BRIO Labyrinth 2
-
Einrichtung als Testbed für Lernarchitekturen und EEG / fMRI Untersuchungen

Elsa Kirchner, Larbi Abdenebaoui, Jan Hendrik Metzen,
Marc Tabie, Johannes Teiwes, Constantin Bergatt,
and Frank Kirchner

15. April 2009

1 Zusammenfassung der im Projekt durchgeführten Arbeiten

Zusammenfassung

Es wurde die Ansteuerung des BRIO Labyrinths mit einem Joystick implementiert (AP1). Es ist möglich, das Spiel parallel oder diagonal anzusteuern. Weiterhin wurde ein Zentrieralgorithmus entwickelt der gewährleistet, dass die Spielfläche vor jedem Spielzyklus waagerecht ausgerichtet wird (AP2). Es wurde ein Ballmagazin entwickelt. Damit ist es jetzt möglich, nach dem Verlust einer Kugel eine neue auf die Startposition zu positionieren. Dies kann automatisch oder manuell geschehen. Weiterhin wurde die Kugeleinschlagsdetektion mit dem Piezosensor in der Bodenplatte des Spiels analysiert und verbessert. Der verbesserte Algorithmus wurde dazu verwendet, automatisch eine Kugel aus dem Ballmagazin auszulösen (AP3). Es wurde ein Echtzeit-Zustandsschätzer für das BRIO Labyrinth System entwickelt, der die Position des Balls, seine Geschwindigkeit sowie die momentane Brettneigung basierend auf dem Kamerabild und den Potentiometer-Werten schätzt (AP5). Es wurden Schnittstellen für das Maja Machine Learning Framework (MMLF) entwickelt, die erlauben, sowohl die simulierte als auch die reale Version des BRIO Labyrinth Systems anzusteuern. Basierend darauf wurde im MMLF ein Markovscher Entscheidungsprozess definiert und im MMLF enthaltene Reinforcement Learning Verfahren getestet (AP6). Desweiteren wurde ein Verfahren entwickelt, das automatisiert die Parameter der MARS Simulation so anpassen kann, dass das Simulation-Reality Gap verringert wird. Dieses Verfahren wurde an den Parametern der simulierten Servos geprüft (AP7).

AP1: Joystick Integration, Skalierung

Die Arbeiten wurden in Q01-2008 abgeschlossen.

Bisher war die Ansteuerung der Servomotoren nur aus einem Programm heraus mit festen Werten möglich. Jetzt sollen in Echtzeit die Werte eines Eingabegerätes verarbeitet werden. Die bereits vorhandene Software Umgebung des Brio-Labyrinthes wurde so erweitert, dass sie auch Joystickeingaben als Steuerwerte für die Servomotoren verarbeiten kann. Dazu wurde eine neue Klasse *Joystick_Control* implementiert. Diese Klasse bildet das Interface zum Joystick und stellt dessen aktuellen Werte bereit. Um die Werte zu lesen, kann sich jede andere Klasse ein Objekt von *Joystick_Control* erzeugen und auf die implementierten Funktionen zugreifen. In der eigentlichen Klasse *Joystick_Control* werden die Werte vom Joystick in Steuerungssignale für die Motoren übersetzt. Diese Werte werden noch nicht auf die Motoren übertragen sondern nur als Wert zurückgegeben. Über die Auswahl zweier verschiedener Modi (Siehe Abbildung .1) ist es möglich, die Achsen des Joysticks exakt auf die Achsen des Spiels zu übertragen oder die Joystickachsen diagonal auf die Spielachsen abzubilden. Diese Drehung um 45° kann für geübte BRIO-Spieler wichtig sein: Im Handbetrieb wird das Spiel in dieser Position gespielt, da man dabei bequem mit beiden Händen an die Drehknöpfe herankommt.

Abbildung .1: Links: um 45° gedrehte Ansteuerung, rechts: normale Ansteuerung

Nach der Initialisierung läuft das Programm dann als Thread weiter, um unabhängig von anderen Operationen zu sein. Dies ist vor allem deswegen nötig, da die Ansteuerung mit dem Joystick in Echtzeit erfolgen muss.

Integration des Joysticks Nach einiger Recherche zum Thema Python und Joystickansteuerung ergab sich, dass es am einfachsten ist, für unsere Aufgabe das Pythonmodul PYGAME [2] zu verwenden. Bei diesem Modul handelt es sich um Freeware, welche meistens zur Programmierung kleinerer Spiele genutzt wird. Da es sich bei einem Joystick um ein klassisches Spieleingabegerät handelt, verfügt PYGAME auch über eine Joystick Bibliothek [3].
Mit dieser Bibliothek wird ein Joystick Handle erzeug, hierbei werden an den Computer angeschlossenen Joysticks automatisch erkannt und erhalten fortlaufende Adressen, beginnend bei 0. Im Normalfall wird nur ein Joystick zurzeit an den Computer angeschlossen, daher wird standardmäßig mit der Adresse 0 gearbeitet.
Das erzeugte Handle besitzt mehrere Funktionen, von denen werden vornämlich zwei benutzt: Die Funktion $get\_axis(achse)$ dient zum Auslesen der Joystickauslenkung und $get\_button(knopf)$ zum Auslesen des Zustandes eines Knopfes des Joysticks. Als Übergabewert erwarten beide Funktionen einen Integerwert. Ähnlich wie bei der Joystickadresse handelt es sich bei diesem Wert auch um eine joystickinterne Adresse. Diese Adresse gibt an, welche Achse bzw. welcher Knopf des Joysticks ausgelesen werden soll.

- Die Funktion $get\_axis(achse)$ liefert Fließkomawerte zurück. Der Wertebereich liegt zwischen -1 und 1, die Auflösung beträgt sechs Stellen hinter dem Komma. Wenn der Joystick kalibriert ist repräsentiert der Rückgabewert 0 die Mittelsstellung des Joysticks. Negative Werte für die X-Achse entsprechen einer Auslenkung des Joysticks in negative X-Richtung im karthesischesn Koordinatensystem. Analog verhält es sich mit positiven Werten.

- Die Funktion $get\_button(knopf)$ liefert integer/boolean Werte zurück. Es können die Werte 0 und 1 angenommen werden. Hierbei entspricht 1 dem Zustand Knopf gedrückt und 0 Knopf nicht gedrückt.

Bei der Initialisierung von PYGAME wird automatisch ein Handle auf alle angeschlossenen Joysticks erstellt. Dieses Handle stellt alle beschriebenen Funktionen bereit. Das Auslesen und die Weiterverarbeitung der Joystickwerte erfolgt in einer Schleife. In dieser Schleife werden laufend alle Joystickwerte ausgelesen skaliert und weiterverarbeitet. Diese Schleife hat ein Flag als Abbruchkriterium und wird durch den Aufruf der *stop()* Funktion aus dem Hauptprogramm heraus abgebrochen.

Skalierung der Werte Im Rahmen dieses Arbeitspaketes wurden die Ausgelesenen Joystickwerte so aufgearbeitet, dass es nun möglich ist diese zur Ansteuerung des Brio-Labyrinthes zu nutzen.
Es gibt zwei Verfahren die Werte zu skalieren. Das erste Verfahren bildet die Joystickachsen direkt auf die Spielachsen ab. Diese Art der Ansteuerung ist die parallele Ansteuerung.
Das zweite Verfahren dreht die Joystickachsen um 45° und bildet sie dann auf die Spielachsen ab. Diese Art der Ansteuerung ist die diagonale Ansteuerung.

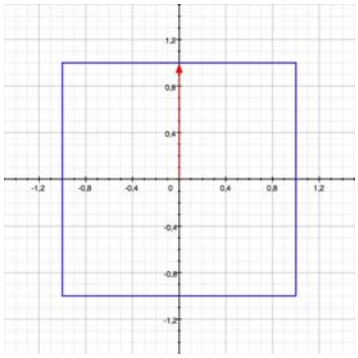
Parallele Ansteuerung: Bei der parallelen Ansteuerung wird je eine Joystickachse auf eine Labyrinthachse abgebildet. Der Wertebereich einer Joystickachse liegt zwischen -1 und 1. Die Joystickrohwerte werden als Fliesskommazahlen Zahlen mit einer Genauigkeit von sechs Stellen hinter dem Komma zurückgeliefert. Der Arbeitsbereich des Servomotors umfasst 300°, die in 1024 Schritte unterteilt sind. Damit ergibt sich für den Servomotor eine Auflösung von 0,293° pro Schritt.
Wir haben es uns zu Nutze gemacht, dass der Ausschlag jeder Fläche von der waagerechten Position in jede Richtung symmetrisch ist. Mit Hilfe eines Zentrieralgorithmus wird vor Begin eines Spieldurchgangs die Labyrinthfläche waagerecht ausgerichtet. Dieser Zentrieralgorithmus liefert des Weiteren vier Werte zurück. Für jede Achse wird die Servomotorposition für die Mittelstellung zurückgeliefert. Zusätzlich wird noch der Wert von dieser Position bis zum maximalen Ausschlag der Ebene in eine Richtung angegeben.
Die Skalierung wird im Folgenden am Beispiel einer Ebene unter der Annahme erklärt, dass sich die Servomotorposition für die Mittelstellung bei 512 befindet und der Vollausschlag der Ebene in jede Richtung ca. 8° beträgt.
Von der Mittelposition kann die Ebene maximal um 28 Positionen nach links bzw. nach rechts gedreht werden. Dies entspricht den Positionen 484 und 540, der Bewegungsraum ist also symmetrisch genau wie der Wertebereich des Joysticks.
Um die Nullpositionen beider Systeme übereinander zu bringen, wird zu dem Joystickwert 512 dazu addiert. Somit liefert der Joystick in der Mittelstellung den Wert 512 zurück, welcher die Servomotorposition für die Mittelstellung der Spielfläche repräsentiert.
Zu diesem Wert wird dann noch das gerundete Produkt aus Joystickrohwert und 28 dazuaddiert. Die Skalierungsformel sieht also wie folgt aus:

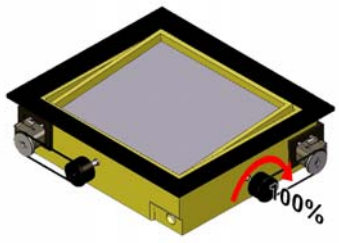
$$Servomotorstellwert = 512 + |Joystickrohwert * 28| \tag{.1}$$

Die in dem Beispiel verwendeten Werte für die Mittelposition und dem maximalen Ausschlag werden in dem realen System durch die von dem Zentrieralgorithmus bestimmten Werte ersetzt.

Diagonale Ansteuerung: Wenn ein Proband das Brio-Labyrinth von Hand spielt, steht es normalerweise diagonal vor ihm. Das ist komfortabler da die Position der beiden Drehgriffe des Labyrinths sich so am ehesten der Ergonomie des menschlichen Körpers anpasst. Da das Labyrinth für Vergleichsstudien der Hirnaktivitäten eines Probanden bei Hand- bzw. Joystickbetrieb genutzt werden soll, ist es sehr wichtig, dass die zwei Szenarien sich möglichst wenig voneinander unterscheiden. Daher ist es nötig die Joystickrohwerte um 45° zu drehen, damit das Labyrinth auch mit dem Joystick diagonal gespielt werden kann.

Bei den Joystickrohwerten handelt es sich um kartesische Koordinaten. Der Wertebereich beschreibt ein Quadrat mit der Kantenlänge 2 dessen Mittelpunkt sich im Ursprung des Koordinatensystems befindet. Jede Achse hat einen Wertebereich von -1 bis 1.





