden. Dazu wurde die Bildakquise des BRIO®-Testbeds um die OpenCV-Funktion *cvUndistort2()* erweitert. Diese Funktion transformiert das aufgenommene Bild entsprechend der intrinsischen Parameter des Kameraobjektivs. Für jedes Pixel des transformierten Bildes berechnet die Funktion die Koordinaten der zugehörigen Position im aufgenommenen Bild und interpoliert aus den Intensitätswerten der angrenzenden Pixel des aufgenommenen Kamerabilds den Intensitätswert des transformierten Pixels. Die Auswirkungen der Kompensation der Kameraverzeichnung ist in Abbildung 7.6 gezeigt. Während im Original (Abbildung 7.6 (a) und (c)) eine kissenförmige

Verzeichnung des eigentlich rechtwinkligen Spiels deutlich zu erkennen ist, bildet das entzerrte Bild (Abbildung 7.6 (b) und (d)) die rechteckige Geometrie des Spielbretts besser ab.



(a) Original



(b) entzerrtes Bild

(c) Original (Ausschnitt)

(d) entzerrtes Bild (Ausschnitt)

Abbildung 7.6: Vergleich des aufgenommenen Bildes (a) und (c) mit dem Bild nach der Kompensation der Kameraverzeichnung (b) und (d)

Abschließend wurden die Auswirkungen der Kompensation der Kameraverzeichnung auf das Rauschen der Pixelkoordinaten untersucht. Dazu wurde das in Kapitel 7.1 durchgeführte Experiment mit der erweiterten Bildakquise erneut durchgeführt. Bei der Auswertung der Daten zeigte sich eine ähnliche Verteilung des Rauschens der Pixelkoordinaten der Kugel wie in Abschnitt 7.1. Hierbei schwankte die Position der Kugel wieder zwischen drei benachbarten (x,y)-Koordinaten. Dabei wurde die Kugel zu $98,9\%$ bei den Pixelkoordinaten (318,289) bestimmt. Dem gegenüber wurde die Kugelposition in nur $0,7\%$ der Messungen an der Position (318,290) bzw. mit $0,4\%$ bei (317,289) erkannt.

Zusammenfassend kann festgestellt werden, dass die Kompensation der Kameraverzeichnung keine erkennbaren Auswirkungen auf das Rauschen der Pixelkoordinaten der weißen Kugel im Kamerabild hat. Somit wird weiterhin angenommen,

dass die Pixelkoordinaten der detektierten Kugel unverrauscht sind.

7.4 Koordinatentransformation der Brettkoordinaten

Bisher wurden die Winkel des Spielbretts bei der Bestimmung der Brettkoordinaten der Kugelposition der BRIO®-Simulation sowie des BRIO®-Testbeds nicht berücksichtigt. In Abbildung 7.7 sind die Auswirkungen unterschiedlicher Winkel der Spielebene auf die Position der Kugel visualisiert. Obwohl alle drei dargestellten Kugeln sich im Kamerabild des BRIO®-Testbeds an der gleichen Stelle befinden, ist deren Abstand zum Mittelpunkt des Spielbretts $\triangle x$, $\triangle x'$ und $\triangle x''$ unterschiedlich. Ähnlich verhält es sich bei der BRIO®-Simulation. Auch hier werden lediglich die x- und y-Koordinaten der Position des Kugelmittelpunktes im kartesischen Koordinatensystem als Zustand des Systems zurückgegeben. Somit ist es möglich, dass sich die simulierte Kugel obwohl gleicher x- und y-Koordinaten an verschiedenen Positionen bezüglich der ausgelenkten Spielebene befinden. Folglich ist die Berücksichtigung der Brettwinkel bei der Bestimmung der Kugelposition notwendig.

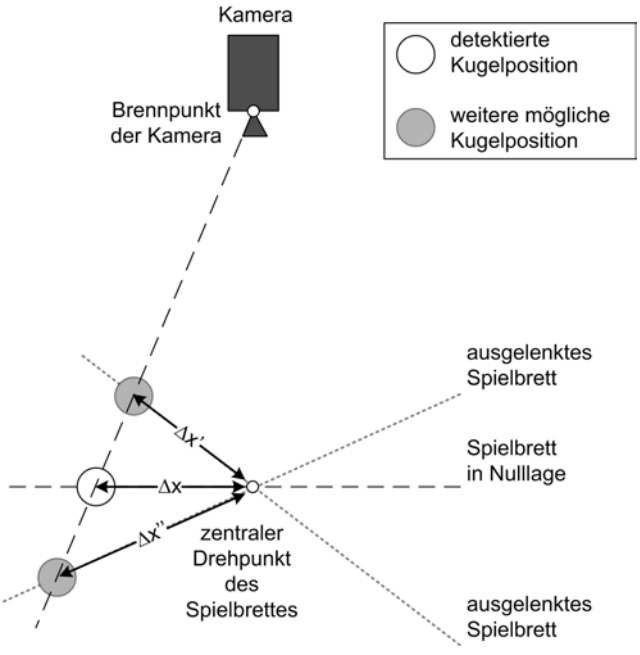


Abbildung 7.7: Einfluss des Brettwinkels auf die Kugelposition

Aus diesem Grund wurde die BRIO®-Simulation sowie das BRIO®-Testbed um entsprechende Koordinatentransformationen erweitert, so dass beide Systeme ein gemeinsames Koordinatensystem zur Beschreibung der Kugelposition verwenden

(Abbildung 7.8), dessen Ursprung im Schnittpunkt der beiden realen Drehachsen liegt und dessen x-y-Ebene stets parallel zur ausgelenkten Spielebene verlaufen soll. Um dies zu erreichen, waren Anpassungen beider Systeme notwendig, die in den folgenden Abschnitten vorgestellt werden.

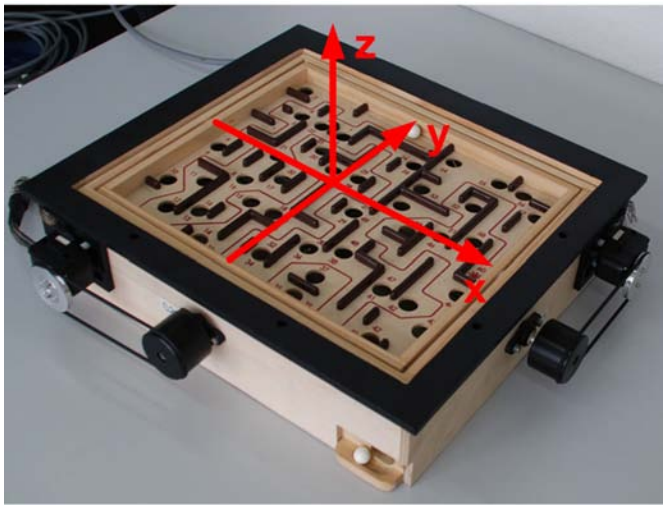


Abbildung 7.8: Grafische Darstellung des gemeinsamen Koordinatensystems anhand des BRIO®-Testbeds

7.4.1 Umrechnung der Kugelkoordinaten der Simulation

Als erstes wurde die Simulation des BRIO®-Labyrinth-Spiels auf die Verwendung eines Koordinatensystems relativ zur Ebene des Spielbretts umgestellt. Zu diesem Zwecke waren zunächst einige Anpassungen an der BRIO®-Umgebung der Simulation notwendig. Der Schnittpunkt der beiden Drehachsen der simulierten Spielebene wurde entsprechend angepasst, so dass sich dieser, wie beim BRIO®-Testbed, in einem Abstand von $7,5mm$ über der Mitte der Spielebene befindet. Zusätzlich wurden die Koordinaten der einzelnen Elemente der Simulation (3D-Modell des Spielbretts, Motoren, usw.) entsprechend angepasst, so dass der Ursprung des kartesischen Koordinatensystems mit dem Schnittpunkt der beiden Drehachsen der Simulation übereinstimmt.

Ein weiterer Vorteil der Simulationsumgebung gegenüber dem BRIO®-Testbed ist die Kenntnis der genauen Position der Mitte der simulierten Kugel sowie die genaue Auslenkung des simulierten Spielbretts gegenüber der Nulllage. Somit können die Koordinaten des Kugelmittelpunktes relativ zur Ebene des aktuell ausgelenkten Spielbretts durch Rotation der kartesischen (x,y,z)-Koordinaten des Kugelmittelpunktes berechnet werden. Dabei wird zunächst eine Rotation der Koordinaten um die x-Achse um den Winkel α und anschließend eine Rotation

um die ebenfalls rotierte y-Achse um den Winkel β durchgeführt. Die Winkel α und β sind dabei durch die Winkel der Spielebene bezüglich der zugehörigen Drehachsen bestimmt. Mathematisch entspricht die Rotation eines Vektors (x,y,z) im mathematisch positiven Drehsinn (entgegen dem Uhrzeigersinn) um den Winkel α um die x-Achse der Multiplikation des Vektors mit der Rotationsmatrix R_X (Formel 7.2) [3][13]. Entsprechend kann ein Vektor um die y-Achse um den Winkel β durch Multiplikation mit der Rotationsmatrix R_Y (Formel 7.3) rotiert werden.

$$R_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{7.2}$$

$$R_Y = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & & \cos\beta \end{pmatrix} \tag{7.3}$$

Bei der Bestimmung der Drehwinkel α und β ist zu beachten, dass die Rotation der Auslenkung des Spielbretts entgegenwirken soll. Somit erhält man die jeweiligen Drehwinkel durch Umkehren des Vorzeichens der Brettwinkel ϕ_X und ϕ_Y. Folglich können die Koordinaten der Kugel relativ zur Ebene des Spielbretts (x^*, y^*, z^*) aus den kartesischen Koordinaten der Simulation (x, y, z) mittels der Formel (7.4) errechnet werden.

$$\begin{pmatrix} x^* \\ y^* \\ z^* \end{pmatrix} = R_Y \cdot R_X \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{7.4}$$

7.4.2 Umrechnung der Kugelkoordinaten des BRIO®-Testbeds

Als zweites wurde die Bestimmung der Kugelposition des BRIO®-Testbeds auf die Verwendung eines Koordinatensystems relativ zur Ebene des Spielbretts umgestellt. Entgegen der BRIO®-Simulation stehen dem BRIO®-Testbed nicht alle notwendigen Informationen zur Umrechnung der Koordinaten direkt zur Verfügung und müssen daher aus den verfügbaren Sensorwerte ermittelt werden. In den nun folgenden Abschnitten werden die einzelnen Schritte der Umrechnung vorgestellt.

7.4.2.1 Umrechnung der zweidimensionalen Koordinaten der detektierten Kugel

Wie bereits in Kapitel 3.2.3.3 erwähnt, wird die Position der Kugel bisher in zweidimensionalen Koordinaten, relativ zur oberen linken Ecke des Spielbretts als Koordinatenursprung, bestimmt. Zur Umrechnung der Koordinaten werden aber die dreidimensionalen Koordinaten der Kugel relativ zum Schnittpunkt der beiden Drehachsen des BRIO®-Testbeds benötigt. Aufbaubedingt liegt der Schnittpunkt der beiden Drehachsen genau in der Mitte des Spielbretts in einem Abstand von 7,5 mm über der Spielfläche.
Zu diesem Zweck werden aus den abgebildeten Kugelkoordinaten x und y zunächst die Koordinaten der Kugel x' und y' relativ zur Mitte des Spielbretts berechnet (Formeln (7.5) und (7.6)). Als Bezugspunkt der zuvor bestimmten Brettkoordinaten der Kugel dienen die Koordinaten der oberen linken Ecke des Spielbretts im Bild. Die Verschiebung des Koordinatenursprungs erfolgt somit durch einfache Addition bzw. Subtraktion der Hälfte der jeweiligen Abmaße des Spielbretts.

$$x' = x - 0,1555m \tag{7.5}$$

$$y' = y + 0,1325m \tag{7.6}$$

7.4.2.2 Bestimmung der dreidimensionalen Kugelposition unter Berücksichtigung der Brettwinkel und der Position der Kamera

Anschließend muss aus den zuvor erhaltenen, zweidimensionalen Kugelkoordinaten die dreidimensionale Position der Kugel relativ zum Schnittpunkt der Drehachsen des BRIO®-Spiels ermittelt werden. Zusätzlich sollen hierbei, wie in Abbildung 7.7 gezeigt, die Brettwinkel ϕ_X und ϕ_Y sowie der Blickwinkel und damit die Position der Kamera (x_c, y_c, z_c) berücksichtigt werden.

Während die Winkel des Spielbretts des BRIO®-Testbeds direkt zur Verfügung stehen, muss die Position der Kamera in Bezug auf den Schnittpunkt der beiden Drehachsen des BRIO®-Spiels zunächst bestimmt werden. Aufgrund des Aufbaus des BRIO®-Testbeds wurde der Wert der z-Koordinate der Kamera z_c als konstant betrachtet. Der senkrechte Abstand zwischen dem angenommenen Brennpunkt der Kamera und dem Drehpunkt des BRIO®-Testbeds wurde mit Hilfe eines Bandmaßes mit einen Wert von $1,188m$ gemessen. Da die x- und y-Koordinaten stark von der Position des BRIO®-Spiels unter der Kamera ab-

hängig sind, können diese von Versuch zu Versuch variieren. Aus diesem Grund werden diese Koordinaten erst im Anschluß an die Landmarkendetektion (Kapitel 3.2.3.3) zur Laufzeit bestimmt. Vereinfacht wird davon ausgegangen, dass die Kamera senkrecht nach unten ausgerichtet ist und somit der Bildmittelpunkt des Kamerabilds sich senkrecht unter der Kamera befindet. Folglich können die Koordinaten x_c und y_c der Kameraposition durch Umrechnung der Pixelkoordinaten der Bildmitte des Kamerabilds in Koordinaten bezüglich der Mitte des Spielbretts berechnet werden.