Da es sich um kartesische Koordinaten handelt, wurde die Punktrotation für den kartesischen 2D-Raum benutzt. Bei dieser Rotation wird ein Punkt, ein Wertepaar bestehend aus den Rohdaten der zwei Joystickachsen, um einen bestimmten Winkel gedreht. Dabei verändert sich der Abstand zu dem Ursprung nicht.
Die Rotation ergibt sich aus den folgenden Formeln:

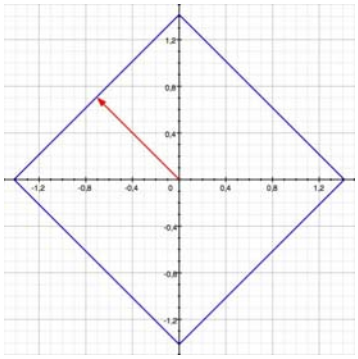
$$\varphi = 45° \tag{.2}$$

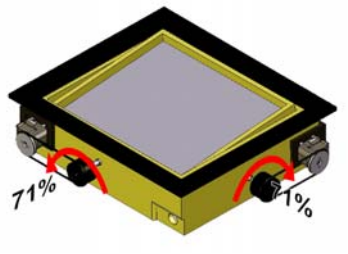
$$x' = x * cos(\varphi) - y * sin(\varphi) \tag{.3}$$

$$y' = x * sin(\varphi) + y * cos(\varphi) \tag{.4}$$

Dabei stellen x' und y' die rotierten Werte, x und y die Ausgangswerte dar. φ ist der Winkel um den gedreht wird.
Mit dieser Methode wurden die Joystickrohwerte um 45° gegen den Uhrzeigersinn (positive Winkelrichtung) gedreht.

Nach der Rotation erhält man die gedrehten Werte des Joysticks. Trägt man alle möglichen Wertepaare in ein Koordinatensystem ein erhält man eine auf der Spitze stehende Raute. Die Eckpunkte liegen bei $(\sqrt{2}, 0)$, $(0, \sqrt{2})$, $(-\sqrt{2}, 0)$ und $(0, -\sqrt{2})$.

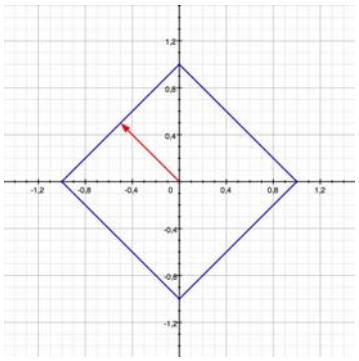


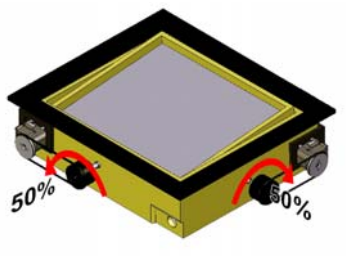


Die Drehung der Koordinaten reicht allerdings nicht aus. Die Werte müssen jetzt noch skaliert werden. Jetzt liegen die Joystickrohwerte bereits um 45° gedreht vor. Durch die Rotation bewegen sich die Joystickrohwerte der einzelnen Achsen nicht mehr zwischen -1 und 1 sondern zwischen $-\sqrt{2}$ und $\sqrt{2}$.

Bevor die Joystickrohwerte auf die Servomotorpositionen abgebildet werden, wird der Wertebereich der rotierten Joystickwerte denen der parallele Ansteuerung angenähert. Dafür wurde der Wertebereich von $-\sqrt{2}$ bis $\sqrt{2}$ für jede Achse auf -1 bis 1 skaliert, indem alle Joystickrohwerte durch $\sqrt{(2)}$ geteilt wurden.

Die Werte beschreiben weiterhin eine Raute im Arbeitsbereich. Jetzt ist der gesamte Bereich um den Faktor $\sqrt{2}$ kleiner. Die Eckpunkte liegen nun wieder innerhalb der gewünschten Grenzen. Jedoch besitzt der markierte Vektor nur noch eine Länge von $\frac{1}{\sqrt{2}}$.





In einem ersten Ansatz wurden diese Werte bereits auf die Servomotorpositionen abgebildet. Es war zu erkennen, dass diese Methode prinzipiell funktioniert. Jetzt ergab sich ein weiteres Problem: Die vollständige Auslenkung der einen Joystickachse resultierte wie gewollt in einer vollständigen Auslenkung beider Spielflächenachsen. Die vollständige Auslenkung beider Joystickachsen hingegen resultierte lediglich in einer halben Auslenkung eines Servomotors. Diese Reduktion des Bewegungsraumes ist die Folge der Rotation. Wie oben beschrieben wird zum Beispiel das Koordinatenpaar $(1, 1)$ auf den Punkt $(0,71, 0,71)$ abgebildet. Bei dem ersten Skalierungsschritt (teilen durch $\sqrt{2}$) wird daraus der Punkt $(0, 5, 0, 5)$. Dies erklärt den verringerten Bewegungsraum.
In dem nächsten Schritt wird das um 45° gedrehte Quadrat so skaliert, dass es wieder einem nicht gedrehten Quadrat ähnelt. Ziel ist es, den Arbeitsbereich so zu vergrößern,

dass eine vollständige Auslenkung einer Joystickachse in einer vollständigen Auslenkung der Spielfläche resultiert. Das bedeutet, dass die vier Punkte $(\pm 0,5, \pm 0,5)$ auf die vier Punkte $(\pm 1, \pm 1)$ abgebildet werden sollen. Zur Lösung diese Problems wurde folgende Skalierungsformel verwendet:

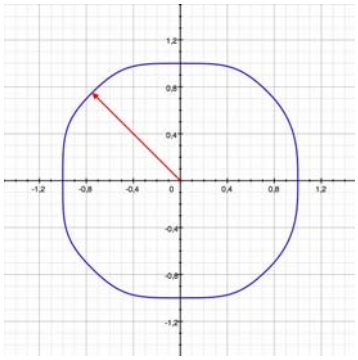
$$x = \begin{cases} x = & x * (1 + |x|) & wenn|x| & < & 0.5 \\ x = & x * (2 - |x|) & wenn|x| & \geq & 0.5 \end{cases} \tag{.5}$$

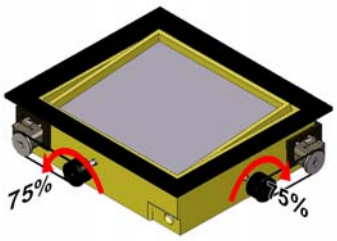
Analog dazu:

$$y = \begin{cases} y = & y * (1 + |y|) & wenn|y| & < & 0.5 \\ y = & y * (2 - |y|) & wenn|y| & \geq & 0.5 \end{cases} \tag{.6}$$

Die so mit den Formeln skalierten Werte wurde dann wie bei der parallelen Ansteuerung verwendet.

Werden die Formeln .5 und .6 auf die vorhandenen Werte angewendet nähert sich die Form des Arbeitsbereiches wieder eines Quadrats an. In den Bereichen in denen nur eine Joystickachse aktiv ist wird der Arbeitsbereich vergrößert.





Hat die Bewegung einer Joystickachse zuvor Auswirkung auf nur einen Wert des Koordinatenpaares gehabt, so beeinflusst sie jetzt beide Werte. Wenn man den Joystick gerade nach vorne auslenkt, neigt sich die Labyrinthfläche in die vorderste Ecke wenn das Spiel diagonal zum Spieler steht. Vor der Rotation hätte diese Auslenkung des Joysticks lediglich die Y-Achse beeinflusst.

AP2: Ausrichtung des Labyrinths Die Arbeiten wurden in Q01-2008 abgeschlossen.

Implementierung des Zentrieralgorithmus Die verwendeten Servomotoren (Dynamixel DX-117) können 1024 Positionen anfahren zwischen denen jeweils $0,293°$ liegen [1]. Dies entspricht 300°. Von diesen 300° werden nur ca. 16° für die X-Achse und 13° für die Y-Achse benötigt, um eine Fläche des Brio-Labyrinth in beide Richtungen voll auszulenken. Das entspricht ca. 55 Servopositionen. Die Ansteuerung der Servomotoren erfolgt über eine Funktion, der man die absolute Position (0-1023), die der Servomotor anfahren soll, als integerwert übergibt.

Manuelle Kalibrierung Zuerst wurden von Hand die minimalen und maximalen Werte der Servomotoren bestimmt. Dazu haben wir die Servomotoren in ihre Mittelstellung

gebracht und die zwei Ebenen danach Waagerecht ausgerichtet. Mit einem eigens entwickelten Testprogramm wurde die maximale Position in eine Richtung mit dem Servo angefahren. Den so ermittelten Motorstellwert haben wir als maximalen Ausschlag definiert. Das wurde für jeden Motor und für jede Richtung durchgeführt. So haben wir die 4 Positionen erhalten, die zu den 4 Maximalauslenkungen der Spielfläche korrespondieren. Aus diesen Werten haben wir die Maximalauslenkung der Servomotoren relativ zu ihrer Mittelstellung bestimmt. Aufgrund der Symmetrie des Brio-Labyrinthes ist die Rotation von der Mittelstellung zum linken bzw. rechten Maximalwert identisch. Diese Werte liegen bei ca. 50 Schritten des Servomotors. Die Servomotorposition für die mittlere Stellung berechnet sich als wie folgt:

$$S1_{mittel} = \frac{S1_{max} + S1_{min}}{2} \tag{.7}$$

$$S2_{mittel} = \frac{S2_{max} + S2_{min}}{2} \tag{.8}$$

Automatische Kalibrierung Im zweiten Ansatz wurde die Kalibrierung automatisiert. Aus diesem Grund wurde eine Zentrierroutine geschrieben, die die jeweiligen minimalen und maximalen Werte ermittelt. Diese Werte werden zur Ausrichtung der Spielfläche und zur Skalierung des Joysticks verwendet.
Das Brio-Labyrinth verfügt über vier Endschalter. Jeder dieser Schalter signalisiert das Erreichen der maximalen Auslenkung einer Achse in eine Richtung. Zu Beginn werden die beiden Servos in ihre Mittelposition (512, 512) gefahren. Im Folgenden wird der Servomotor mit Schritten von einer Position in um eine Achse gedreht, bis der entsprechende Endschalter ausgelöst wird. Nach dem Auslösen des Endschalters wird die Position des Servomotors zu dieser Auslenkung gespeichert. Dieses Verfahren wird analog für die andere Richtung angewendet. Die so ermittelten Werte repräsentieren die minimale und maximale Servomotorposition. Damit die Endschalter während des Spielens nicht auslösen wird der Aktionsraum des Servos auf jeder Seite um 5 Servomotorpositionen verringert. Die Mittelstellung ergibt sich aus dem arithmetischen Mittelwert der zwei ermittelten Positionen. Siehe Formeln 3.1 und 3.2. Diese Methode wird analog für den zweiten Servo angewendet.
Mit den so ermittelten Werten wird die Spielfläche waagerecht ausgerichtet. Wie in Kapitel 2.2.2 beschrieben werden für die Skalierung des Joysticks die Mittelstellung und die maximale Auslenkung relativ zur Mittelstellung in eine Richtung benötigt. Diese Werte werden dem Joystick-Objekt von dem Zentrieralgorithmus bereitgestellt. Dann beginnt der eigentliche Spielvorgang.

Test Nach der Fertigstellung des Zentrieralgorithmus haben wir ihn einer Reihe von Tests unterzogen.

Funktion der Kalibrierung Der erste Test sollte zeigen, ob der Algorithmus prinzipiell funktioniert. Hierzu wurden die Gummiriemen, die die Servomotoren mit den Drehgriffen des Brio-Labyrinths verbinden entfernt und zunächst beide Servomotoren auf ihre Mittelstellung (512) gefahren. Danach haben wir die Ebene leicht außerhalb ihrer waagerechten Position ausgerichtet und die Gummiriemen wieder montiert. Im diesem Test näherten wir uns den vier Endschaltern mit einer Geschwindigkeit von einem Servomotorschritt an. Der Test ergab, dass diese Geschwindigkeit bei drei der vier Endschalter funktioniert. Bei dem vierten Endschalter trat das Problem auf, dass die Rückstellfeder des Schalters stärker als die Sicherheitsfeder [1] des Brio-Labyrinths ist. Dadurch dehnt sich die Sicherheitsfeder bis zu einem gewissen Grad und erst dann wird der der Endschalter ausgelöst. Das hat zur Folge, dass sich der Servomotor in Richtung dieses Endschalters weiter dreht als er sollte. Dadurch wird ein falscher Wert zur Berechnung der Mittelstellung genutzt. Nach der Ausrichtung ist die Ebene in Richtung dieses Endschalters geneigt.
Um dieses Problem zu beheben haben wir die Geschwindigkeit systematisch der Servomotoren erhöht. Es stellte sich heraus, dass ab einer Geschwindigkeit von 5 Servomotorpositionen pro Schritt der Endschalter zum richtigen Zeitpunkt auslöst.
Das Erhöhen der Anfahrgeschwindigkeit ergab wiederum einen Nachteil: Die Auflösung der ermittelten Position für den vierten Schalter war nun geringer und konnte im schlimmsten Fall einen Fehler von 5 Servomotorschritten aufweisen, was einem Offset von 1,6° entspricht.
Dieses Problem wurde durch einen Austausch der aktuellen Feder durch einen O-Ring gelöst. Damit war gegeben, dass der Endschalter zuerst auslöst, bevor sich die Feder bzw. der O-Ring am dem Faden spannt.
Jetzt konnten wieder alle Endschalter mit einer Geschwindigkeit von einem Servomotorschritt angefahren werden. Damit stehen für die Berechnung der Mittelposition Werte in der höchsten zu erreichenden Auflösung zur Verfügung.
Nachdem der Algorithmus optimiert wurde haben wir untersucht, ob der Algorithmus deterministisch ist. Hierzu haben wir die Ebene zentriert und danach den Algorithmus 20 mal ausgeführt. Nach den 20 Durchläufen war die Ebene noch immer zentriert. Der Algorithmus ist also robust genug, um auch eine bereits zentrierte Ebene zuverlässig waagerecht auszurichten.
Als Letztes haben wir gestestet wie sich der Algorithmus bei verschiedenen Ausgangslagen der Ebene verhält. Hierzu haben wir die Servomotoren in ihre Mittelstellungen (512) gefahren und danach die Ebene willkürlich ausgerichtet. Bei jeder Startkonfiguration war der Algorithmus in der Lage die Labyrinthebene waagerecht auszurichten.

[1] Die Ebenen des Brio-Labyrinths sind durch Fäden mit den Achsen verbunden. Diese Fäden übertragen die Rotation der Achsen auf die Fläche. Um ein Überdrehen des Spiels zu vermeiden, bzw. ihm entgegen zu wirken ist je eine Feder pro Faden installiert. Wird die Spielfläche weiter als eine Maximalposition ausgelenkt verhindert dei Feder das Reissen des Fadens indem sie das Überdrehen durch Dehnung kompensiert.

Lagesensoren In einer theoretischen und praktischen Untersuchung sollte ermittelt werden, ob der Einsatz eines Lagesensors zum Ausrichten der Spielfläche erforderlich ist. Mit den Daten eines Lagesensors kann man die Spielfläche absolut waagerecht im Raum ausrichten. Dafür muss der Sensor unter der Spielfläche angebracht werden und in das Gesamtsystem integriert werden. Mit dem Sensor ist es möglich auch während eines Spielvorgangs die Mittelstellung der Spielfläche in der Waagerechten zu halten. Damit ist der Schlupf, der an den Gummiriemen während einer Bewegung auftritt und zu bleibenden Fehlern in der Mittelstellung führt, kompensiert.
Technisch kann das Problem mittels eines geschlossenen Regelkreises verdeutlicht werden. Ein Regelkreis beschreibt den Zusammenhang zwischen Sollwert, Istwert, Störungen und Steuersignal. Der Sollwert wird vorgegeben. Der Istwert soll möglichst schnell den Sollwert annehmen. Dabei gilt es, alle Störungen, die von Au"sen auf den Istwert einwirken, zu kompensieren. Die Differenz zwischen Soll- und Istwert wird durch die Rückkopplung eines gemessenen Istwertes an den Anfang des Regelkreises berechnet. Der Istwert wird dabei vom Sollwert abgezogen und ergibt so einen kleineren Sollwert als Führungsgröße.
Wird so eine Regelung technisch realisiert, übernehmen Proportional-, Differential- und Integral-Glieder die Anpassung des Istwertes an den Sollwert. Diese Glieder verändern das an sie angelegte Steuersignal und reichen es an das zu steuernde Element weiter. Ein Proportionalglied verstärkt sein Eingangssignal um einen konstanten Faktor. Das Differentialglied bildet die Ableitung vom bestehenden Fehler zwischen Soll- und Istwert. Im Integralglied wird der bestehende Fehler integriert und als Ausgangssignal weitergegeben.
In dem Fall, dass ein Mensch das Brio-Labyrinth bedient, stellt sich der Signalfluss wie folgt dar: Den Sollwert für eine Position gibt der Spieler vor. Auf den Istwert der Kugelposition wirken während der Umsetzung viele Störfaktoren ein. Über die visuelle Wahrnehmung der Position bekommt der Spieler eine Rückmeldung über seine Aktion und passt den Sollwert dementsprechend an.

Lässt der Spieler den Joystick los, stellt er sich mit Federn automatisch in die Mittelposition. Im Idealfall ist das Spielfeld nun auch exakt waagerecht. Durch die bleibenden Störeinflüsse ist dies aber nicht gegeben. Durch den Einsatz des Lagesensors kompensiert man viele Störgrößen, da die Lage des Spielbretts als Referenz verwendet wird. Somit kann man eine waagerechte Ausrichtung der Spielfläche in der Ruheposition des Joysticks erreichen.

Wird das Spiel von einem Menschen gespielt, abstrahiert er den komplexen Zusammenhang der oben beschriebenen Regelstrecke und denkt, er steuert die Kugel. Somit wird das Regelproblem auf die Zwei Komponenten Joystick und Ball runtergebrochen. Der Rest der Regelung wird unbewusst vom Spieler ausgeführt.

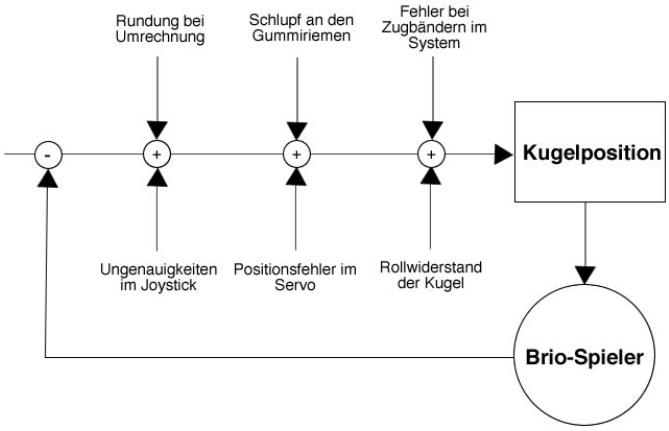


Abbildung .2: Regelkreis mit Einfluss des Spielers

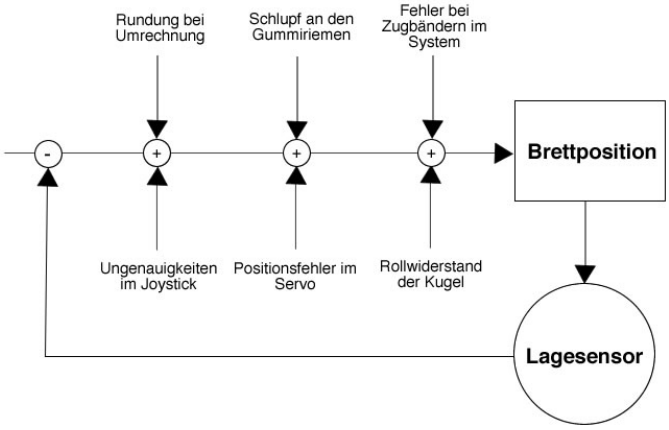


Abbildung .3: Ausrichtung des Spielbretts in der Ruheposition

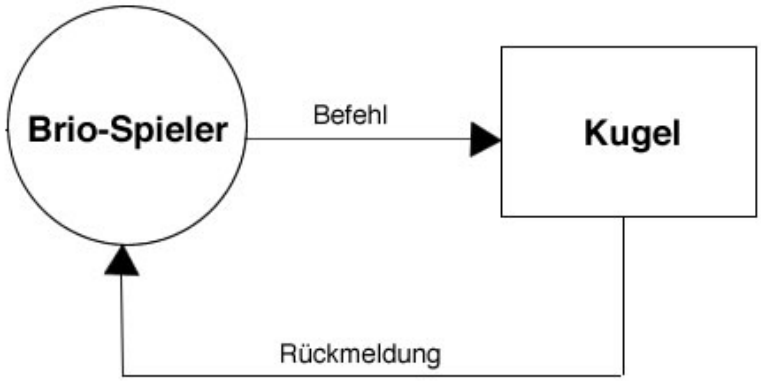


Abbildung .4: Abstrahierung des Problem durch aktive Kompensation der Störgrößen durch den Spieler

Zusammenfassung: Im Vorfeld konnte nicht abgeschätzt werden, wie gut ein Zentrieralgorithmus basierend auf den Endschalterwerten funktioniert. Die Praxis hat gezeigt, das keine absolute Genauigkeit erforderlich ist. Das ist darauf zurückzuführen, dass der Mensch bei der Steuerung nur relative Bewegungen macht und sich weder an der Nullstellung des Joysticks orientiert noch diese als Ruheposition nutzt. Im Briospiel werden Ruhepositionen erreicht, indem der Ball in eine Ecke bewegt wird und durch einen Vollausschlag der Spielfläche in die entsprechenden Richtungen dort gehalten wird. Aufgrund dieser Erkenntnisse ist es zu diesem Zeitpunkt nicht nötig, einen Lagesensor in das Brio System zu integrieren.
Wird in der Zukunft das Spiel von einem Agenten bzw. einem Lernverfahren gesteuert, kann der Einbau eines solchen Sensor erforderlich sein. Hat der Agent zuvor in der Simulation den Umgang mit einem absolut waagerecht ausgerichteten Spielbrett gelernt, wird es ihm auf dem realen System leichter fallen, wenn er die waagerechte Ausrichtung auch hier vorfindet. Abhängig vom Verfahren, mit dem der Agent in der Simulation gelernt hat, steuert er das Spiel dann u.U. nicht durch relative Änderungen, sondern durch absolute Steuerbefehle, die als Referenz die Mittelposition des Spielbretts haben.