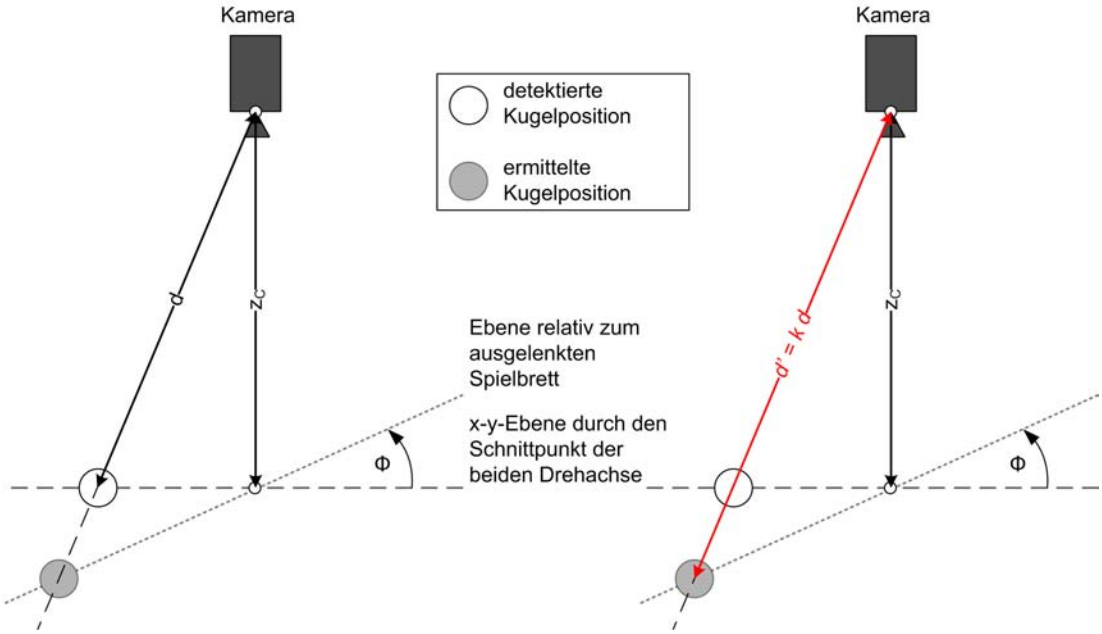


Abbildung 7.9: Berechnung der dreidimensionalen Koordinaten der Kugelposition als Schnittpunkt der Geraden durch die detektierte Kugelposition mit der ausgelenkten Spielebene

Somit stehen dem BRIO®-Testbed alle benötigten Informationen zur Berechnung der dreidimensionalen Koordinaten des Kugelmittelpunkts zur Verfügung. Die tatsächliche Kugelposition, relativ zum Schnittpunkt der beiden Drehachsen des BRIO®-Labyrinth-Spiels, liegt im Schnittpunkt der durch die Brettwinkel ϕ_X und ϕ_Y ausgelenkten Ebene (7.8) und der durch die Koordinaten der Kameraposition (x_c, y_c, z_c) und den Koordinaten der detektierten Kugel $(x', y', -0,0015m)$ (7.7) definierten Gerade (Abbildung 7.9). Hierbei muss die Lage des Schnittpunkts der beiden Drehachsen berücksichtigt werden. Dieser befindet sich $7,5mm$ über der Oberfläche des Spielbretts. Somit ist der angenommene Mittelpunkt der Kugel aufgrund ihres Radius von $6mm$ jeweils $1,5mm$ unterhalb der durch den Schnitt-

punkt verlaufenden x-y-Ebene zu finden.

$$Geradengleichung: \begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} + k \cdot \begin{pmatrix} x' - x_c \\ y' - y_c \\ -0,0015 - z_c \end{pmatrix} \tag{7.7}$$

$$Ebenengleichung: \begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0,0015 \end{pmatrix} + s \cdot \begin{pmatrix} \sin\phi_X \\ 0 \\ \cos\phi_X \end{pmatrix} + t \cdot \begin{pmatrix} 0 \\ \cos\phi_Y \\ \sin\phi_Y \end{pmatrix} \tag{7.8}$$

Setzt man beide Formeln gleich und löst das resultierende Gleichungssystem nach k der Geradengleichung auf, erhält man zur Berechnung von k folgende Formel (7.9).

$$k = \frac{z_c + 0.0015 - x_c \cdot \tan\phi_X - y_c \cdot \tan\phi_Y}{z_c + 0.0015 + (x' - x_c) \cdot \tan\phi_X + (y' - y_c) \cdot \tan\phi_Y} \tag{7.9}$$

Durch Kombination der Formeln (7.7) und (7.9) erhält man eine Berechnungsvorschrift der dreidimensionalen Koordinaten des Kugelmittelpunkts (x'', y'', z'') relativ zum Schnittpunkt der beiden Drehachsen des BRIO®-Spiels aus den beiden Brettwinkeln ϕ_X und ϕ_Y, den Koordinaten der Kamera (x_c, y_c, z_c) sowie den zweidimensionalen Koordinaten der im Kamerabild detektierten Kugel x' und y' (Abbildung 7.9).

7.4.2.3 Rotation der Kugelposition

Um die Koordinaten des Kugelmittelpunkts bezüglich der Ebene relativ zum ausgelenkten Spielbrett zu erhalten, müssen abschließend noch die ermittelten Kugelkoordinaten (x'', y'', z'') um die entsprechenden Winkel des Spielbretts $-\phi_X$ und $-\phi_Y$ rotiert werden. Zur Rotation der Koordinaten wird das in Kapitel 7.4.1 eingeführte Verfahren genutzt.

$$\begin{pmatrix} x^* \\ y^* \\ z^* \end{pmatrix} = R_X \cdot R_Y \cdot \begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} \tag{7.10}$$

7.5 Untersuchung des Fehlers der Kugelposition

In dem nun folgenden Abschnitt soll der Fehler der detektierten Kugelposition untersucht werden. Dazu wurde der jeweilige Fehler der detektierten Kugelposition des BRIO®-Testbeds ohne (*alte Methode*) und mit (*neue Methode*) des

in Kapitel 7.4.2 entwickelten Verfahrens und der Kompensation der Kameraverzeichnung (Kapitel 7.3) bei unausgelenktem Spielbrett ermittelt und miteinander verglichen. Im Anschluß wurden die Auswirkungen verschiedener Auslenkungen des Spielbretts auf den entstehenden Fehler näher untersucht.
Hierbei werden jeweils die entstehenden Fehler unter Verwendung des alten Verfahrens sowie zweier Varianten des neuen Verfahrens ermittelt und untersucht. Bei Variante A des neuen Verfahrens wird davon ausgegangen, dass sich die Kamera mittig über dem Spielbrett befindet, wohingegen bei Variante B die aus der Mitte verschobene Position der Kamera berücksichtigt wurde. Hierdurch sollen die Auswirkungen der Berücksichtigung der x- und y-Koordinaten der Kameraposition untersucht werden.

7.5.1 Untersuchung des Fehlers der Kugelposition bei unausgelenktem Spielbrett

In diesem Abschnitt soll der Fehler der verschiedenen Verfahren und Varianten bei unausgelenktem Spielbrett untersucht werden und die Ergebnisse der beiden Varianten des neuen Verfahrens mit den Ergebnissen der alten Methode verglichen werden.

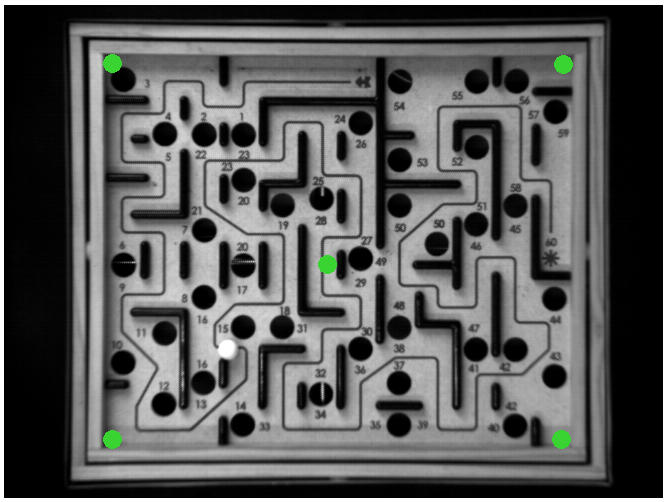


Abbildung 7.10: Testpositionen zur Bestimmung des Fehlers der Kugeldetektion des BRIO®-Testbeds

Zu diesem Zweck wurde die weiße Kugel nacheinander an fünf verschiedenen Positionen auf dem Spielbrett fixiert (Abbildung 7.10). Anschließend wurden jeweils 100 Mal pro Testposition die Koordinaten der detektierten Kugel mit den drei verschiedenen Verfahren aufgezeichnet.
Im Anschluß wurden die Referenzkoordinaten jeder der fünf Testpositionen per Hand ausgemessen und die mittlere Abweichung sowie die Standardabweichung

der detektierten Kugelpositionen von diesen Koordinaten berechnet. Die errechneten Werte sind für die x-Koordinate der detektierten Kugelposition in Tabelle 7.1 sowie für die y-Koordinate in Tabelle 7.2 angegeben.

Testposition	gemessene Kugelposition $x[mm]$	mittlerer Fehler der Kugelposition (altes Verfahren) $x[mm]$	mittlerer Fehler der Kugelposition (neues Verfahren; Variante A) $x[mm]$	mittlerer Fehler der Kugelposition (neues Verfahren; Variante B) $x[mm]$
oben links	$-131,5$	$-0,9 \pm 0,2$	$-1,7 \pm 0,0$	$-1,7 \pm 0,0$
oben rechts	$132,5$	$-2,4 \pm 0,0$	$-0,6 \pm 0,3$	$-0,6 \pm 0,1$
unten rechts	$132,5$	**-3,9 ± 0,0**	$-0,4 \pm 0,0$	$-0,4 \pm 0,0$
unten links	$-130,5$	$-1,0 \pm 0,0$	**-2,1 ± 0,0**	$-2,1 \pm 0,0$
Mitte	$-6,5$	$-1,3 \pm 0,0$	$-1,9 \pm 0,0$	**-2,4 ± 0,0**

Tabelle 7.1: Mittlere Abweichung der x-Koordinaten der vom BRIO®-Testbed ermittelten Kugelpositionen von der gemessenen Kugelposition für verschiedene Kugelpositionen und bei Spielbrett in Nullposition

Testposition	gemessene Kugelposition $y[mm]$	mittlerer Fehler der Kugelposition (altes Verfahren) $y[mm]$	mittlerer Fehler der Kugelposition (neues Verfahren; Variante A) $y[mm]$	mittlerer Fehler der Kugelposition (neues Verfahren; Variante B) $y[mm]$
oben links	$109,5$	$1,0 \pm 0,0$	**1,4 ± 0,3**	**1,5 ± 0,3**
oben rechts	$109,5$	$-0,6 \pm 0,0$	$1,0 \pm 0,2$	$1,0 \pm 0,2$
unten rechts	$-107,5$	**2,1 ± 0,0**	$-0,9 \pm 0,0$	$-1,1 \pm 0,2$
unten links	$-108,5$	$-0,6 \pm 0,0$	$-1,1 \pm 0,0$	$-1,1 \pm 0,0$
Mitte	$-6,5$	$0,5 \pm 0,3$	$0,4 \pm 0,3$	$0,1 \pm 0,0$

Tabelle 7.2: Mittlere Abweichung der y-Koordinaten der vom BRIO®-Testbed ermittelten Kugelpositionen von der gemessenen Kugelposition für verschiedene Kugelpositionen und bei Spielbrett in Nullposition

Tendenziell unterscheiden sich die entstehenden Fehler der detektierten Kugelposition aller drei Verfahren bei unausgelenktem Spielbrett nicht voneinander.
Als Ursachen für die Unterschiede in den mittleren Fehlern der Kugelposition wurden Schwankungen der Ausleuchtung des BRIO®-Testbeds während und zwischen den einzelnen Experimenten, Schwankungen bei der Bestimmung der Landmarken und damit voneinander abweichende Umrechnungen der Pixelkoordinaten

in die entsprechenden Brettkoordinaten angenommen. Zusätzlich muss die Messunsicherheit der fünf Testpositionen mittels eines Lineals ebenfalls berücksichtigt werden.
Mithin zeigen alle drei Verfahren annähernd ähnliches Verhalten bei unausgelenktem Spielbrett. Des Weiteren konnte kein Vorteil der Berücksichtigung einer verschobenen Kameraposition über dem sich in Nulllage befindenden Spielbrett erkannt werden.

7.5.2 Untersuchung des Einflusses der Brettwinkel auf den Fehler der Kugelposition

Im Anschluß sollen nun die Auswirkungen der Brettwinkel auf den Fehler der Kugelposition untersucht werden.

Die Kugel wurde dazu erneut an den verschiedenen Testpositionen auf dem Spielbrett fixiert. Daraufhin wurde die Kugelposition, wie bereits im vorherigen Abschnitt (Kapitel 7.5.1), 100 Mal pro Testposition bei unausgelenktem Spielbrett und anschließend jeweils 100 Mal für 5 verschiedene Auslenkungen des Spielbretts (Abbildung 7.11) bestimmt und aufgezeichnet.

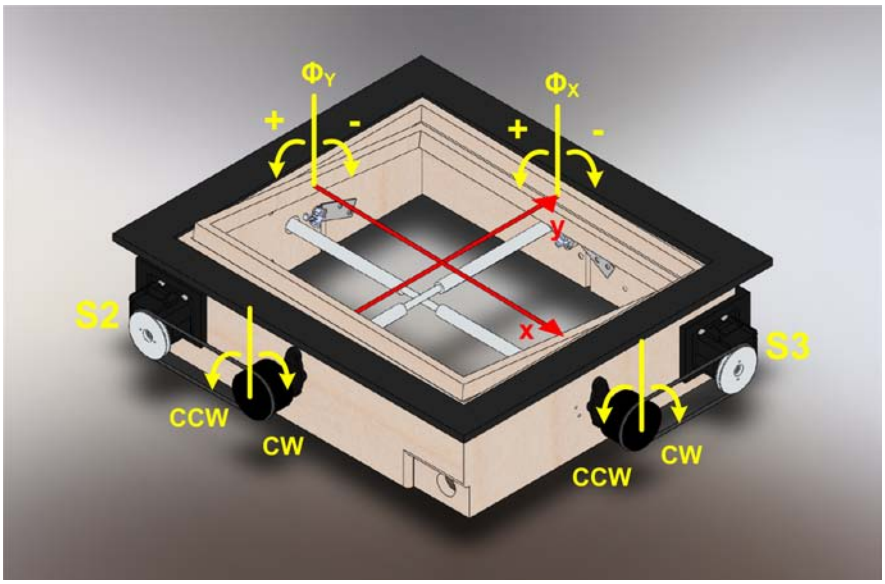


Abbildung 7.11: Zusammenhang zwischen den Drehrichtungen der Servomotoren $S2$ und $S3$ und den Brettwinkeln ϕ_X und ϕ_Y des BRIO®-Testbeds

Zu diesem Zweck wurde das Spielbrett je einmal für den Brettwinkel ϕ_X entgegen (CCW; Auslenkung I) und im Uhrzeigersinn (CW; Auslenkung II) maximal ausgelenkt, wobei ϕ_Y bei 0° gehalten wurde. Dazu wurde das Spielbrett so lange gedreht bis der jeweilige Endschalter auslöste. Analog wurde das Spielbrett für

den Brettwinkel ϕ_Y ebenfalls entgegen (CCW; Auslenkung III) und im Uhrzeigersinn (CW; Auslenkung IV) maximal ausgelenkt, diesmal für ϕ_X gleich 0°. Abschließend wurde das Spielbrett für beide Brettwinkel ϕ_X und ϕ_Y entgegen dem Uhrzeigersinn maximal ausgelenkt (Auslenkung V).

Bei der Auswertung der aufgezeichneten Brettwinkel diente der Mittelwert der 100 detektierten Kugelpositionen des unausgelenkten Spielbretts als Referenzposition der jeweiligen Testposition. Davon ausgehend wurde die mittlere Abweichung sowie die Standardabweichung der detektierten Kugelposition für verschiedene Auslenkungen des Spielbretts der einzelnen Testpositionen berechnet (Tabelle 7.3 und Tabelle 7.4) und miteinander verglichen.
Es hat sich gezeigt, dass mit Hilfe der neu entwickelten Methode die Abhängigkeit der Detektion der Kugelposition von der jeweiligen Auslenkung des Spielbretts verringert werden konnte. Während bei der bisherigen Kugeldetektion die Brettwinkel und die Verzeichnung der Kamera unberücksichtigt blieben und somit je nach Auslenkung die detektierte Kugelposition um bis zu $6,9mm$ in x-Richtung und bis zu $4,1mm$ in y-Richtung von der Referenzposition abwich, beträgt die maximale Differenz zwischen Referenzposition und detektierter Kugel bei ausgelenktem Spielbrett mit beiden Varianten des neuen Verfahrens für die x-Richtung $1,3mm$ und für die y-Richtung $1,1mm$ und liegt somit in der Größenordnung des Fehlers bei einem unausgelenkten Spielbrett.

7.6 Modellieren des Rauschens der Kugelposition

Zum Abschluss soll auch das Rauschen der detektierten Kugelposition modelliert werden. Anhand der aufgezeichneten Daten wird ein normalverteiltes Rauschen um die tatsächliche Kugelposition angenommen. Als Orientierung wurde hierbei die größte Abweichung der detektierten Kugelposition von der tatsächlichen Position des neuen Verfahrens unter Berücksichtigung einer verschobenen Kameraposition gewählt. Entsprechend kann das Rauschen der Koordinaten der detektierten Kugelposition durch ein um den Mittelwert 0 normalverteiltes Rauschen mit einer Standardabweichung von $2,4mm$ für die x-Koordinate und von $1,5mm$ für die y-Koordinate modelliert werden.

Test-position	Auslenkung	Abweichung von der Referenz-position (altes Verfahren) $x[mm]$	Abweichung von der Referenz-position (neues Verfahren; Variante A) $x[mm]$	Abweichung von der Referenz-position (neues Verfahren; Variante B) $x[mm]$
oben links	I	$-5,1 \pm 0,0$	$-1,1 \pm 0,00890$	$-0,4 \pm 0,00822$
	II	$1,1 \pm 0,2$	$-0,4 \pm 0,1$	$-1,0 \pm 0,1$
	III	$1,2 \pm 0,0$	$0,1 \pm 0,0$	$-0,2 \pm 0,0$
	IV	$-1,7 \pm 0,0$	$-0,6 \pm 0,0$	$0,1 \pm 0,3$
	V	$-2,6 \pm 0,3$	$-0,1 \pm 0,0$	$0,3 \pm 0,0$
oben rechts	I	$-0,6 \pm 0,0$	$0,7 \pm 0,0$	$0,2 \pm 0,0$
	II	$2,9 \pm 0,0$	$-0,3 \pm 0,0$	$0,4 \pm 0,0$
	III	$-1,2 \pm 0,0$	$0,3 \pm 0,0$	$0,1 \pm 0,1$
	IV	$0,6 \pm 0,0$	$0,0 \pm 0,0$	$0,4 \pm 0,0$
	V	$0,0 \pm 0,1$	$0,9 \pm 0,0$	$0,2 \pm 0,0$
unten rechts	I	$-0,3 \pm 0,3$	$0,7 \pm 0,3$	$0,2 \pm 0,0$
	II	$3,4 \pm 0,0$	$-0,7 \pm 0,2$	$0,5 \pm 0,1$
	III	$1,1 \pm 0,0$	$-0,2 \pm 0,3$	$0,5 \pm 0,0$
	IV	$-1,1 \pm 0,2$	$-0,1 \pm 0,1$	$0,2 \pm 0,1$
	V	$-1,1 \pm 0,2$	$0,4 \pm 0,3$	$0,2 \pm 0,1$
unten links	I	$-4,6 \pm 0,0$	$-0,4 \pm 0,0$	$0,3 \pm 0,0$
	II	$0,7 \pm 0,2$	$-0,4 \pm 0,0$	$-1,1 \pm 0,0$
	III	$-1,7 \pm 0,0$	$-0,7 \pm 0,0$	$-0,4 \pm 0,0$
	IV	$1,2 \pm 0,0$	$0,1 \pm 0,0$	$-0,2 \pm 0,0$
	V	$-6,9 \pm 0,0$	$-0,9 \pm 0,0$	$0,0 \pm 0,0$
Mitte	I	$0,0 \pm 0,0$	$0,4 \pm 0,0$	$-0,2 \pm 0,1$
	II	$0,3 \pm 0,3$	$0,0 \pm 0,3$	$-0,2 \pm 0,0$
	III	$0,0 \pm 0,0$	$0,0 \pm 0,0$	$0,0 \pm 0,2$
	IV	$0,0 \pm 0,1$	$0,0 \pm 0,0$	$0,0 \pm 0,0$
	V	$0,0 \pm 0,0$	$0,4 \pm 0,0$	$-0,1 \pm 0,0$

Tabelle 7.3: Vergleich der Abweichungen der x-Koordinate der mit den verschiedenen Verfahren detektierten Kugelposition für verschiedene Auslenkungen des Spielbretts

Test-position	Auslenkung	Abweichung von der Referenz-position (altes Verfahren) $y[mm]$	Abweichung von der Referenz-position (neues Verfahren; Variante A) $y[mm]$	Abweichung von der Referenz-position (neues Verfahren; Variante B) $y[mm]$
oben links	I	$2,3 \pm 0,0$	$0,6 \pm 0,0$	$0,7 \pm 0,0$
	II	$-1,6 \pm 0,3$	$0,2 \pm 0,0$	$0,4 \pm 0,0$
	III	$0,0 \pm 0,0$	$0,7 \pm 0,0$	$0,3 \pm 0,0$
	IV	$1,7 \pm 0,0$	$0,2 \pm 0,3$	$-0,1 \pm 0,0$
	IV	$1,7 \pm 0,0$	$-0,1 \pm 0,0$	$0,0 \pm 0,0$
oben rechts	I	$-1,7 \pm 0,1$	$0,7 \pm 0,2$	$0,7 \pm 0,1$
	II	$1,7 \pm 0,0$	$0,5 \pm 0,1$	$0,5 \pm 0,0$
	III	$0,0 \pm 0,0$	$0,5 \pm 0,0$	$0,5 \pm 0,0$
	IV	$1,2 \pm 0,1$	$0,2 \pm 0,1$	$0,2 \pm 0,0$
	V	$-1,7 \pm 0,0$	$0,7 \pm 0,0$	$0,7 \pm 0,0$
unten rechts	I	$2,3 \pm 0,0$	$0,4 \pm 0,0$	$0,5 \pm 0,0$
	II	$-2,3 \pm 0,0$	$0,0 \pm 0,0$	$-0,6 \pm 0,0$
	III	$-1,7 \pm 0,0$	$0,2 \pm 0,1$	$-0,3 \pm 0,0$
	IV	$0,4 \pm 0,2$	$0,0 \pm 0,0$	$0,0 \pm 0,2$
	V	$1,1 \pm 0,0$	$0,3 \pm 0,0$	$0,4 \pm 0,0$
unten links	I	$-2,3 \pm 0,1$	$-0,5 \pm 0,0$	$-0,3 \pm 0,3$
	II	$2,3 \pm 0,0$	$0,0 \pm 0,1$	$0,0 \pm 0,1$
	III	$-1,7 \pm 0,0$	$-0,4 \pm 0,0$	$-0,4 \pm 0,0$
	IV	$0,6 \pm 0,0$	$0,0 \pm 0,0$	$0,0 \pm 0,0$
	V	$-4,1 \pm 0,1$	$-1,3 \pm 0,1$	$-1,3 \pm 0,0$
Mitte	I	$0,2 \pm 0,3$	$0,1 \pm 0,3$	$0,0 \pm 0,0$
	II	$0,1 \pm 0,3$	$-0,1 \pm 0,3$	$0,0 \pm 0,1$
	III	$0,0 \pm 0,3$	$0,0 \pm 0,3$	$-0,4 \pm 0,2$
	IV	$0,3 \pm 0,2$	$0,1 \pm 0,1$	$-0,1 \pm 0,1$
	V	$-0,1 \pm 0,1$	$0,0 \pm 0,3$	$-0,3 \pm 0,3$

Tabelle 7.4: Vergleich der Abweichungen der y-Koordinate der mit den verschiedenen Verfahren detektierten Kugelposition für verschiedene Auslenkungen des Spielbretts

Kapitel 8

Optimierung der Motorparameter der BRIO®-Simulation

Ziel der Optimierung der Motorparameter war es, das Verhalten der Motoren der BRIO®-Simulation (Kapitel 3.3.1) dem Verhalten der Servomotoren des BRIO®-Testbeds anzupassen. Hierzu wurde das Verhalten der realen Motoren analysiert und versucht durch Anpassung geeigneter Parameter das Verhalten der simulierten Motoren zu optimieren.

Da in den Spezifikationen der Servomotoren [21] keinerlei Informationen zur internen Regelung zu finden waren, wurde das bereits in der Simulation implementierte Motormodell als Grundlage der Optimierung eingesetzt. Die Motoren der BRIO®-Simulation sind geschwindigkeitsgeregelt, d.h. das Verhalten der simulierten Motoren wird durch die Regelung der Drehgeschwindigkeit mittels eines *PID-Reglers* gesteuert. Hierbei wird die Drehgeschwindigkeit der Motoren v basierend auf der Differenz zwischen dem anzufahrenden und dem aktuellen Motorwinkel $\triangle\alpha$ nach Formel (8.1) geregelt.

$$v_n = P \cdot \triangle\alpha_n + I \cdot (\sum_{k \leq n} \triangle\alpha_k \cdot \triangle t_k) + D \cdot (\triangle\alpha_n - \triangle\alpha_{n-1}) \tag{8.1}$$

$$\triangle t_k = t_k - t_{k-1} \tag{8.2}$$

Die aktuelle Drehgeschwindigkeit des Motors v_n setzt sich aus drei unterschiedlich gewichteten Bestandteilen zusammen: Einem *proportionalen*, einem *integralen* und einem *abgeleiteten* Anteil.
Der proportionale Anteil der Drehgeschwindigkeit des Motors ergibt sich aus der Multiplikation der aktuellen Differenz zwischen Soll- und Istwinkel $\triangle\alpha_n$ und dem Parameter P.