AP3: Ballmagazin Das Brio Labyrinth wird für EEG/fMRI-Versuche mit Probanden und als Testbed für Lernalgorithmen verwendet. Bei fMRI-Versuche befindet sich der Proband in einer liegenden Position und kann die Spielfläche nur über einen Spiegel betrachten. Daher ist es dem Probanden nur sehr schwer möglich einen verlorenen Ball auf die Startposition zurückzulegen, da er sich in einer ungewohnten Lage befindet und sich nicht ausreichend bewegen kann. Aufgrund dessen sollte ein Ballmagazin mit folgenden Spezifikationen entwickelt werden:

- Kapazität für mindestens 10 Bälle.
- Verwendung von Bällen mit einem Durchmesser von 12mm bis 12,7mm.

- Manuelles Bereitstellen eines Balles.
- Automatisches Bereitstellen eines Balles.
- Konstruktion aus unmagnetischen Materialien.
- Kein Verdecken der Spielfläche.
- Einfaches An- und Abmontieren des Magazins.

Im Verlauf des Projektes wurden 3 Konzepte entwickelt. Im Folgenden wird jedes Konzept skizziert und erklärt. Ziel ist es, ein machbares Konzept zu erhalten, welches allen Anforderungen entspricht.

Konzept 1 Das erste Konzept für das Ballmagazin bestand aus einer senkrecht stehenden Röhre in der bis zu zehn Kugeln übereinander liegen. Mittels eines Kippmechanismus sollte es möglich sein, die unterste Kugel über eine Rinne auf das Spielfeld zu befördern. Alle anderen Kugeln werden blockiert, so dass jeweils eine Kugel herausrollen kann. Wird die Rinne wieder an die Röhre geklappt fallen die verbleibenden Kugeln nach.
Diese Konzept wurde verworfen, da es schwierig geworden wäre, eine Kugel manuell

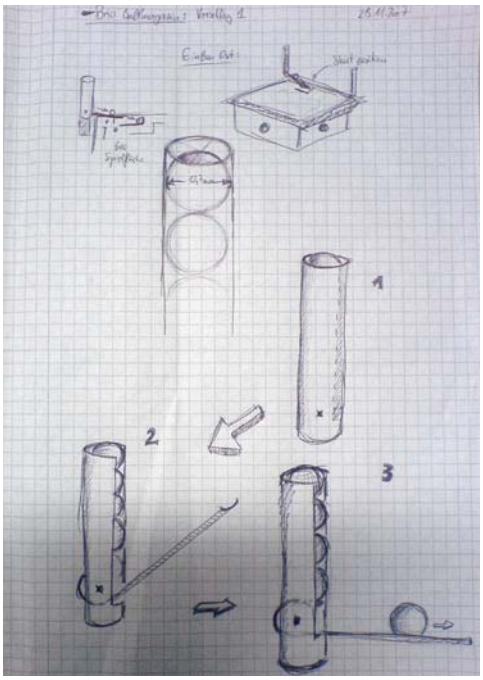


Abbildung .5: Erstes Konzept

auszulösen. Die notwendige Bewegung wäre nur sehr schwer mit einem Druckknopf realisierbar. Über einen Druckknopf hätte eine lineare Bewegung von 1-2cm erzeugt werden können. Für eine Bewegung der Rinne um mehr als 90° wäre eine zu große Übersetzung der Knopfbewegung nötig gewesen. Dadurch wäre der Auslösemachanismus schwer zu

bedienen bzw. mit einem hohen Kraftaufwand verbunden und sehr anfällig gegenüber Abnutzung geworden.

Konzept 2 Im zweiten Konzept wurde das Prinzip der senkrecht stehenden Röhre zur Kugelaufbewahrung aufgegriffen. Der Auslösemachanismus wird durch eine schwenkbare Platte, an der eine weitere Röhre befestigt ist, realisiert. Die Platte kann um 90° geschwenkt werden und so zwei Positionen einnehmen. In der ersten Position fällt eine Kugel durch das Loch in der Platte in die untere Röhre, wird da aber noch blockiert. In der zweiten Position werden alle verbleibenden Kugeln von der Platte blockiert, die das Loch in der Platte durch die Drehung nicht mehr unter der ersten Röhre befindet. Die Kugle, die in der ersten Position in die untere Röhre gefallen ist, wird nun freigegeben und rollt auf die Spilefläche.

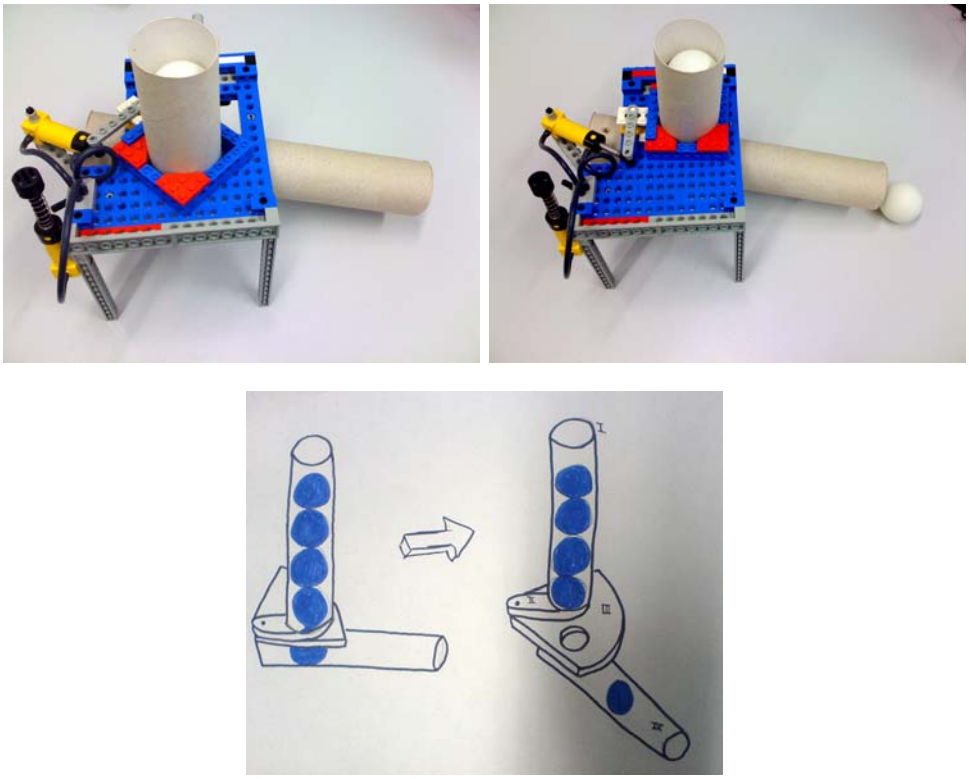


Abbildung .6: Testaufbau Zweites Konzept und Skizze

Das Konzept wurde in einem ersten Testaufbau realisiert. Der Testaufbau aus Lego-Technik© demonstrierte die prinzipielle Machbarkeit. Beim Anfertigen der ersten Zeichnungen in Solidworks stellte sich heraus, dass alle Teile und somit der gesamte Aufbau sehr filigran und instabil geworden wäre. Aus diesem Grund wurde auch dieses Konzept verworfen.

Konzept 3 Im dritten Konzept haben wir die Röhre nicht senkrecht gestellt, sondern nur leicht aus der Waagerechten bewegt. Damit können die Kugeln langsam aus der Röhre rollen.
Der Auslösemechanismus basiert auf dem Prinzip einer Wippe. Stifte, die auf jeder Seite der Wippe angebracht sind, können durch zwei Löcher in das Innere der Röhre eindringen und so die Kugeln blockieren. In der ersten Position der Wippe werden alle Kugeln blockiert. In der zweiten Position wird die vorderste Kugel freigegeben und alle verbleibenden blockiert.

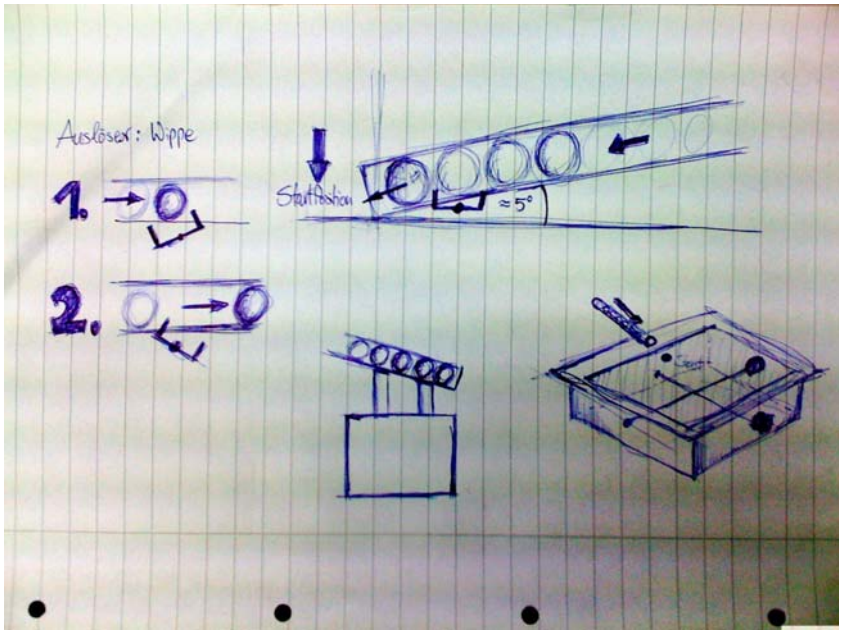


Abbildung .7: Drittes Konzept

In einem Testaufbau haben wir die Machbarkeit des Konzepts überprüft. Dabei befestigten wir eine Röhre direkt neben der Startposition des Brio-Labyrinths. Die Öffnung der Röhre war so ausgerichtet, dass eine Kugel aus der Röhre direkt auf die Startposition rollen kann. Da das Magazin die Spielfläche nicht verdecken darf endete die Röhre direkt über dem Rand der Spielfläche. Der Test soll zeigen, dass die Kugel zuverlässig auf der Startposition landet. Nach dem verschiede Steigungen gestestet wurden, kamen wir zu dem Ergebnis, dass bei einer Steigung von 5° die Geschwindigkeit der Kugel ausreicht, um auf der Starposition zu landen. Bei höheren Steigungen war die Kugel so schnell, dass sie über Barrieren auf dem Spielfeld sprang und direkt in eines der Löcher fiel.
Dieses Konzept haben wir ausgearbeitet, da der Mechanismus zum Auslösen einer Kugel einfach zu realisieren war. Weiterhin ergaben die ersten Konstruktionszeichnungen, dass das System stabil und robust werden würde.

Konstruktion und Bau des Magazins Die Konstruktion und Verifikation der Funktion des Ballmagazins wurde mit dem Softwaretool Solidworks 2008 durchgeführt. Bevor

die Konstruktion beginnen konnte, haben wir das Brio-Labyrinth vermessen um so die ersten Maße für die Struktur des Magazins zu bekommen.

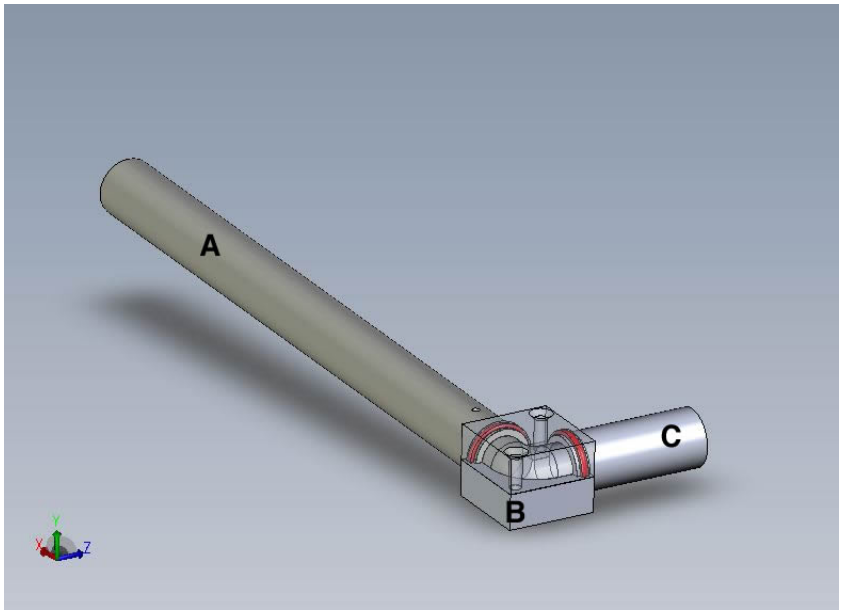


Abbildung .8: Röhren für Kugelreservoir (A), Umlenkung (B) und Auslassröhre (C).