Zur Berechnung des integralen Anteils der Drehgeschwindigkeit wird die Summe der Produkte aus allen vergangenen bis einschließlich der aktuellen Differenz zwischen Motor- und Zielwinkel $\triangle\alpha_k$ und dem zugehörigen Zeitintervall $\triangle t_k$ (Formel (8.2)) zwischen zwei Messungen k und $k-1$ des Motorwinkels mit dem Wert des Parameters I multipliziert.
Der abgeleitete Anteil ist das Produkt aus dem Parameter D und der Differenz zwischen aktueller und vorheriger Abweichung vom aktuellen Motorwinkel zum jeweiligen Zielwinkel $(\triangle\alpha_n - \triangle\alpha_{n-1})$. Je nach Wahl der Parameter verändert sich das Verhalten der Motoren der Simulation entsprechend. Im Rahmen dieser Master-Arbeit wurde sich für eine Optimierung aller drei Parameter P, I und D entschieden, um alle Möglichkeiten des PID-Reglers der simulierten Motoren, das Verhalten der realen Servomotoren nachzuahmen, ausschöpfen zu können.
Eine generelle Beschreibung des Aufbaus und der Funktionsweise eines PID-Reglers kann in [5] nachgeschlagen werden.

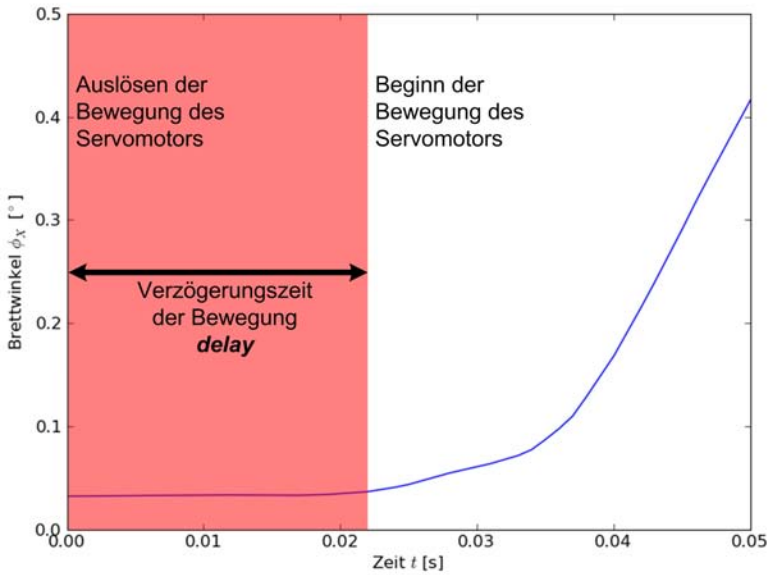


Abbildung 8.1: Darstellung der zeitlichen Verzögerung *delay*

Zusätzlich wurde bei der Untersuchung der beiden Servomotoren des BRIO®-Testbeds eine zeitliche Verzögerung zwischen dem Absenden eines Befehls und der Ausführung der Bewegung durch die Servomotoren festgestellt (Abbildung 8.1). Diese zeitliche Verzögerung der Reaktion des BRIO®-Testbeds ist auf die Übertragung des Befehls an die Servomotoren mittels der seriellen Schnittstelle des Steuer-PCs zurückzuführen. Um diese zeitliche Verzögerung innerhalb der Simulation zu berücksichtigen, wurde der zusätzliche Parameter *delay* eingeführt. In Abhängigkeit von seinem Wert wird die Ausführung der Drehbewegung der simulierten Motoren entsprechend verzögert.

8.1 Manuelle Optimierung der Motorparameter

Als erster Schritt wurde eine manuelle Kalibrierung der Parameter P, I, D und der Verzögerung *delay* der simulierten Motoren durchgeführt.

Zu diesem Zweck wurde zunächst das Verhalten der Servomotoren des BRIO®-Testbeds genauer untersucht. Hierzu wurden durch beide Servomotoren verschiedene Bewegungen jeweils 100 Mal durchgeführt und der zeitliche Verlauf des jeweiligen Brettwinkels ϕ_X bzw. ϕ_Y aufgezeichnet. Das Spielbrett wurde hierzu für den Brettwinkel ϕ_X jeweils von $0°$ auf $-5°$, $-4,5°$, ..., $4°$, $4,5°$, $5°$ und für den Brettwinkel ϕ_Y bedingt durch den Aufbau des BRIO®-Labyrinth-Spiels lediglich von $0°$ auf $-4°$, $-3,5°$, ..., $3°$, $3,5°$, $4°$ ausgelenkt. In Abbildung 8.2 ist beispielhaft die Messreihe für eine Drehbewegung des Servomotors $S2$ auf $5°$ dargestellt.

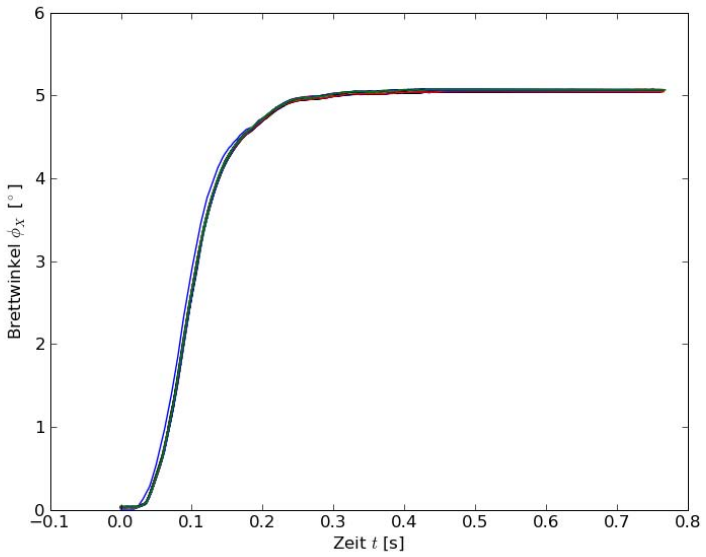


Abbildung 8.2: Beispielhafte Darstellung der Messreihe für eine Drehbewegung des Servomotors $S2$ von $0°$ auf $5°$

Anschließend wurden die 100 zeitlichen Verläufe einer einzelnen Messreihe gemittelt. Dazu wurde, jeweils in einem zeitlichen Abstand von $1ms$, vom Zeitpunkt $t = 0s$ ausgehend, der Mittelwert über alle 100 Brettwinkel zu den verschiedenen Abtastzeitpunkten bestimmt. Zu einem Zeitpunkt fehlende Brettwinkelwerte, entstanden durch zeitliche Verzögerungen der Komponenten des BRIO®-Testbeds, wurden mittels linearer Interpolation aus vorhergehendem und nachfolgendem Wert des Brettwinkels ermittelt.

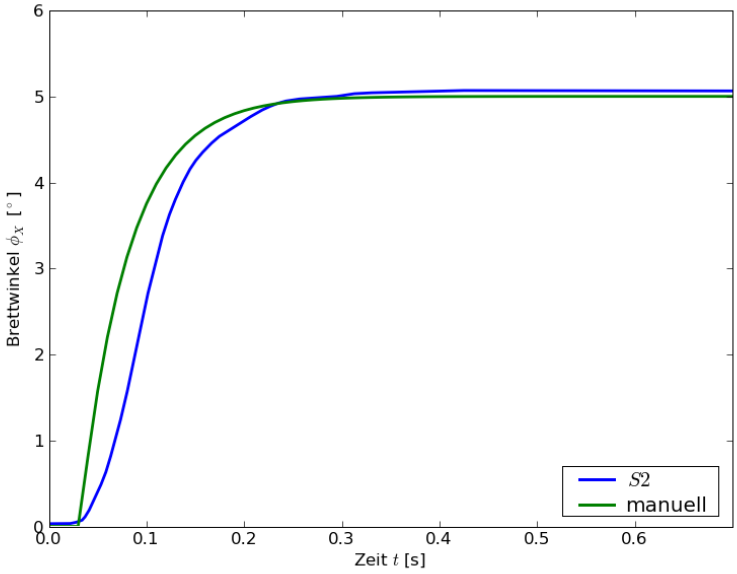


Abbildung 8.3: Vergleich der zeitlichen Verläufe der Mittelwerte einer Messreihe mit den Brettwinkeln der Simulation bei manuell optimierten Parametern

Unter Berücksichtigung der verschiedenen Referenzkurven der einzelnen Drehbewegungen der realen Servomotoren (blaue Kurve in Abbildung 8.3) wurden von Hand verschiedene Parameter-Konfigurationen der simulierten Motoren getestet und die einzelnen zeitlichen Verläufe der Brettwinkel der verschiedenen Bewegungen der Simulation (grüne Kurve in Abbildung 8.3) mit den Referenzkurven des Servomotors verglichen. Bei der groben manuellen Anpassung der Motorparameter der Simulation zeigten die simulierten Motoren bei einem P-Wert von 20, einem I- und D-Wert von 0 und einer zeitlichen Verzögerung *delay* von $0,03s$ ein zu den realen Motoren ähnliches Verhalten.

8.2 Optimierung der Motorparameter mittels Evolutionsstrategie

Aufgrund des hohen Aufwands der manuellen Optimierung der Motorparameter der Simulation wurde in einem zweiten Schritt versucht, die Anpassung der Motorparameter mittels *Evolutionsstrategie* automatisch durchzuführen. Das Verfahren der Evolutionsstrategie zur Optimierung von technischen Systemen nach Prinzipien der biologischen Evolution wurde von Ingo Rechenberg entwickelt [19].

Zur Anpassung der Parameter der simulierten Motoren wurde eine $(5 + 100)$-

Evolutionsstrategie in Python[1] implementiert. Ein einzelnes Individuum codiert einen Satz verschiedener Werte der vier zu optimierenden Parameter (P, I, D und *delay*). Die Population setzt sich aus jeweils 100 solcher Individuen zusammen. Zur Erzeugung eines Nachkommens wird aus den 5 Eltern-Individuen zufällig ein Individuum ausgewählt und dessen Satz von Parameterwerten mutiert. Bei der Mutation eines einzelnen Parameters ergibt sich der neue Wert des Parameters x_{n+1} aus der Summe des aktuellen Parameterwertes x_n und dem Produkt der Mutationsschrittweite des Individuums σ und einer normalverteilten Zufallszahl $N(0,1)$ (Formel 8.3).

$$x_{n+1} = x_n + \sigma \cdot N(0,1) \tag{8.3}$$

Die Mutationsschrittweite σ wird hierbei durch eine Variante des Verfahrens der *Mutativen Schrittweiten-Regelung* (MSR) beeinflusst. Bei diesem Verfahren wird jedem Individuum eine eigene Mutationsschrittweite zugeordnet und beim Erzeugen eines Nachkommens vor der Mutation der einzelnen Parameterwerte die Mutationsschrittweite des Nachkommen-Individuums σ_{n+1} aus der Schrittweite des Eltern-Individuums σ_n mittels (8.4) berechnet.

$$\sigma_{n+1} = \sigma_n \cdot N(0,1)^2 \tag{8.4}$$

Anschließend wird die Fitness aller Eltern- und Nachkommen-Individuen ermittelt und aus allen 105 Individuen die 5 Fittesten als Eltern der nächsten Generation von Nachkommen ausgewählt. Zur Berechnung der Fitness wurde ein Verfahren, welches im Rahmen des Brio2-Projekts[2] des DFKI Bremen entwickelt wurde, eingesetzt. Hierbei werden die vorher auf dem BRIO®-Testbed aufgezeichneten Bewegungen innerhalb der Simulation unter Verwendung der jeweiligen Parameterwerte der einzelnen Individuen wiederholt. Anschließend wird für jede dieser Bewegungen die Abweichung der Brettwinkel des BRIO®-Testbeds von den simulierten Brettwinkeln über die Zeit integriert. Hierbei stehen für beide Brettwinkel jeweils nur diskrete Werte zur Verfügung. Das Integral E der Abweichungen der simulierten Brettwinkel ϕ^S von den realen Brettwinkeln ϕ^R kann somit als Summe der Beträge der Differenz beider Brettwinkel multipliziert mit der Länge eines Zeitschritts $\triangle t$ von $1ms$ berechnet werden (Abbildung 8.4).
Aufgrund der unterschiedlichen Länge der einzelnen zeitlichen Verläufe des realen Systems wird das Integral E anschließend durch die Zeitdauer t_{ges} der aufgezeich-

[1] http://www.python.org/

[2] http://robotik.dfki-bremen.de/de/forschung/projekte/kognitive-robotik/brio-labyrinth-2.html

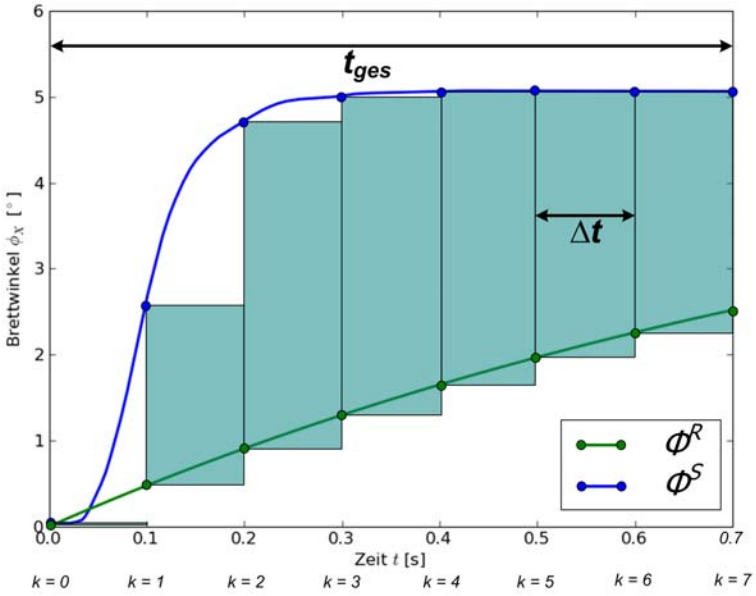


Abbildung 8.4: Darstellung des Integrals E der Abweichungen der simulierten Brettwinkel ϕ^S von den realen Brettwinkeln ϕ^R für Zeitschritte $\triangle t$ von $100ms$

neten Brettwinkel des BRIO®-Testbeds geteilt (Formel 8.5).