Kugel Reservoir Im Folgenden wird das Kugel-Reservoir, dargestellt in Abbildung .8, erläutert:

Teil A: Die Spielkugeln haben einen Durchmesser zwischen 12mm (Keramikkugeln) und 12,7mm (Stahlkugeln). Mit einer ausreichenden Toleranz wurde der Innendurchmesser der Röhre mit 14mm bestimmt. Für eine ausreichende Steifigkeit und um noch genug Material zur Befestigung der Röhre zu haben wählten wir den Außendurchmesser mit 18mm. In der Spezifikation wurde festgelegt, dass mindestens 10 Kugeln in der Röhre Platz finden sollen. Ausgegegangen von den Stahlkugeln ergibt sich daraus eine Länge von mindestens 127mm. Zusätzlich werden noch noch 21mm Platz für den Auslösemachnismus benötigt. Die am Ende konstruierte Länge von 191mm haben wir gewählt, um auch bei konstruktiven Änderungen noch genug Platz für 10 Kugeln zu haben.
Teil C: Eine kürzere Röhre mit dem gleichen Außen- und Innendurchmesser steht im 90° Winkel zum Kugel Reservoir. Durch diese Röhre rollt die Kugel über den Rand des Brio-Labyrinths zur Startposition.
Teil B: Die Verbindung zwischen den Röhren wird von einer um 90° abgewinkelten Röhre hergestellt. Diese Röhre besteht aus einer Ober- und einer Unterhälfte die durch eine Schraubverbindung zusammengehalten werden. Auf der Innenseite jeder Hälfte des Verbindungsstücks ist eine halbrunde Vertiefung auf einer Viertelkreisbahn eingefräst. Zusammengesetzt ergeben die beiden Hälften also ein um 90° abgewinkeltes Röhrenstück

welches denselben Innendurchmesser wie die beiden anderen Röhren hat. Die Röhre des Reservoirs und die Auslassröhre dringen jeweils 4mm in das Verbindungsstück ein. Die O-Ringe der zu verbindenden Röhren fassen dabei in die Nuten des Verbindungsstücks und arretieren so die Röhren im Verbindungsstück.

Auslösemechanismus Der Auslösemechanismus muss garantieren, dass nur eine Kugel zur Zeit das Ballmagazin verlässt. Wir haben uns für einen Wippmechanismus entschieden. Die Wippe ist in einer Aufnahme unterhalb der Reservoirröhre angebracht. Auf der Wippe befinden sich im Abstand von 14mm zwei Pins mit einer Höhe von 5 mm. Diese Pins können durch Löcher in der Reservoirröhre hereinragen. Der Abstand zwischen Wippe und Reservoirröhre ist so bestimmt, dass beim Anschlagen der Wippe auf einer Seite der Röhre je ein Pin komplett in die Röhre ragt und der andere die Kugeln in der Röhre nicht behindert. Der gesamte Mechanismus wird über Schubstangen und einen Umlenkungswinkel bedient. Somit ist es möglich, durch eine lineare Bewegung an der Schubstange in eine rotatorische an der Wippe umzusetzen (Siehe Abbildung .9).

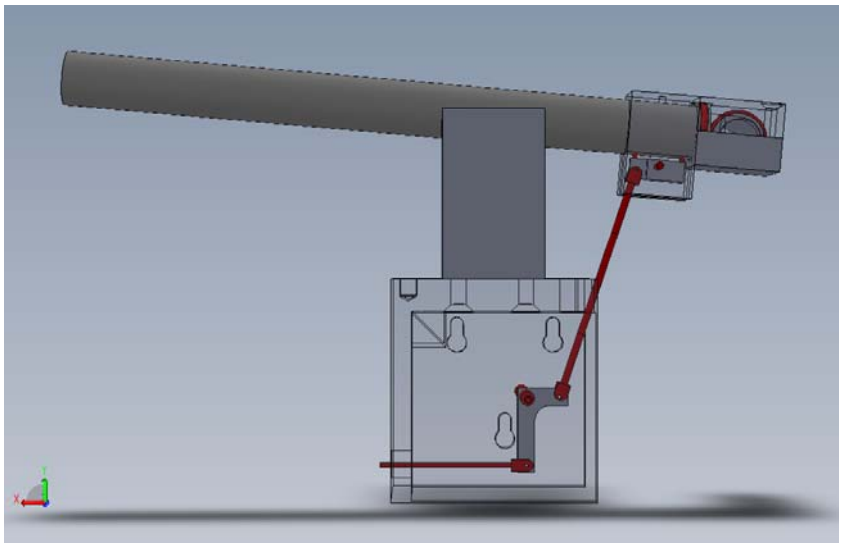


Abbildung .9: Auslösemachnismus über Wippe und Stifte

Korpus Die Grundform für den Korpus ist ein an der Vorder- und Unterseite geöffneter Kasten. Auf der Oberseite wird die Aufnahme für das Kugelreservoir angeschraubt. Weiterhin stellt der Korpus eine feste Verbindung zwischen dem Magazin und dem Spiel her. Die Verbindung zwischen Spiel und Magazin wird über 3 Langlöcher mit einem Durchmesser von 4mm hergestellt. Über eine 6mm große Bohrung an der Unterseite jedes Langlochs wird der Kopf einer M4 Schraube durch die Wand des Korpuss gesteckt. Anschliessend kann der Korpus 6mm abgesenkt werden und klemmt zwischen Schraubenkopf und der Au"senwand des Brio-Labyrinths.
Zwei weitere Bohrungen ermöglichen die Aufnahme einer Welle. Diese Welle dient zum Umlenken der Bewegung der Schubstangen des Auslösemechanismusses im Inneren des

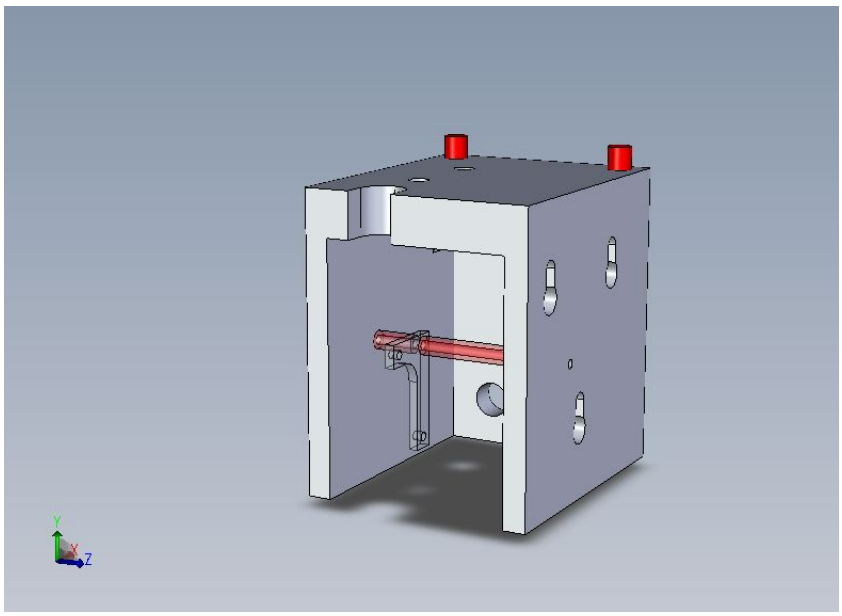


Abbildung .10: Korpus

Korpusses.
Ein weiteres Loch dient zur Aufnahme eines Gleitlagers, in dem die durch die Umlenkung entstehenden radialen Bewegungen der Schubstange ausgeglichen werden. (Siehe Abbildung .10)

Module zum Auslösen Als Aktuator für den Auslösemachnismus soll ein mechanischer und ein elektronischer Auslöser sorgen. Es werden zwei Auslösemodule gebaut. Von den äußeren Maßen sind die beiden Module identisch. Durch einfaches Aufstecken auf zwei Stifte auf der Rückseite des Magazins stellen sie eine Verbindung zum Korpus des Magazins her. Jedes Modul verfügt an der dem Brio-Labyrinth zugewandten Seite über genau die gleichen Langlöcher wie der Korpus des Magazins. In diese Langlöcher werden genau wie bei dem Korpus M4 Schrauben gesteckt um eine feste Verbindung zwischen dem Modul und dem Brio-Labyrinth herzustellen.

Mechanischer Auslöser Im Mechanischen Auslöser (Siehe Abbildung .11) wird eine Schubstange durch einen Knopf, der durch die Aussenwand des Modules ragt, in Bewegung gesetzt. Der maximale Weg der Stange wird durch zwei Anschläge auf 7,5mm begrenzt. Zusätzlich sorgt eine Druckfeder dafür, dass der Knopf immer in seine Ausgangsposition zurückkehrt. An der dem Knopf gegenüberliegenden Seite ist eine Scheibe mit einem Durchmesser von 10mm angebracht. Über diese Scheibe wird die Kraft und Bewegung auf die Schubstange des Magazins übertragen.

Elektronischer Auslöser Um eine einfaches An- und Abbauen des elektronischen Auslösers (Siehe Abbildung .12) zu gewährleisten befindet sich im elektronischen Auslöser

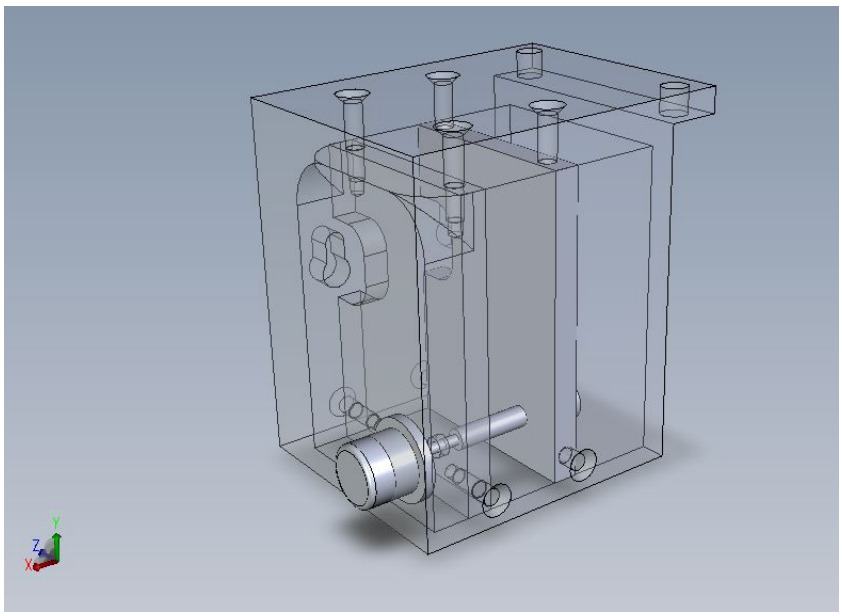


Abbildung .11: Mechanischer Auslöser

ein Servo des Typs Dynamixel DX-117. Servos diesen Typs werden bereits für die Ansteuerung der Spielflfäche im Brio-Labyrinth verwendet. Die Ansteuerung der Servos erfolgt über ein Bussystem, welches einfach erweiterbar ist. Hierbei ist nur ein Servo mit der seriellen Schnittstelle verbunden. Jeder weitere Servo wird mit seinem Vorgänger verbunden, also ist die Art der Verschaltung eine "Daisy-Chain". Damit kann nun der Servo einfach an den zweiten Servo des Spiels angeschlossen und verwendet werden.
Auf der Nabe des Servos ist ein Flansch montiert. Auf diesem Flansch ist eine Platte verschraubt. Die Platte ist so angeordnet, dass durch Rotation des Servos die Schubstange des Magazins bewegt werden kann. Ein Anschlag für die Platte sorgt dafür, dass durch unsachgemäße Bedienung des Servos das Magazin nicht beschädigt werden kann.