$$E = \frac{1}{t_{ges}} \cdot \sum_k |\phi_k^R - \phi_k^S| \cdot \triangle t \tag{8.5}$$

Der Wert der Fitness F eines einzelnen Individuums setzt sich somit aus der Summe der Integrale E aller aufgezeichneten Bewegungen eines Servomotors zusammen (Formel 8.6).

$$F = \sum_i E_i \tag{8.6}$$

Die Motorparameter der Simulation wurden, basierend auf den im vorherigen Abschnitt (Kapitel 8.1) aufgezeichneten Drehbewegungen des entsprechenden Servomotors $S2$ bzw. $S3$ des BRIO®-Testbeds, separat für beide simulierten Motoren optimiert. Nach jeweils 150 Generationen wurde die Evolutionsstrategie gestoppt und die angepassten Parameterwerte ausgelesen. Es ergaben sich folgende Werte:

Konfiguration	P	I	D	$delay[s]$
ES (S2)	24,515225	-0,000082	0,947691	0,048942
ES (S3)	20,235733	-0,000140	-7,330479	0,048993

Tabelle 8.1: Automatisch optimierte Motorparameter der beiden simulierten Motoren

Wie in Abbildung 8.5 beispielhaft dargestellt, konnte das Verhalten der simulierten Motoren bezüglich der Ähnlichkeit zum Verhalten der Servomotoren des

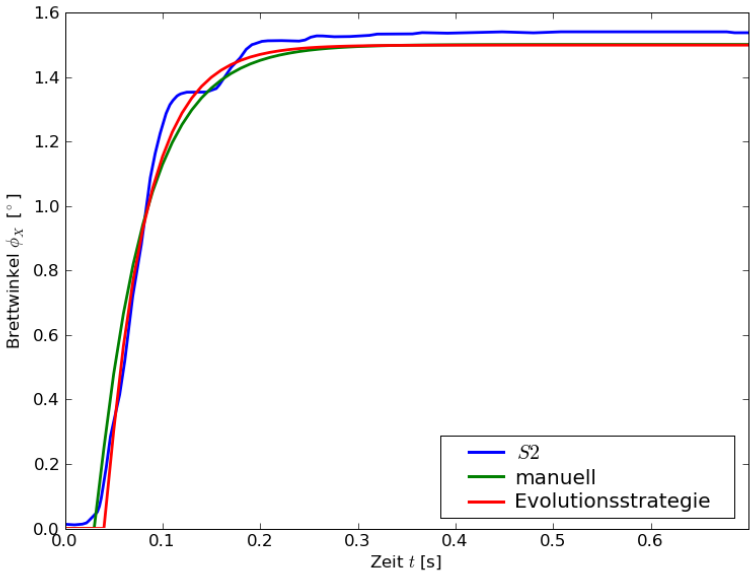


Abbildung 8.5: Vergleich der realen, der manuell und der automatisch optimierten Motorkurven, beispielhaft für den Servomotor $S2$

BRIO®-Testbeds optimiert werden.
Hierbei weichen die manuell angepassten Motorparameter nur geringfügig von den Werten der automatischen Kalibrierung ab.
Die Unterschiede in den P- und D-Werten sind auf unterschiedliche lokale Minima der jeweiligen Fitnessfunktion zurückzuführen. Bei mehreren Durchläufen der Simulation konvergierte die Evolutionsstrategie immer gegen andere Werte der einzelnen Parameter. Entsprechend wurden die Parametersätze mit den besten Fitnesswert nach der Evolution der 150 Generationen als Parametersatz der einzelnen simulierten Motoren ausgewählt.

Kapitel 9

Vergleich des Verhaltens beider Systeme

Zum Abschluß der Master-Arbeit sollen die eingeführten Anpassungen des BRIO®-Testbeds zur Minimierung des Simulation-Reality-Gaps zwischen der Simulation und dem BRIO®-Testbed untersucht werden. Zu diesem Zweck wurde eine Folge von Aktionen auf dem BRIO®-Testbed und innerhalb der Simulation unter Berücksichtigung der verschiedenen Anpassungen beider Systeme durchgeführt. Anschließend wurden die Trajektorien (Bahnkurven der Zustände eines Systems [3]) ausgewählter Sensoren miteinander verglichen.

9.1 Epsilon-Tunnel

Zum Vergleichen der Trajektorien der Zustände beider Systeme wurde ein sogenannter *Epsilon-Tunnel* eingesetzt. Bei diesem Verfahren wurde zunächst eine Folge von Aktionen 10 Mal auf dem BRIO®-Testbed ausgeführt. Der Zustand des Systems wurde dabei kontinuierlich aufgezeichnet, so dass im Anschluß daran mit Hilfe der verschiedenen Trajektorien der Zustandsraum des BRIO®-Testbeds zu den einzelnen Abtastzeitpunkten t_n mit einer festen Frequenz von $100Hz$ ermittelt werden konnte. Hierzu wurde der mittlere Zustand des realen Systems S_{mean} durch Mittelung der aufgezeichneten Zustände S_i des BRIO®-Testbeds zum jeweiligen Abtastzeitpunkt t_n berechnet (Formel (9.1)).

$$S_{mean}[t_n] = \frac{1}{10} \cdot \sum_{i=0}^{9} S_i[t_n] \tag{9.1}$$

Daraufhin wurden zu jedem einzelnen Abtastzeitpunkt t_n jeweils die größten positiven Abweichungen δ_+ und auch negativen Abweichungen δ_- der einzelnen

Sensorwerte vom mittleren Zustand des BRIO®-Testbeds bestimmt.

$$\epsilon_+[t_n] = \frac{1}{11} \cdot \sum_{k=-5}^{5} |\delta_+[t_{n+k}]| \tag{9.2}$$

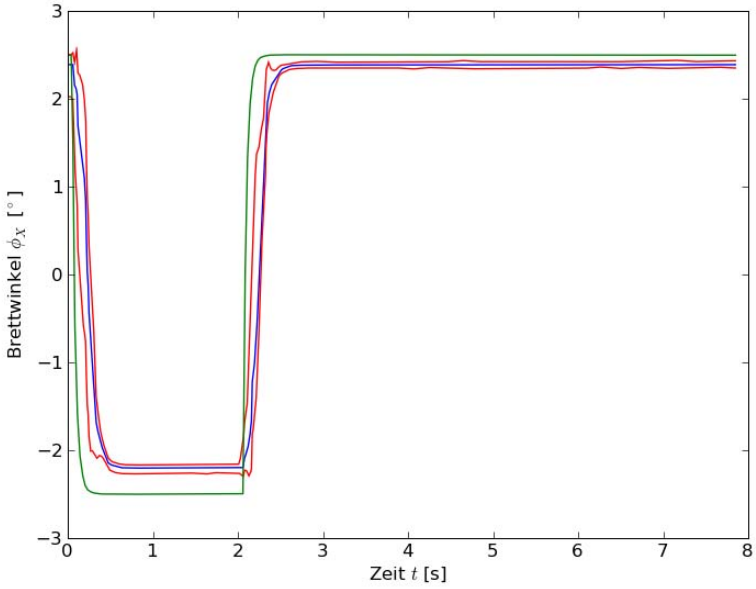
$$\epsilon_-[t_n] = \frac{1}{11} \cdot \sum_{k=-5}^{5} |\delta_-[t_{n+k}]| \tag{9.3}$$

Im Anschluß wurden die einzelnen Werte von ϵ_+ und ϵ_- zu den Abtastzeitpunkten t_n durch Mittelwertbildung aus dem aktuellen, den fünf vorherigen und den fünf nachfolgenden Werten von δ_+ bzw. δ_- geglättet (Formel (9.2) bzw. (9.3)).
Als nächstes wurde die gleiche Aktionsfolge innerhalb der BRIO®-Simulation einmal durchgeführt und dabei der Zustand der Simulation $S_{sim}[t_n]$ ebenfalls mit einer Abtastrate von $100Hz$ aufgezeichnet. Aufgrund der Determiniertheit der Simulation genügt ein einmaliges Ausführen der Aktionsfolge zur Aufzeichnung des Zustands der Simulation. Anschließend wurden die Trajektorien der Simulation mit denen des BRIO®-Testbeds verglichen (Abbildung 9.1). Zu diesem Zweck wurde die Anzahl der Abtastpunkte, für die sich der Zustand der Simulation innerhalb der Grenzen des Epsilon-Tunnels des BRIO®-Testbeds befindet, bestimmt. Der prozentuale Anteil der simulierten Zustände, welche (9.4) erfüllen, bezogen auf die Gesamtdauer der Aktionsfolge, wird im Rahmen dieser Master-Arbeit als Maß für die *Überdeckung* des Verhaltens beider Systeme verwendet.

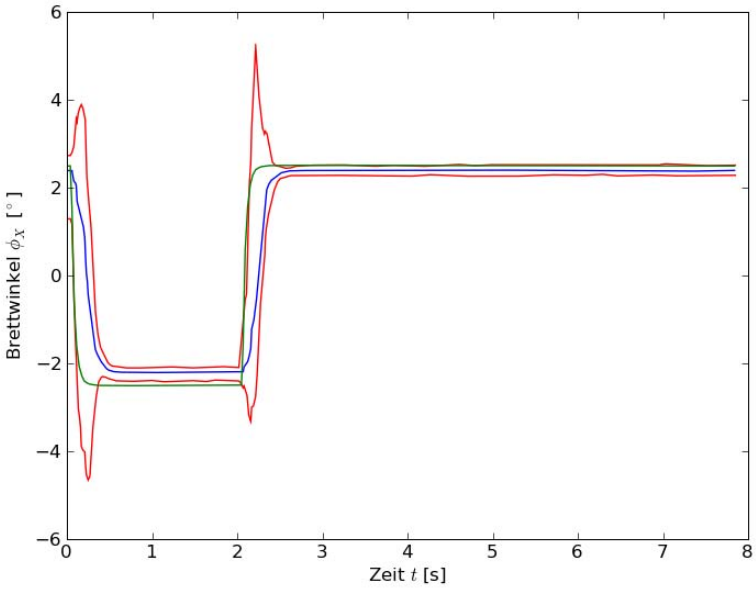
$$(S_{mean}[t_n] - k \cdot \epsilon_-[t_n]) \leq S_{sim}[t_n] \leq (S_{mean}[t_n] + k \cdot \epsilon_+[t_n]) \tag{9.4}$$

In (9.4) wurde zusätzlich ein Skalierungsfaktor k eingeführt. Dieser ermöglicht es, die Größe des Epsilon-Tunnels zu beeinflussen. Je nach Wert dieses Parameters kann der Durchmesser des Bereichs erlaubter Zustände vergrößert oder verkleinert werden (Abbildung 9.1).

Ein Vorteil dieser Herangehensweise zum Vergleichen des Verhaltens des BRIO®-Testbeds mit dem Verhalten des simulierten BRIO®-Labyrinth-Spiels besteht in der Berücksichtigung der Unsicherheit des BRIO®-Testbeds zu den einzelnen Abtastzeitpunkten. Je stärker die einzelnen Sensorwerte zu einem gewissen Zeitpunkt differieren, desto breiter wird der Bereich von erlaubten Sensorwerten. Anders ausgedrückt: Je stabiler das BRIO®-Testbed einen Zustand zu einem bestimmten Zeitpunkt einnimmt, desto kleiner wird der Epsilon-Tunnel zu diesem Zeitpunkt und damit der Bereich der dort erlaubten Zustände der Simulation.



(a) $k = 1$

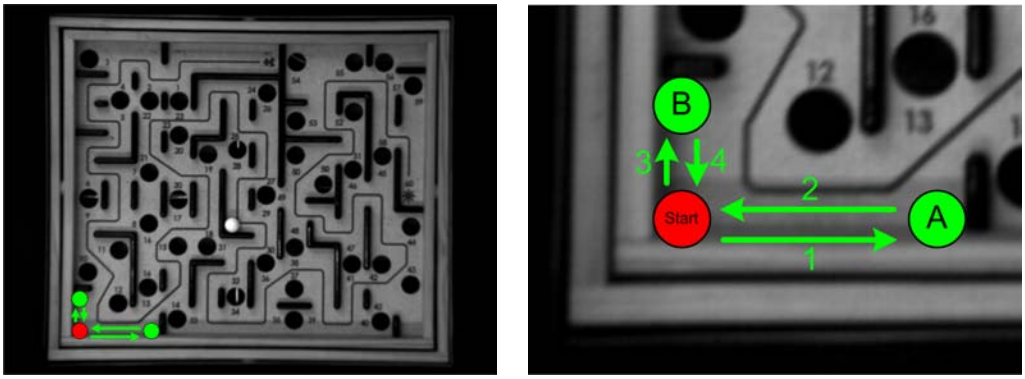


(b) $k = 3$

Abbildung 9.1: Beispielhafte Darstellung des *Epsilon-Tunnels* für $k = 1$ (a) und $k = 3$ (b) (*Erläuterung:* Die blaue Linie repräsentiert den zeitlichen Verlauf des gemittelten Brettwinkels ϕ_X des BRIO®-Testbeds, die beiden roten Linien sind die obere und untere Grenze des Epsilon-Tunnels des Brettwinkels ϕ_X und die grüne Linie stellt den Verlauf des Brettwinkels ϕ_X der Simulation dar.)