Analyse der Daten des Piezosensors zur Einschlagserkennung einer Kugel Über eine Karte von National Instruments sind an den Computer analoge Sensoren angeschlossen. Die Frequenz, mit der neue Werte der Sensoren verfügbar sind liegt bei ca. 600 Hz. Die Rohwerte des Sensors sind digitalisiert und liegen zwischen Null und Fünf.

Einschlagsdetektion durch Zeitversetzte-Schwellwertüberprüfung In Abbildung .13 sind exemplarisch typische Signalverläufe dargestellt, die experimentell ermittelt wurden. Darauf aufbauend exisitert in dem Brio-Softwareframework bereits ein Verfahren zur Kugel-Einschlagserkennung. Dieses Verfahren basiert auf einfachen Schwellwerten. Der ungestörte Sensorwert wird zunächst auf die Mitte seines Wertebereichs abgebildet (Formel .9). Falls der aktuelle Wert dadurch ins negative gerutscht ist wird er um fünf

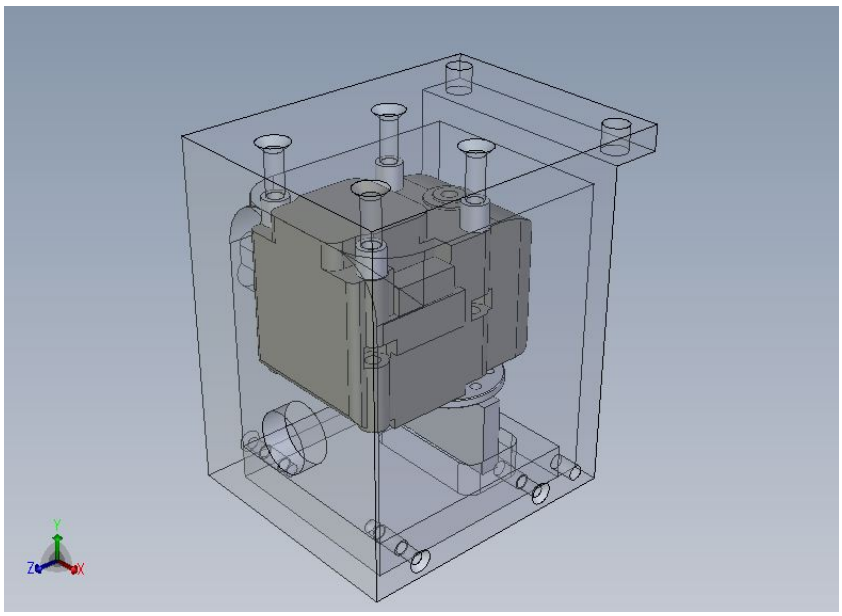


Abbildung .12: Elektronischer Auslöser

erhöht (Formel .10). Danach wird überprüft, ob der Sensorwert zwischen der oberen $(3,25)$ und unteren $(1,75)$ Grenze liegt (Formel .11). Ist dies nicht der Fall, wird ein Kugeleinschlag detektiert. Zusätzlich ist in dem Verfahren noch Zeitbegrenzung eingebaut. In den drei Sekunden nach einem detektierten Kugeleinschlag liefert die Funktion keinen Wert zurück um Mehrfacherkennungen des selben Einschlags zu vermeiden.

$$piezoValue_1 = rohWert - 2,45 \tag{.9}$$

$$piezoValue_2 = \begin{cases} piezoValue_1 + 5 & wenn \quad piezoValue_1 < 0 \\ piezoValue_1 \end{cases} \tag{.10}$$

$$Kugeleinschlag \quad wenn \quad \begin{cases} piezoValue_2 < 1,75 \\ piezoValue_2 > 3,25 \end{cases} \tag{.11}$$

Die Bilder in Abbildung .13 stammen aus einer Messreihe zur Ermittlung von charakteristischen Sensorantworten. Auf den Plots bezeichnet die Y-Achse Volt[1] und die X-Achse die Zeit, unterteilt in Samples. Dabei wurde für jeden Event zwei Sekunden lang bei 600 Hz also 1200 Werte in einem Logfile gespeichert. Die in dieser Testreihe aufgezeichneten Aktionen waren: Das Durchfallen einer Kugel (für jedes Loch), das Anschlagen der Ebene an allen Anschlägen und Messungen während eines Spielvorgangs. Diese Messwerte wurden dann, wie im weiterem erläutert, ausgewertet.
Ziel der Untersuchung sollte sein, ein Einschlag einer Kugel gegenüber dem Anschlagen

[1]Bei dem Piezosensor handelt es sich um einen analogen Sensor. Die über dem Sensor abfallende Spannung wird gemessen und in einer begrenzten Auflösung über einen AD-Wandler (Siehe Seite ??) digitalisiert. Daher ist die Einheit des Signals Volt.

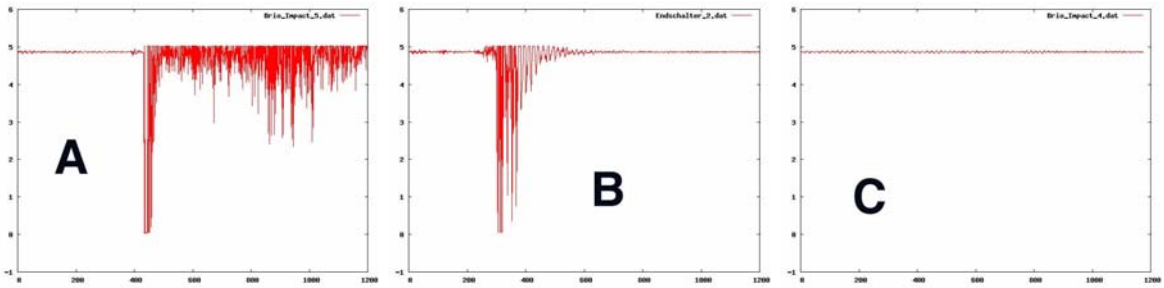


Abbildung .13: Charakteristische Signalverläufe des Piezosensors. **A:** Aufschlag einer Kugel. **B:** Auslösen des Endschalters. **C:** Ruhesignal

eines Endschalters zu unterscheiden. Weiterhin sollte die Einschlagserkennung verbessert werden.

Erster Ansatz: Analyse der Power des Signals Es gilt zunächst mal einige Annahmen zu den Signalen zu treffen. Schlägt ein Ball auf der Bodenplatte des Spiels auf, erzeugt der Aufschlag für einen kutzen Moment einen sehr starken Ausschlag auf dem Piezo-Sensor. Durch die Schräge der Bodenplatte rollte die Kugel nach einem Aufschlag über einen Teil der Bodenplatte und erzeugt dabei über einen längeren Zeitraum einen leichten Ausschlag des Signals des Piezosensor. Diese Charakteristiken des gesuchten Signals kann man sich zu Nutze machen. Dazu wurden die aufgenommenen Werte in Fenster eingeteilt. Diese Fenster haben eine Länge von 50 Werten.

Detektieren von Einschlägen:

Den Abstand zwischen zwei Datenpunkten erhält man durch Quadrierung der Differenz. Aufsummiert über eine Fensterlänge von 50 Werten erhält man somit die Summe der Abstände zwischen allen aufeinander folgenden Datenpunkten. Geteilt durch die Länge des Fensters ergibt sich ein durchschnittlicher Abstand.

$$
\begin{array}{llcl}
\textit{Piezomesswerte} & p_{t-1}, p_t & & \\
\textit{Schwellwert} & S & = & 1,25 \\
 & dist & = & (p_{t-1} - p_t)^2 \\
 & Summe_{window} & = & \sum_{I=0}^{50} dist_i \\
 & Summe_{avg} & = & \frac{Summe_{window}}{50} \\
\textit{Kugeleinschlag} & wenn & Summe_{avg} & > \quad S
\end{array}
\tag{.12}
$$

Wird der Wert $Summe_{avg}$ größer als der Schwellwert S wird in der Klasse BrioState der Zeitwert des letzten Ballverlustes auf die aktuelle Zeit gesetzt. Dieses Verfahren erkennt starke Einschläge der Bälle auf dem Boden des Brio-Labyrinths.

Detektieren von Rollen über den Boden:

Ein Ringbuffer mit der Länge von 10 Elementen nimmt die 10 letzten Werte von $Summe_{avg}$ auf. Eine zweite Überprüfung der Werte greift darauf zurück. In einer Schleife werden die Werte der n letzten Fenster zu einem Wert aufsummiert und Anschlie"send durch n geteilt. Überschreitet die Summer der n zurückliegenden Fenster einen zweiten Schwellwert wird ebenfalls ein Ballverlust detektiert.

$$\begin{aligned} Schwellwert \quad S &= 1,25 \\ Summe_{avg} &= \frac{Summe_{window}}{50} \\ Summe_{roll} &= \frac{\sum_{i=0}^{n} Summe_{avg}}{n} \\ Kugeleinschlag \quad wenn \quad Summe_{avg} &> \frac{S}{n} \end{aligned} \tag{.13}$$

Diese Methode eignet sich gut, um das Rollen einer Kugel über die Bodenplatte zu detektieren.
Um Überlagerungen der beiden Methoden zu vermeiden, sorgt ein Timer dafür, dass höchstens alle 5 Sekunden ein Ballverlust detektiert werden kann. Sonst wäre es möglich, dass Beim Einschlag einer, und beim Rollen der zweite Ballverlust detektiert werden würde.

Test und Vergleich der Verfahren: Der Test soll einen direkten Vergleich der beiden Verfahren ermöglichen. Daher haben wir einen praktisches Testverfahren ausgewählt. Die Güte der beiden Verfahren wurde am realen System getestet. Eine Keramikkugel wurde fünf mal in Folge durch jedes Loch fallen gelassen. Im besten Fall wurden dabei auch fünf Einschläge detektiert.
Der Test sollte folgende Ergebnisse liefern:

- Zuverlässigkeit des aktuellen Verfahrens.
- Vergleich der Performance des Window-basierten Verfahrens gegenüber dem aktuellen.
- Lokalisierung von Zonen auf dem Spielfeld, in denen die Einschlagsdetektion generell schwierig ist.

Im folgenden werden wir die erreichten Ergebnisse als True Positive (TP), False Negative (FN) und False Positive (FP) bezeichnen. True Positive bedeutet, dass ein Balleinschlag stattgefunden hat und auch detektiert wurde. False Negative ist das Ergebnis, wenn ein Einschlag stattgefunden hat, jedoch nicht erkannt wurde. False Positive bedeutet, dass ein Einschlag erkannt wurde, der nicht stattgefunden hat.

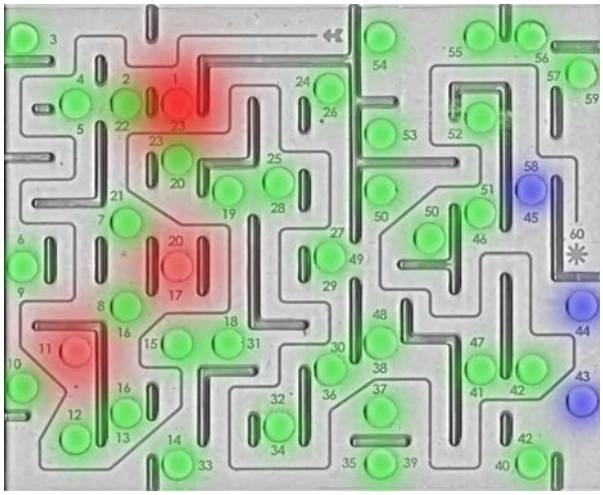
Zeitvers.-Schw.-Ü.(Seite 19):			**Windowbas. (Seite 21):**			
TruePos	FalsePos	FalsNeg	TruePos	FalsePos	FalsNeg	Lochnummer
3	2		5			23 bzw. 1
5			5			22 bzw. 2
5			5			3
5			5			4 bzw. 5
5			5			6 bzw. 9
5			5			21 bzw. 7
5			5			8 bzw. 16
5			5			10
4	1		4		1	11
5			4		1	12
5			5			13
5			5			14 bzw. 33
5			5			15
5			5			17
5			5			18 bzw. 31
4	1		5			19
4	1		5			20
5			5			24 bzw. 26
5			5			25 bzw. 28
5			5			27 bzw. 29 bzw. 49
5			5			30 bzw. 36
5			5			32 bzw. 34
5			5			35 bzw. 39
5			5			38 bzw. 48
5			5			37
5			5			40
5			5			41 bzw. 47
5			4		1	42
4		1	5			43
4		1	5			44

Tabelle fortgesetzt:

Zeitvers.-Schw.-Ü.(Seite 19):			**Windowbas. (Seite 21):**			
TruePos	FalsePos	FalsNeg	TruePos	FalsePos	FalsNeg	Lochnummer
4		1	5			45 bzw. 58
5			5			51 bzw. 46
5			5			50_1
5			5			50_2
5			5			52
5			5			53
5			5			54
5			5			55
5			5			56
5			5			59 bzw. 57

Test des Bestehenden Verfahrens:

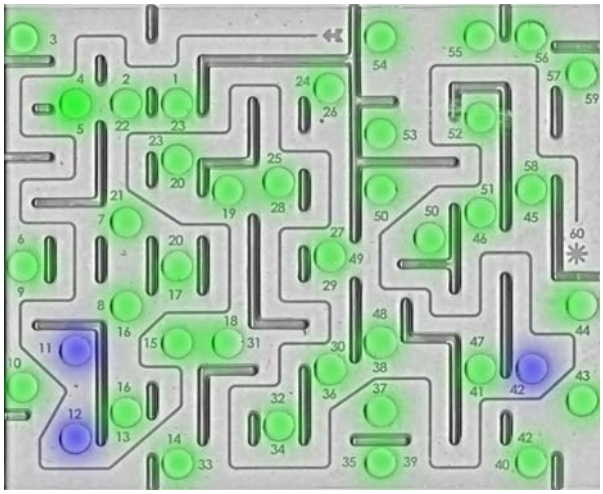
Dieses Verfahren ist recht zuverlässig bei der Einschlagsdetektion. Jedoch ist es genenüber Vibrationen im Spiel zu anfällig. Ein leichter Stoß auf den Tisch oder Spiel reicht aus um eine False Positive zu erzeugen. Die False Negatives am rechten Rand des Spiel erklären sich über den Kurzen Weg der Kugel zum Ausgang.



Performance des bestehenden Verfahrens.
Legende: ●TP, ●FN, ●FP

Test des optimierten Verfahrens:

Dieses Verfahren zeigt eine größere Robustheit gegenüber Vibrationen. Ein Nachteil ist allerdings, dass dadurch False Negatives entstehen. Insgesamt schneidet das Verfahren besser als das bestehende ab und bietet darüber hinaus noch mehr Einstellmöglichkeiten.



Performance des optimierten Verfahrens.
Legende: ●TP, ●FN, ●FP

Testergebnisse

- Die Zuverlässigkeit des aktuellen Verfahrens ist ausreichend, kann aber noch verbessert werden. 193 von 200 Einschlägen wurden korrekt erkannt. Dreimal wurde ein Einschlag nicht erkannt. Viermal erkannte das Verfahren Einschläge wo keine stattgefunden hatten.

- Das windowbzw.basierte Verfahren schneidet insgesamt besser ab. 197 von 200 Einschlägen wurden korrekt erkannt. Drei Einschläge wurden nicht erkannt (False Negative). Weiterhin ist es nicht so anfällig gegenüber Störungen des Signals des Piezosensors.

- Eine Region, in der die Detektion von Einschlägen schwierig ist, liegt genau vor dem Kugelausgang in der rechten unteren Ecke des Spielbretts. Das kann man dadurch begründen, dass der Weg der Kugel über die Bodenplatte nur sehr kurz ist und die Schwingfähigkeit der Platte bei einem Einschlag nah am Rand sehr gering ist.

Absolut korrekt erkennt keines der vorgestellten Verfahren einen Kugeleinschlag. Zusammenfassend kann man sagen, dass es am Ende darauf ankommt, welche Art des Fehler in der Anwendung schlimmer ist. Lässt man FalsePositives zu, werden vom Ballmagazin mehrere Kugeln ins Spiel gebracht. Bei FalseNegatives wird nach einem Kugelverlust keine Kugel auf der Startposition positioniert.
Das Verhalten der Algorithmen kann man über die Schwellwerte sowie die Zeit, die vergehen muss bevor ein neuer Einschlag erkannt werden kann, einstellen. Je nach Umgebung kann dann die Einschlagserkennung entsprechend eingestellt werden.

zweiter Ansatz: Frequenzanalyse Im zweiten Ansatz haben wir die diskrete Fouriertransformation (DFT) benutzt. Bei der DFT handelt es sich um eine Transformation, die ein diskretes Signal nach folgender Formel in sein Frequenzspektrum transformiert:

$$a_k = \sum_{j=0}^{Nbzw.1} \omega^{j*k} * a_j \quad \text{für} \quad k = 0, \ldots, Nbzw.1$$

Ziel bei dieser Untersuchung war es Frequenzen zu finden, die beim Durchfallen der Kugel durch ein Loch, eine starke Intensität haben und bei allen anderen Events eher klein sind, sich also signifikant von diesen Unterscheiden. Da die Analyse aller Werte sehr aufwendig ist, haben wir ein C++ Programm geschrieben, welches die Messreihe analysiert. Die Analyse umfasste de folgende Schritte:

1. Alle Messwerte fouriertransformieren.
2. Nach Frequenzintensitäten suchen, die für jedes Loch größer sind als bei den zwei anderen Events.
3. Gefundene Frequenzen mit dem Intensitätsunterschied ausgeben.

Diese Analyse hatte zu Ergebnis, dass bei einer Frequenz von 241 Hz eine Mindestdifferenz von 0,11 zwischen der Intensität der Frequenz beim Durchfallen einer Kugel und den zwei anderen aufgenommenen Events besteht. Dies war auch die einzige Frequenz, bei der die Intensität für alle Testdaten, in denen ein Kugeleinschlag stattgefunden hat, größer war als die Intensitäten der anderen Events. Das Signal des Piezosensors liegt im Bereich zwischen 0 und 5, eine Differenz von 0,11 entspricht also 2,2% des gesamten Bereichs.
Nach der Analyse war nicht klar, ob dieser geringe Unterschied zur Einschlagsdetektion bei manchen Löchern ausreichen würde.
Im Folgenden wurde die BRIO-Softwareumgebung um dieses Verfahren erweitert. In einem Ringbuffer wurden die letzten 600 Werte des Piezosensors gespeichert. Bei einem Peak in den Werten des Piezosensors wurden 600 weitere Werte Aufgezeichnet und ebenfalls gespeichert. Auf diese 1200 Werte wurde dann die DFT angewandt und die Frequenzintensitäten im Bereich von 235 bzw. 245 Hz ausgeben. Während der Test dieses Verfahrens stellte sich heraus, das die Intensitäten einer Frequenz bei gleichen Events sehr stark schwankten, und somit nicht aussagekräftig sind. Dies kann darauf zurückgeführt werden, dass die Fallhöhe, der Aufschlagwinkel und die Eigenrotation der Kugel je nach Neigung der Spielfläche variiert. Es ist somit nicht möglich das Durchfallen einer Kugel mit diesem Ansatz zu detektieren.

Analyse des Ergebnisses: Mit dem ersten Ansatz ist es möglich das Durchfallen einer Kugel zu detektieren. Im Gegensatz zu dem vorhanden Verfahren ist diese Methode sehr viel robuster gegen das Anschlagen der Spielfläche an eine der vier Endpositionen. Um das Anschlagen der Spielfläche als Kugelverlust zu detektieren, muss die Spielfläche sehr stark auf den Endschalter aufschlagen. Bei dem vorhandenen Verfahren hingegen reicht oftmals ein leichtes Berühren des Endschalters aus um den fälschlichen Verlust einer

Kugel zu detektieren. Dafür weist die von uns entwickelte Methode bei drei Löchern ein false negative Verhalten auf. Der Verlust einer Kugel wird also nicht detektiert. Diese Löcher befinden sich alle nahe an zwei Spielrändern. Bei diesen Löchern reicht das Herunterfallen nicht aus um den Einschlag direkt zu detektieren. Zwei der Löcher liegen sehr nah bei dem Auslass der Kugeln und somit reicht der kurze Zeitraum in dem die Kugel aus dem Spiel rollt nicht aus um das Rollen zu detektieren. Das dritte Loch liegt auf der linken Seite des Spiels. Unsere Vermutung ist, dass die Kugel sehr schnell an die untere Kante des Spiels rollt und dort zunächst gebremst wird. Von dieser Position fängt sie erneut an zu Rollen, da sie sehr langsam ist, reicht die Intensität des Piezosensorsignals nicht aus um als Rollen detektiert zu werden. Wie auf Seite 26 beschrieben, ist es mit dem zweiten Ansatz nicht möglich, das durchfallen einer Kugel eindeutig zu detektieren. Als Alternative oder Unterstützung des vorhandenen Verfahrens und unserem zweiten Ansatz haben wir folgende Vorschläge:

- **Lichtschranke:**
  Die Lichtschranke kann am Kugelauslass des Spiels montiert werden. Sie würde zuverlässig das Herausrollen einer Kugel detektieren. Wenn eins der zwei Verfahren einen Kugelverlust detektiert, könnte man sich den Zeitpunkt merken und nachdem die Lichtschranke ausgelöst wurde nachträglich zu dem geloggten Werten dazu schreiben. Wenn nach einer gewissen Zeit die Lichtschranke nicht ausgelöst wurde, könnte der Zeitpunkt verworfen werden da das Verfahren fälschlicher Weise das durchfallen einer Kugel detektiert hätte. Im anderen Fall, dass der Verlust einer Kugel nicht detektiert wurde, könnte man den Zeitpunkt des Auslösens der Lichtschranke in die geloggten Werte aufnehmen und vorher ein gewissen Offset abziehen.