9.2 Aktionsfolge

Um das Verhalten des BRIO®-Testbeds mit dem Verhalten des simulierten BRIO®-Spiels vergleichen zu können, musste eine Folge von Aktionen, welche jeweils auf beiden Systemen ausgeführt werden konnte, ausgewählt werden. Die so aufgezeichneten Trajektorien sollten eine möglichst fundierte Aussage über eine Verbesserung bzw. Verschlechterung des Simulation-Reality-Gaps (Kapitel 1.2) zulassen. Zum einen sollte die Aktionsfolge Bewegungen der Kugel in x- wie auch in y-Richtung hervorrufen und zum anderen verschiedene Auslenkungen des Spielbretts für ϕ_X und ϕ_Y beinhalten.
Der Aufbau des Labyrinths verkomplizierte die Auswahl der Aktionsfolge, denn es musste eine Region des Spielbretts gefunden werden, wo die Kugel während der Ausführung der einzelnen Aktionen nicht durch eines der Löcher fallen konnte. Unter Berücksichtigung der vorher genannten Gründe wurde die untere linke Ecke des Spielbretts als Startposition ausgewählt (rote Kugel in Abbildung 9.2). Ein weiterer Vorteil dieser Brettregion ist die maximale Auswirkung der verschiedenen Auslenkungen der Spielfläche auf die Position der Kugel.



(a) gesamtes Spielbrett (b) Detailansicht

Abbildung 9.2: Darstellung der Aktionsfolge auf dem gesamten Spielbrett und in der Detailansicht

Zu Beginn der Aktionsfolge soll das Spielbrett für die Brettwinkel ϕ_X und ϕ_Y um jeweils $2,5°$ ($S2$ und $S3$ CCW, siehe Abbildung 7.11) ausgelenkt sein. Zum Startzeitpunkt der Aktionsfolge ($t = 0s$) soll zunächst mit dem Spielbrett für ϕ_X ein Winkel von $-2,5°$ ($S2$ CW) angefahren werden, wodurch die Kugel aus ihrer Startposition in positive x-Richtung rollt (Bewegung *1* in Abbildung 9.2) und sich anschließend an der Position A befindet. Wiederum zwei Sekunden später ($t = 2s$) soll das System erneut in den Startzustand übergehen, woraufhin die Kugel in negativer x-Richtung zurück rollt (Bewegung *2* in Abbildung 9.2). Analog soll zum Zeitpunkt $t = 4s$ mit dem Spielbrett für ϕ_Y ein Winkel von

$-2,5°$ ($S3$ CW) angefahren werden, so dass die Kugel in positive y-Richtung rollt (Bewegung *3* in Abbildung 9.2), und bei $t = 6s$ das System wieder in den Ausgangszustand zurückversetzt werden (Bewegung *4* in Abbildung 9.2). Nach $8s$ wird das Aufzeichnen des Zustands des Systems beendet.

9.3 Untersuchung der Anpassungen der Aktuatoren

Als erstes wurde der Einfluss der verschiedenen vorgenommenen Anpassungen der Aktuatoren sowohl in der Simulation als auch am BRIO®-Testbed untersucht. Hierzu wurden die Trajektorien beider Brettwinkel ϕ_X und ϕ_Y des BRIO®-Testbeds mit der *standardmäßigen* und der im Rahmen dieser Master-Arbeit entwickelten, *erweiterten* Ansteuerung der Servomotoren (Kapitel 6.2.2 und 6.3) aufgezeichnet und mit den Trajektorien der Brettwinkel der Simulation unter Verwendung von drei verschiedenen Sätzen von Motorparametern (Tabelle 9.1) mittels des Epsilon-Tunnels für verschiedene Werte von k verglichen.

Parametersatz der simulierten Motoren	P	I	D	$delay$ $[s]$
Standard	1,0	0,0	0,0	0,0
Manuell	20,0	0,0	0,0	0,03
ES (S2)	24,515225	-0,000082	0,947691	0,048942
ES (S3)	20,235733	-0,000140	-7,330479	0,048993

Tabelle 9.1: Verwendete Sätze von Motorparametern der Simulation

Bei der Auswertung der Trajektorien der Brettwinkel des BRIO®-Testbeds wurde festgestellt, dass mit den im Rahmen dieser Master-Arbeit entwickelten Verfahren der Positionierungsfehler des BRIO®-Testbeds nicht vollständig kompensiert werden konnte. Die Servomotoren erreichen trotz der vorgenommen Anpassung die vorgegebenen Brettwinkel meist nicht (Abbildung 9.3).
Des Weiteren werden die Zielpositionen der einzelnen Bewegungen innerhalb der Simulation im Vergleich zum BRIO®-Testbed schneller erreicht. Die Servomotoren des BRIO®-Testbeds bewegen sich besonders zu Beginn und am Ende einer Bewegung im Vergleich zu den Motoren der Simulation langsamer, da sie zu Beginn einer Bewegung zunächst Geschwindigkeit aufbauen müssen und bei Erreichen der Zielposition diese stärker als die Motoren der Simulation reduzieren. Die Zeitdauer zum Erreichen der Zielposition durch das reale Spielbrett ist somit länger. Entsprechend reagieren die Servomotoren des BRIO®-Testbeds im

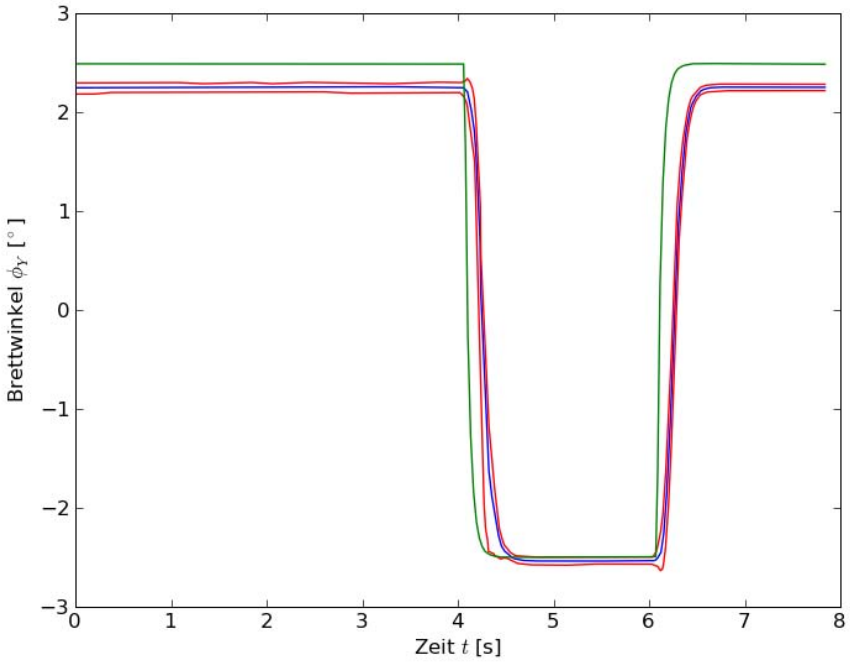


Abbildung 9.3: Positionierungsfehler und zeitlicher Versatz der Brettwinkel zwischen realem und simuliertem System (*Erläuterung:* Die blaue Linie repräsentiert den zeitlichen Verlauf des gemittelten Brettwinkels ϕ_Y des BRIO®-Testbeds, die beiden roten Linien sind die obere und untere Grenze des Epsilon-Tunnels des Brettwinkels ϕ_Y und die grüne Linie stellt den Verlauf des Brettwinkels ϕ_Y der Simulation dar.)

Vergleich zur Simulation träger.

Ansteuerung der Servomotoren des BRIO®-Testbeds	Parametersatz der Motoren der Simulation	Überdeckung des Brettwinkels ϕ_X der Simulation mit dem Epsilon-Tunnel des BRIO®-Testbeds in [%] für				
		$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Standard	Standard	3,82	5,22	6,75	7,26	7,77
Standard	Manuell	1,78	6,24	9,30	11,21	12,36
Standard	ES	1,78	6,50	9,55	11,59	12,36
Erweitert	Standard	13,74	31,30	42,75	45,67	47,84
Erweitert	Manuell	0,64	5,98	66,03	77,10	98,35
Erweitert	ES	0,76	5,34	73,79	77,35	98,47

Tabelle 9.2: Vergleich der Überdeckung des Brettwinkels ϕ_Y der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$

Wie in Tabelle 9.2 und 9.3 dargestellt, erreichen die simulierten Zustände für kleine Werte von k ($k \leq 2$) zumeist nur eine sehr geringe Überdeckung mit dem Epsilon-Tunnel des BRIO®-Testbeds. Grund hierfür ist der, verglichen mit dem Positionierungsfehler und der zeitlichen Verzögerung zwischen den Brettwinkeln

Ansteuerung der Servomotoren des BRIO®-Testbeds	Parametersatz der Motoren der Simulation	Überdeckung des Brettwinkels ϕ_Y der Simulation mit dem Epsilon-Tunnel des BRIO®-Testbeds in [%] für				
		$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Standard	Standard	7,77	12,99	17,83	26,88	40,13
Standard	Manuell	21,27	23,18	23,57	23,82	25,48
Standard	ES	18,60	23,31	23,69	24,08	26,75
Erweitert	Standard	32,70	39,31	44,15	48,35	51,15
Erweitert	Manuell	87,40	95,17	98,35	99,11	99,75
Erweitert	ES	88,67	95,67	98,60	99,75	100,00

Tabelle 9.3: Vergleich der Überdeckung des Brettwinkels ϕ_Y der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$

beider Systeme, recht schmale Epsilon-Tunnel. Bei Verwendung der Standardparameter scheint die Ähnlichkeit zwischen beiden Systemen, aufgrund des trägen Verhaltens der simulierten Motoren, für den Fall $k = 1$ bzw. $k = 2$ höher.
Für höhere Werte k ($k \geq 3$) ist durch die erhöhte Toleranz des Epsilon-Tunnels gegenüber Abweichungen zwischen den Trajektorien beider Systeme eine Zunahme der Überdeckung der Brettwinkel durch die vorgenommenen Anpassungen an beiden Systemen zu erkennen.
Sowohl die Verwendung der erweiterten Ansteuerung der Servomotoren als auch die Verwendung der optimierten Motorparameter verbessern den prozentualen Anteil der Überdeckungen der in der Simulation ermittelten Werte mit dem Epsilon-Tunnel des BRIO®-Testbeds. Mithin konnte durch die Verwendung der mittels Evolutionsstrategie optimierten Motorparameter gegenüber der Verwendung der manuell optimierten Motorparameter eine höhere Ähnlichkeit des Verhaltens der Aktuatoren beider Systeme erreicht werden.
Ferner wurden für den Brettwinkel ϕ_Y aufgrund des geringeren Positionierungsfehlers des Servomotors $S3$ gegenüber dem Servomotor $S2$ bessere Ergebnisse als für den Brettwinkel ϕ_X erzielt.

Zusammenfassend hat sich gezeigt, dass mit Hilfe der entwickelten Verfahren bzw. der vorgenommenen Anpassungen das Simulation-Reality-Gap zwischen der Simulation und dem BRIO®-Testbed bezüglich der Aktuatoren des simulierten und des realen Systems verringert werden konnte.

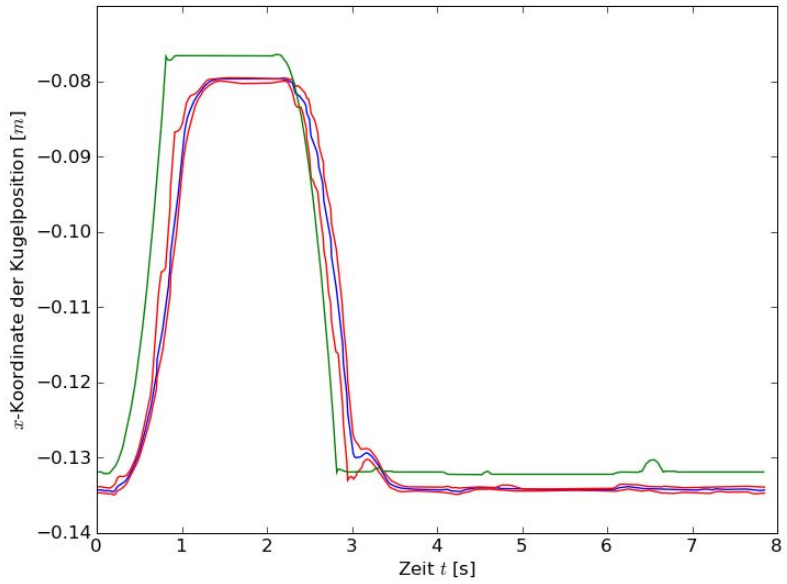
9.4 Untersuchung der Anpassungen der Bestimmung der Kugelposition

Als zweites wurden die Anpassungen des BRIO®-Testbeds und der Simulation bei der Erfassung der Kugelposition betrachtet. Hierzu wurden die Trajektorien der x- und y-Koordinaten der Kugelposition aus der Simulation den Trajektorien des BRIO®-Testbeds gegenübergestellt. Hierbei wurden die Trajektorien der Simulation bei *aktivierter* oder *deaktivierter* Berücksichtigung der Brettwinkel (Kapitel 7.4.1) bestimmt. Die Aufzeichnung der Trajektorien des BRIO®-Testbeds wurde unter Verwendung der *Standard-* bzw. der *erweiterten* Kugeldetektion (Kapitel 7.4.2) durchgeführt.
Die erste Auswertung der Daten des Epsilon-Tunnels ließ jedoch nur schwer Aussagen über eine Verbesserung bzw. Verschlechterung der Dynamik oder der Bestimmung der Kugelposition durch die vorgenommenen Anpassungen zu. Aufgrund einer systematischen Abweichung sowie einem zeitlichen Versatz zwischen der realen und der simulierten Kugelposition konnten die Trajektorien beider Systeme nicht direkt miteinander verglichen werden. Des Weiteren wurde eine Abhängigkeit der y- von der x-Koordinate der realen Kugel festgestellt. Bewegungen der Kugel in x-Richtung beeinflussten, aufgrund der miteinander durch den Spielaufbau verbundenen Drehachsen, den Wert der y-Koordinate. In Abbildung 9.4 steigt der Wert der y-Koordinate (Abbildung 9.4 (b)) während der Bewegung der realen Kugel in positive x-Richtung (Abbildung 9.4 (a)), bleibt konstant für die Zeit in der sich die Kugel in x-Richtung nicht bewegt und sinkt beim Zurückrollen der Kugel in x-Richtung auf den Wert zu Beginn der Aktionsfolge wieder ab.

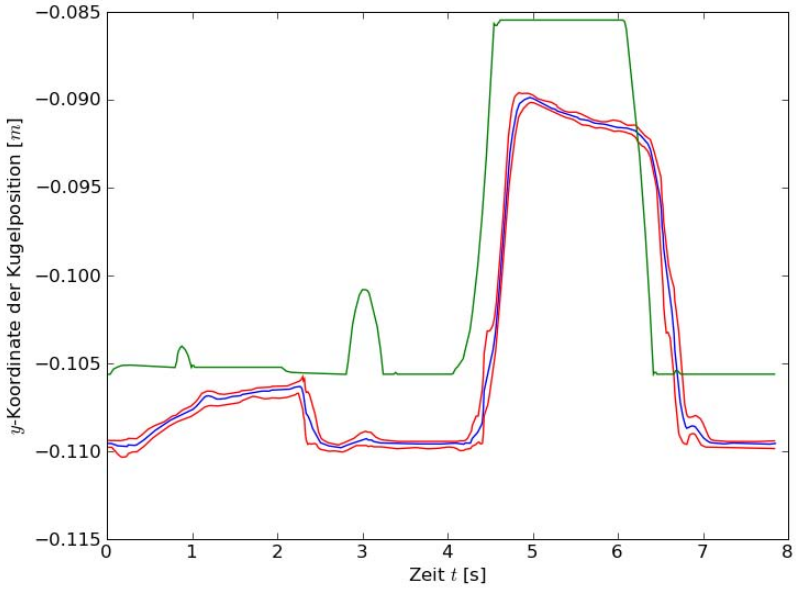
Mögliche Ursachen für die systematische Abweichung der beiden Kugelpositionen als auch der Abhängigkeit der Koordinaten der Kugelposition untereinander sollen im folgenden kurz dargestellt werden:

- Während der Suche nach möglichen Ursachen der beiden Phänomene wurde eine Verformung der Geometrie des Spielbretts festgestellt. Zwar sind die gegenüberliegenden Seitenränder des Spielbretts gleich lang, die Längen der beiden Diagonalen des Spielbretts weichen jedoch ($41,1cm \neq 40,5cm$) voneinander ab. Die Geometrie des Spielbretts entspricht somit nicht, wie bisher angenommen, einem Rechteck, sondern einem Parallelogramm.

- Die Blickrichtung der Kamera scheint nicht exakt senkrecht nach unten gerichtet zu sein. Die Geometrie des Spielbretts sowie die Position der Kugel



(a) x-Koordinate



(b) y-Koordinate

Abbildung 9.4: Vergleich der Trajektorien der Koordinaten der Simulation sowie des BRIO®-Testbeds (*Erläuterung:* Zu erkennen ist neben einer generellen Abweichung der Kugelpositionen eine zeitliche Verzögerung des BRIO®-Testbeds gegenüber der Simulation sowie eine Verzerrung des Verlaufs der Trajektorie des realen Systems. Die blaue Linie repräsentiert den zeitlichen Verlauf der jeweiligen Kugelkoordinate des BRIO®-Testbeds, die beiden roten Linien sind die obere und untere Grenze des Epsilon-Tunnels und die grüne Linie stellt den Verlauf der Kugelkoordinate der Simulation dar.)

werden somit nur verzerrt aufgezeichnet. Auch die Bestimmung der Landmarken und der Kugelposition im Kamerabild wird durch den verschobenen Blickwinkel verfälscht.

- Die Verzerrung des Spielbretts durch die Kameraverzeichnung konnte nicht vollständig kompensiert werden, dies beeinflusst die Bildverarbeitung des BRIO®-Testbeds wiederum nachteilig.

- Die Positionen der detektierten Landmarken stellen eine weitere Fehlerquelle dar. Je nach Ausleuchtung des BRIO®-Testbeds können diese um wenige Pixel von der tatsächlichen Position der Ecken des Spielbretts im Kamerabild abweichen und somit die Umrechnung der Kugelposition von Pixel- in Brettkoordinaten verfälschen.

Als mögliche Ursachen des zeitlichen Versatzes zwischen dem Verhalten der realen und der simulierten Kugel wurden folgende Fehlerquellen identifiziert:

- Wie im vorherigen Kapitel (Kapitel 9.3) festgestellt, können die angefahrenen von den vorgegebenen Brettwinkeln des BRIO®-Testbeds abweichen. Zudem benötigen die beiden Servomotoren im Vergleich zu den simulierten Motoren längere Zeit zum Erreichen der Zielposition. Aufgrund der trägeren Reaktion der Brettwinkel des BRIO®-Testbeds wirken sich deren Änderungen ebenfalls zeitlich verzögert auf die reale Kugel und deren Position aus.

- Ebenso könnte die Ursache der zeitlichen Verzögerung in einer Abweichung der realen und der simulierten Dynamik der Kugel liegen. Möglicherweise werden die komplexen dynamischen Vorgänge der realen Kugel (z.B. Reibung und Abpraller der Kugel an den Wänden des Labyrinths) mit der aktuellen Simulation nur ungenügend nachgebildet.

- Da die Geschwindigkeit der Kugel nicht direkt gemessen werden kann, wird zur Bestimmung der Kugelgeschwindigkeit ein Kalman-Filter eingesetzt (Kapitel 3.2.3.4). Dieser wird regelmäßig mit den aktuellen Werten der Brettwinkel sowie der detektierten Kugelposition des BRIO®-Testbeds aktualisiert und soll darauf basierend die Position sowie die Geschwindigkeit der realen Kugel schätzen. Folglich beinhaltet der Zustand des BRIO®-Testbeds nicht die direkt gemessene Kugelposition, sondern lediglich eine Schätzung derselben. Aufgrund dessen könnte die zeitliche Verzögerung der Trajektorien auf die Verwendung des Kalman-Filters zurückzuführen sein. Eventuell betrachtet der Kalman-Filter die Messwerte des BRIO®-Testbeds als zu unsicher und passt deshalb nur zögerlich seine Schätzungen entsprechend der realen Werte an.

Um die Auswirkungen der im Rahmen dieser Master-Arbeit eingeführten Anpassungen der Bestimmung der Kugelposition sowohl des BRIO®-Testbeds als auch der Simulation trotzdem untersuchen zu können, wurde die systematische Abweichung zwischen den aufgezeichneten Werten beider Systeme durch entsprechende Verschiebung der Werte der Trajektorien des simulierten Systems um die Abweichung der Startzustände beider Systeme manuell kompensiert. Daraufhin wurden die prozentualen Überdeckungen der jeweiligen verschobenen Trajektorien der Kugelkoordinate der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds bestimmt und hinsichtlich der Auswirkungen der einzelnen, an beiden Systemen vorgenommenen Anpassungen ausgewertet (Tabelle 9.4 und 9.5).

Kugeldetektion des BRIO®-Testbeds	Berücksichtigung der Brettwinkel der Simulation	Überdeckung der x-Koordinate der simulierten Kugel mit dem Epsilon-Tunnel des BRIO®-Testbeds in [%] für				
		$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Standard	deaktiviert	27,50	31,18	43,35	52,09	58,68
Standard	aktiviert	27,25	31,69	42,84	52,09	58,30
Erweitert	deaktiviert	31,93	43,38	71,37	82,70	91,86
Erweitert	aktiviert	31,55	43,38	70,61	79,52	83,97

Tabelle 9.4: Vergleich der Überdeckung der x-Koordinate der Ballposition der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$

Kugeldetektion des BRIO®-Testbeds	Berücksichtigung der Brettwinkel der Simulation	Überdeckung der y-Koordinate der simulierten Kugel mit dem Epsilon-Tunnel des BRIO®-Testbeds in [%] für				
		$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Standard	deaktiviert	30,93	42,84	47,78	52,09	54,88
Standard	aktiviert	29,15	38,02	46,13	50,32	53,23
Erweitert	deaktiviert	38,93	47,96	52,67	59,41	67,68
Erweitert	aktiviert	30,53	44,78	49,74	57,12	63,23

Tabelle 9.5: Vergleich der Überdeckung der y-Koordinate der Ballposition der Simulation mit dem zugehörigen Epsilon-Tunnel des BRIO®-Testbeds für $k = 1, 2, 3, 4, 5$

Insgesamt hat sich gezeigt, dass mit Hilfe der vorgenommenen Änderungen und unter Vernachlässigung der systematischen Abweichungen die Unterschiede zwischen der Dynamik der simulierten und der realen Kugel verringert werden konnten.

Während der prozentuale Anteil der simulierten Zustände innerhalb des Epsilon-Tunnels des BRIO®-Testbeds durch die Erweiterungen der Kugeldetektion des

BRIO®-Testbeds eindeutig erhöht werden konnte, verschlechterte er sich größtenteils durch die Berücksichtigung der Brettwinkel bei der Bestimmung der Position der simulierten Kugel. Da während der einzelnen Schritte der Bestimmung der Kugelposition des BRIO®-Testbeds eine Vielzahl von Störfaktoren auftreten kann, sind keine genauen Aussagen darüber möglich, ob diese Verschlechterung auf einen Fehler der Rotation der Koordinaten der simulierten Kugel oder aber auf eine höhere tatsächliche Abweichung zwischen dem Verhalten beider Kugeln zurückzuführen ist.
Für kleine Werte des Parameters k ($k \leq 2$) wird für Bewegungen der Kugel sowohl in x- als auch in y-Richtung eine annähernd gleiche Überdeckung der simulierten Kugelposition mit dem Epsilon-Tunnel des BRIO®-Testbeds erreicht. In Folge der Abhängigkeit der y-Position vom Wert der x-Koordinate der realen Kugel ist der Anstieg der prozentualen Überdeckung der x-Koordinate mit zunehmend größeren k-Werten ($k \geq 3$) höher als der Anstieg der Überdeckung der y-Koordinate.

Schlußendlich konnte mit Hilfe der vorgenommenen Erweiterungen der Kugeldetektion des BRIO®-Testbeds das Simulation-Reality-Gap zwischen der Dynamik der realen und der simulierten Kugel verkleinert werden.

Kapitel 10

Ergebnisse

Auf den nun folgenden Seiten soll ein kurzer Überblick über die Ergebnisse der Arbeit sowie ein Ausblick auf mögliche zukünftige Arbeiten gegeben werden.

10.1 Zusammenfassung

Zielsetzung dieser Master-Arbeit war die Untersuchung und Minimierung des Simulation-Reality-Gap anhand eines BRIO®-Labyrinth-Spiels. Aufgrund der zeitlichen Beschränkung einer Master-Arbeit konnte eine vollständige Analyse und Optimierung des Verhaltens beider Systeme nicht durchgeführt werden. Statdessen wurden die durchgeführten Untersuchungen auf ausgewählte Komponenten beider Systeme beschränkt. Hierbei wurden vor allem die Aktuatoren und die Bestimmung der Kugelposition beider Systeme sowie die Winkelmessung des BRIO®-Testbeds untersucht und versucht die auftretenden Fehler durch ensprechende Anpassungen oder Erweiterungen zu reduzieren.

Um das Verhalten beider Systeme miteinander vergleichen zu können, wurde ein Verfahren (Kapitel 9.1) entwickelt, welches die Ähnlichkeit des Verhaltens bzw. der Trajektorien zweier Systeme quantifizieren kann. Bedauerlicherweise wurde in den bisherigen Arbeiten (Kapitel 2) meist die Performance der einzelnen Agenten innerhalb der Simulation mit deren Performance nach der Übertragung auf das reale System verglichen. Dabei wurden lediglich die erreichten Fitnesswerte der Agenten auf dem realen Systems bzw. in der Simulation miteinander verglichen, nicht jedoch deren Verhalten bzw. die Abweichung des Verhaltens beider Systeme. Aufgrund dessen lassen sich die Ergebnisse dieser Arbeit nur schwer mit den Ergebnissen vergangener Arbeiten vergleichen.

Im Verlauf der Untersuchungen der Aktuatoren des BRIO®-Testbeds (Kapitel 6)

konnte die Positioniergenauigkeit der Servomotoren durch die Berücksichtigung des Spiels der Gummiriemen sowie des lokalen Positionierungsfehlers bei der Ansteuerung der Motoren verbessert werden. Eine vollständige Kompensation des Positionierungsfehlers des BRIO®-Testbeds konnte jedoch nicht erreicht werden. Zusätzlich erreichen die Servomotoren des BRIO®-Testbeds ihre Zielposition im Vergleich zu den Motoren der Simulation zeitlich verzögert. In Kombination mit der Verwendung der mittels Evolutionsstrategie optimierten Parameter der simulierten Motoren (Kapitel 8) konnten die Überdeckungen zwischen den Brettwinkeln beider Systeme deutlich verbessert werden. Die größte Überdeckung beider Systeme wurde durch Verwendung der erweiterten Ansteuerung der realen Servomotoren als auch den mittels Evolutionsstrategie optimierten Parameter der simulierten Motoren erreicht.

Im Gegensatz zu den Verbesserungen des Verhaltens der Aktuatoren konnten die Auswirkungen der Erweiterungen der Bestimmung der Kugelposition beider Systeme nicht direkt festgestellt werden. Mit Hilfe der vorgenommenen Erweiterungen der Kugeldetektion konnte der Einfluss verschiedener Auslenkungen des Spielbretts auf die Genauigkeit der Detektion der Kugel reduziert werden. Nichtsdestotrotz wurde neben einer Abweichung der Trajektorien der Kugelkoordinaten beider Systeme ebenfalls eine zeitliche Verzögerung zwischen den Koordinaten des BRIO®-Testbeds und der Simulation festgestellt.
Um die erhaltenen Daten trotzdem miteinander vergleichen zu können, wurden die Auswirkungen der systematischen Abweichung manuell korrigiert. Die anschließende Auswertung der angepassten Trajektorien zeigte eine Reduzierung des Simulation-Reality-Gaps zwischen den Kugelpositionen und damit auch der Dynamik der realen und der simulierten Kugel. In diesem Zusammenhang wurde festgestellt, dass durch die Erweiterung der Bestimmung der Kugelpositon des BRIO®-Testbeds die Überdeckung der Kugelpositionen beider Systeme eindeutig verbessert werden konnte. Die Auswirkungen der Berücksichtigung der Brettwinkel des simulierten Spielbretts konnten allerdings nicht abschließend ermittelt werden.

Zusätzlich wurde parallel zur Untersuchung der einzelnen Komponenten des BRIO®-Testbeds jeweils für die Brettwinkel, die Positioniergenauigkeit der Servomotoren wie auch die detektierte Position der realen Kugel das jeweilige Rauschen untersucht und versucht Modelle des Rauschens der einzelnen Werte zu erstellen. Somit können in weiterführenden Untersuchungen die erstellten Rauschmodelle verwendet werden, um die Unsicherheit des realen Systems innerhalb der Simulation

nachzuahmen.

10.2 Ausblick

Wie sich im Verlauf der Arbeit gezeigt hat, ist der ermittelte Zustand des BRIO®-Testbeds immer noch sehr unsicher. In weiteren Untersuchungen könnte der Versuch unternommen werden, die Genauigkeit einzelner Komponenten durch Entwicklung und Implementierung geeigneter Methoden zu erhöhen.

- Es könnte beispielsweise versucht werden, die Drehbewegungen der beiden Servomotoren mittels eines gespannten Zahnriemens auf die Achsen des BRIO®-Labyrinth-Spiels zu übertragen und die Übertragung somit weniger fehleranfällig zu realisieren.

- Des Weiteren wäre eine gegen Schwankungen der Be- und Ausleuchtung des BRIO®-Testbeds robustere Bestimmung der Kugelposition von Vorteil. Hierzu könnte versucht werden, die Position der Kugel anhand ihrer Form bzw. ihres Musters im Kamerabild zu detektieren.
  Im Rahmen dieser Master-Arbeit wurde ebenfalls ein Algorithmus zur Kugeldetektion unter Verwendung einer Farbsegmentierung im HSV-Farbraum implementiert. Aufgrund von Problemen der Detektion der weißen Kugel wurde auf eine genauere Untersuchung und Diskussion des Algorithmus verzichtet.

- Ein weiterer interessanter Ansatz ist die Verwendung eines adaptiven Neuronalen Netzes zur Berücksichtigung des lokalen Positionierungsfehlers und des Spiel der Gummiriemen bei der Ansteuerung der Servomotoren. Hierbei könnte in einem ersten Schritt ein Neuronales Netz anhand der im Rahmen dieser Arbeit aufgezeichneten Daten trainiert und anschließend während der Laufzeit des Systems die Gewichte der einzelnen Verbindungen optimiert werden. Hierbei könnten verschiedene Verfahren zur Adaption der Gewichte auf ihre Anpassungsfähigkeit an die Eigenheiten des BRIO®-Testbeds untersucht werden.

Durch die erreichte Verringerung des Simulation-Reality-Gaps zwischen beiden Systemen, aufgrund der Verbesserung der Genauigkeit des BRIO®-Testbeds bzw. einzelner Komponenten, stehen beide Systeme auch für weitere Forschungsarbeiten zur Verfügung.

- Um die Untersuchung des Simulation-Reality-Gap zwischen den beiden Systemen zu vervollständigen, sollte weiterführend die Bestimmung der Kugelgeschwindigkeit des BRIO®-Testbeds näher betrachtet werden. Hierbei könnten verschiedene Verfahren auf ihre Eignung, die jeweilige Geschwindigkeit der Kugel zu ermitteln, untersucht und die mit ihnen erzielten Genauigkeiten verglichen werden. Vorstellbar ist in diesem die Untersuchung von Verfahren zur Bestimmung der Kugelposition mittels Induktion, Magnetismus, Druck, Infrarotlicht, etc.

- Zusätzlich könnten die verschiedenen im State of the Art (Kapitel 2) vorgestellten Methoden zum Minimieren des Simulation-Reality-Gap und deren Auswirkungen auf die Performance eines Agenten nach dem Übertragen auf das reale System anhand des BRIO®-Testbeds und der MARS-Simulation untersucht werden. Aber auch eine Analyse verschiedener Agenten oder Lernverfahren auf ihre Befähigung das Simulation-Reality-Gap zu überbrücken wäre denkbar.

- Wie sich im Verlauf der Arbeit gezeigt hat, kann sowohl das Verhalten des BRIO®-Testbeds als auch das der Simulation mit einer Vielzahl von Parametern beeinflusst werden. Erste Schritte einer manuellen Anpassung der einzelnen Parameter wurden hier bereits durchgeführt. Da sich jedoch die einzelnen Parameter untereinander beeinflussen, wäre eine manuelle Optimierung beider Systeme sehr zeitaufwendig.
  Besser wäre es, den Prozess der Kalibrierung beider Systeme zu automatisieren. Hierbei werden verschiedene Aktionsfolgen auf dem realen System durchgeführt und die erhaltenen Sensorwerte aufgezeichnet. Anschließend können mittels verschiedener Verfahren die Parameter der Simulation entsprechend einer Verringerung des Simulation-Reality-Gaps optimiert werden.

Anhang A

Boxplot

Der *Boxplot* oder auch das *Kisten-Diagramm* ist ein Diagramm, welches eingesetzt wird, um numerische Daten mehrerer Messreihen einfach vergleichen zu können. Anhand eines Boxplots soll der Betrachter schnell einen Überblick über den Wertebereich sowie die Verteilung der numerischen Daten bekommen. Zu diesem Zweck werden der Median, die beiden Quartile als auch die Extremwerte oder „Ausreißer“ in kompakter Form dargestellt (Abbildung A.1).

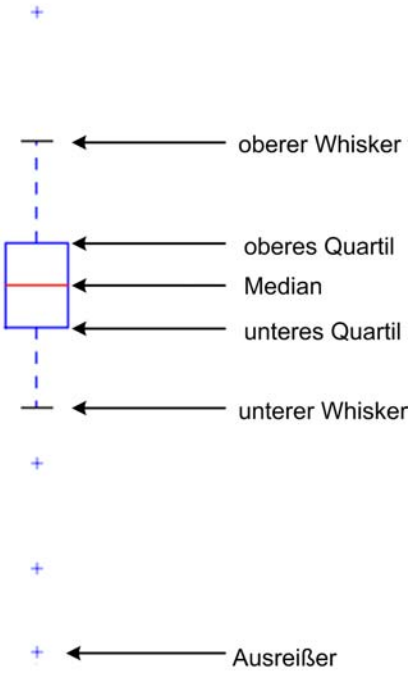


Abbildung A.1: Darstellung der einzelnen Komponenten eines Boxplots

Ein Boxplot besteht immer aus einer Box (*blaues Rechteck* in Abbildung A.1) und zwei Linien (*gestrichelte blaue Linien* in Abbildung A.1), welche die Box verlängern. Diese Linien, auch *Whisker* genannt, werden durch einen Strich abgeschlossen.

Die *rote Linie* in der Mitte des Boxplots kennzeichnet den Median der Verteilung der nummerischen Daten.
Die vertikale Ausdehnung der Box ist hierbei von den Werten des unteren und des oberen Quartils bestimmt. Quartile (siehe *Quantile* [3]) teilen, wie der Name bereits andeutet, die zugrundeliegende Verteilung in je vier Intervalle. Das untere Quartil beschreibt den Wert der Verteilung, für den 25% der Werte kleiner und 75% größer sind. Analog sind 75% der Werte kleiner und 25% größer als der Wert des oberen Quartils. Somit liegen mindestens 50% der Daten innerhalb des Bereichs zwischen oberem und unterem Quartil.
Es wird davon ausgegangen, dass sich die Werte des Großteils der Daten im Bereich zwischen den beiden Whiskers befinden. Die Länge der beiden Whiskers ergibt sich hierbei aus einem Vielfachem des Abstandes zwischen oberem und unterem Quartil (*Interquartilabstand*). Das Ende des unteren Whiskers erhält man, indem man vom Wert des Medians den 1,5-fachen Interquartilabstand abzieht und auf den nächstkleineren in der Verteilung vorkommenden Wert abrundet. Entsprechend erhält man das Ende des oberen Whiskers, indem man zum Wert des Medians den 1,5-fachen Interquartilabstand hinzuaddiert und aufrundet.
Alle Daten der Verteilung, deren Werte außerhalb des Bereichs von oberem und unterem Whisker liegen, werden als Ausreißer betrachtet und im Boxplot entsprechend mit einem *blauen Kreuz* markiert.
Anhand der Lage des Medians innerhalb der Box und der Anzahl an Ausreißern kann man Rückschlüße auf die Art der Verteilung der Werte ziehen (z.B. symmetrische oder schiefe Verteilung; normal-, gleich- oder andersartig verteilt).

Literaturverzeichnis

[1] J. C. Bongard and H. Lipson. Nonlinear system identification using coevolution of models and tests. *IEEE Transactions on Evolutionary Computation*, 9(4):361–384, 2005.

[2] Jean-Yves Bouguet. *Camera Calibration Toolbox for MATLAB*, 2008. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[3] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 5., überarbeitete und erweiterte auflage edition, 2001.

[4] E. A. Di Paolo. Homeostatic adaptation to inversion in the visual field and other sensorimotor disruptions. In *From Animals to Animats VI: Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, pages 421–429. Cambridge, MA: MIT Press, 2000.

[5] Otto Föllinger. *Regelungstechnik - Einführung in die Methoden und ihre Anwendung*. Hüthig Verlag, 10., durchgesehen auflage edition, 2008.

[6] Chris Harris and Mike Stephens. A combined corner and edge detector. In *The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[7] Intel. *Open Source Computer Vision*, 2006. http://opencv.willowgarage.com/wiki/.

[8] Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adapt. Behav.*, 6(2):325–368, 1997.

[9] Nick Jakobi. Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In *Fourth European Conference on Artificial Life*, pages 348–357. MIT Press, 1997.

[10] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life:*

Proc. 3rd European Conference on Artificial Life, pages 704–720. Springer-Verlag, 1995.

[11] R. E. Kálmán. A new approach to linear filtering and prediction problems. *Transaction of the ASME - Journal of Basic Engineering*, pages 35–45, 1960.

[12] Henrik Hautop Lund and Orazio Miglino. From simulated to real robots. In *Proceedings of IEEE 3rd International Conference on Evolutionary Computation*. IEEE Press, 1996.

[13] Gerhard Merziger and Thomas Wirth. *Repetitorium der höheren Mathematik*. Binomi Verlag, 4. auflage edition, 2002.

[14] Orazio Miglino, Henrik Hautop Lund, and Stefano Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2:417–434, 1995.

[15] Orazio Miglino, Kourosh Nafasi, and Charles E. Taylor. Selection for wandering behavior in a small robot. *Artificial Life*, 2:101–116, 1994.

[16] Stefano Nolfi, Dario Floreano, Orazio Miglino, and Francesco Mondada. How to evolve autonomous robots: different approaches in evolutionary robotics. In *Artificial Life IV*, pages 190–197. MIT Press, 1994.

[17] Stefan Pohl. Entwicklung und umsetzung eines laserscannersystems zur erstellung von tiefenkarten. Master's thesis, Universität Rostock, 2006.

[18] PROSILICA. *Prosilica PvAPI*, 1.16 edition, December 2007.

[19] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, 1973.

[20] Malte Römmermann, M. Edgington, Jan Hendrik Metzen, José de Gea, Y. Kassahun, Frank Kirchner, G. Rudolph, T. Jansen, S.M. Lucas, and et al. Learning walking patterns for kinematically complex robots using evolution strategies. In *10th International Conference on Parallel Problem Solving from Nature*, volume 5199 of *Lecture Notes in Computer Science*, pages 1091–1100. Springer, 2008.

[21] ROBOTIS. *Dynamixel DX-113, DX-116, DX-117 User's Manual*, 2nd edition, 11 2005. http://www.robotis.com/.

[22] Russel Smith. *Open Dynamics Engine*, 2006. http://www.ode.org/.

[23] Werner A. Stahel. *Statistische Datenanalyse - Eine Einführung für Naturwissenschaftler*. vieweg Verlag, 5. überarbeitete auflage edition, 2008.

[24] Joseba Urzelai and Dario Floreano. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evol. Comput.*, 9(4):495–524, 2001.

[25] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

Rostock, den 31. März 2009

Ich, Constantin Bergatt (Student der Technischen Informatik an der Universität Rostock, Matrikelnummer 3201352), versichere an Eides statt, dass ich die vorliegende Master-Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

Constantin Bergatt