- **Dehnungsmessstreifen (DMS):**
  Mit zwei DMS wäre es möglich die Verformung der Bodenplatte zu beobachten. Bei dem Aufschlagen einer Kugel sollte sich die Bodenplatte leicht in zwei Richtungen verformen. Diese Verformungen können detektiert und als Kugelverlust geloggt werden.

- **Härtere Bodenplatte:**
  Wenn die derzeitige Bodenplatte durch eine härtere ersetzt wird kann es sein, das sich die Werte des Piezosensors anders Verhalten und eins der zwei funktionierenden Verfahren das Durchfallen der Kugel zuverlässiger detektiert. Zur Zeit wird gerade ein zweites BRIObzw.Labyrinth umgebaut, in diesem wird eine Plexiglasplatte anstelle einer Holzplatte als Boden verbaut wird. Das neue Labyrinth wird mit einem Piezosensor desselben Typs ausgestattet. An diesem Labyrinth wird analysiert werden, ob die härtere Bodenplatte Einfluss auf die Messwerte des Piezosensors hat und sich die Verfahren dadurch verbessern.

Systemintegration Bei der Systemintegration wurde die Servoansteuerung des Magazins in das vorhandene Framework eingebunden. Es gibt zwei Möglichkeiten den Ball auszulösen.

- Wird über das Signal des Piezosensors ein Ballverlust detektiert wird im Magazin automatisch ein neuer Ball ausgelöst.
- Wird das Spiel mit dem Joystick gespielt, kann durch den Spieler bei einem Ballverlust ein Knopf am Joystick gedrückt werden, um einen neuen Ball auf die Startposition zu bekommen.

Da das Detektieren eines Ballverlustes mit dem Piezosensor derzeit noch nicht zuverlässig genug funktioniert, wird die Information des Piezosensors nicht genutzt um einen Ball auszulösen. Sobald eine zuverlässige Balldetektion garantiert ist, wird diese zum Auslösen eines Balls benutzt werden. Daher wird zur Zeit nur die Ansteuerung per Joystick verwendet.
Die Ansteuerung des Magazins ist in der Klasse *Motor_Control* untergebracht. Wird von der Joystickbzw.Klasse ein entsprechendes Signal gesendet, fährt die Spielfläche in die Ausgangsposition und der Servo im Auslösemodul betätigt die Schubstange des Magazins. Sobald die Detektion eines Ballverlustes fehlerlos funktioniert, wird das gleiche Verfahren zum Auslösen eines Balles benutzt.
Zusätzlich kann das Magazin so konfiguriert werden, dass ein Ball manuell ausgelöst werden kann. Dafür wird das automatische Auslösemodul gegen das manuelle ausgetauscht. Jetzt ist es möglich, jederzeit über einen Knopfdruck am Auslösemodul einen neuen Ball zuzuführen. Da in dieser Situation keine Servos zur Ansteuerung des Spiels benutzt werde, ist vom Spieler darauf zu achten, dass sich die Spielfläche in der waagerechten Positionen befindet.

AP4: Untersuchung von Simulationsparameter

Die Arbeiten wurden im Quartal Q3-2008 abgeschlossen. **BRIO Modellierung**
In diesem Arbeitspaket wurden entworfenen Experimente, durchgeführt und die Parameter der Simulation entsprechend optimiert. Die quantitativen Experimente wurden in drei Schritten durchgefhürt:

1. **Data Akquisition:** Die Experimente wurden in dem realen Labyrinth 20-mal wiederholt durchgefhrt, dabei werden die Kugelposition (x und y Position auf dem Brett) und Die Motorposition (die Potentiometers Werte in Grad) mit der Frequenz 1 kHz geloggt
2. **Unterschied minimieren** Die gleiche Experimente sind in der Simulation durchgefhürt und die gewhälten Parameter sind in einem iterativen Verfahren angepasst bis der Unterschied zwischen den realen und simulierten Daten minimal ist.
3. **Visuelle Verifikation** Das simulierte und das reale Spiel sind gleichzeitig durchgefhrt und der Unterschied ist in der Simulation visualisiert.

Es wurden insgesamt 3 Experiment durchgefhrt:

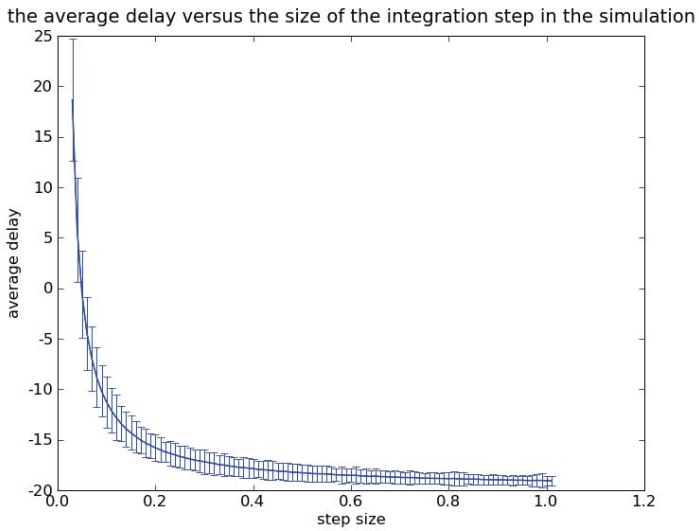


Abbildung .14: Die Beziehung zwischen die Schrittgröße und die Resultierendente Verzögerung um 20 ms zu berechnen. Eine negative Verzgerung (-d) bedeutet dass die Simulation hat durchnitlich (20-d) ms gebraucht um 20 ms zu berechnen. die durchnitt ist über 1000 Wiederholungen berechnet

- **Schritt Größe des Integrators:** Ziel dabei ist, die Bestimung die minimale Zeit zwischen zwei schritte in der Simulation, die in Echtzeit berechenbar ist. Die Abbildung .14 stellt die Beziehung zwischen die Zeitschrittgröße und die durchnittliche Verzgerung dar. Eine Negative Verzgerung $-d$ ist als keine Verzgerung zu interpretieren. Sondern das die Simulation hat $20 - d$ ms gebraucht um 20 ms zu berechnen. Die Schrittgre 0.1 ist ein guter Kompromiss zwischen Genauigkeit und Schnelligkeit.

- **Motoren Parameter:** Die Motoren sind mit ein P-Regler Modelliert und die Wert p = 1.66 ist das beste Werte der das Verhalten der realen Motoren simuliert.

- **Reibungskoefizient:** In der Simulation wird die Reibung durch ein Coulomb-Modell behandelt. Das Coulomb-Modell ist eine einfache aber effektive Methode. Es handelt sich um eine einfache Beziehung zwischen den normalen und tangentialen Kräften in einem Kontaktpunkt. Die maximale Gleitreibungskraft F_{tM} zwischen zwei Körpern wird durch die folgende Gleichung berechnet:

$$|F_{tM}| = \mu * |F_n|$$

  F_n ist der normale Kraftvektor, und μ ist der Reibungskoeffizient. μ kann beliebige Werte zwischen 0 und ∞ annehmen. Der Wert $\mu = 10$ ist die beste manuelle gefundene Lösung für den Reibungskofezient

Um die Ergebnisse in der Brio-Simulation zu optimieren, wurde mit 3ds Max ein neues 3d-Objekt des Brio Labyrinths modelliert. Als Grundlage wurde die Spielfläche eingescannt und ein hochauflösendes Foto der Draufsicht generiert (Abbildung .15). Mithilfe dieses Fotos wurde zuerst ein Teil der Labyrinthwand modelliert, welche anschließend genutzt wurde, um die gesamte Spielfläche zusammenzusetzten.

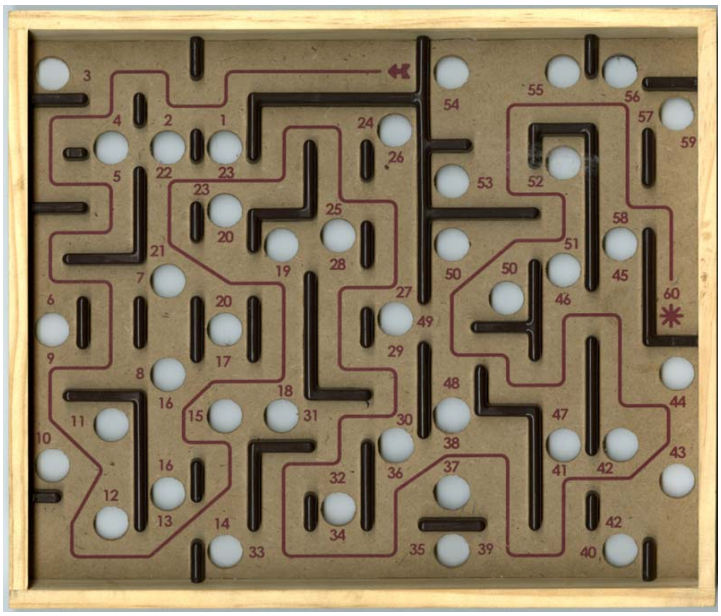


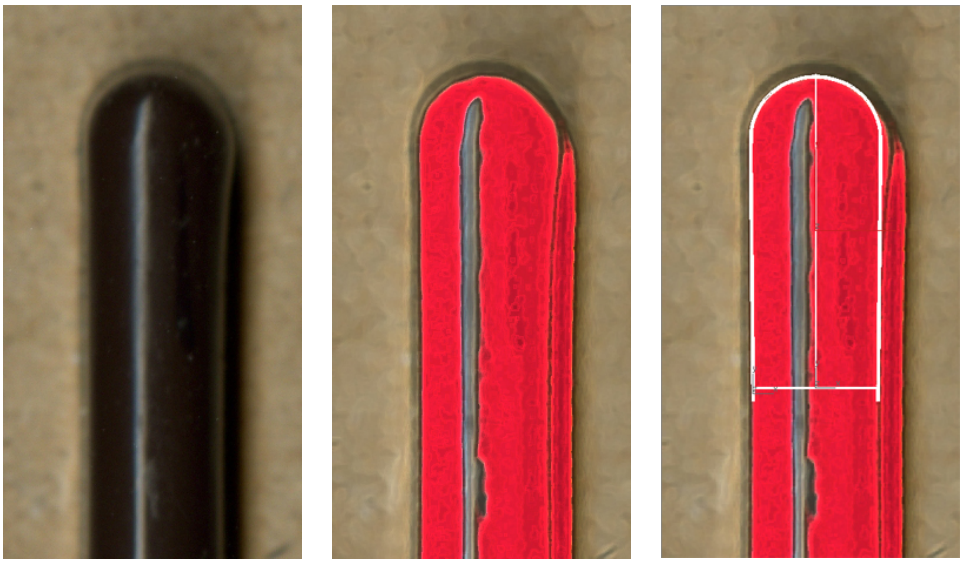
Abbildung .15: Brio Labyrinth Vorlage

Labyrinthwand

3ds Max bietet die Möglichkeit, ein Foto in den Hintergrund zu legen und als direkte Vorlage zu nutzen. Hierbei werden jedoch hochauflösende Fotos stark herunter skaliert, weswegen eine genaue Modellierung der einzelnen Labyrinthwände auf Basis der gesamten Spielfläche nicht möglich war. Aus diesem Grund wurde aus dem Gesamtfoto der Ausschnitt einer Labyrinthwand extrahiert (Abbildung .16(a)). Auf diesem Ausschnitt kann man erkennen, dass die Labyrinthwände nach unten hin größer werden und an der Spitze unregelmäßig geformt sind. Rechts ist außerdem ein deutlicher Schattenwurf zu erkennen. Um den oberen Bereich der Labyrinthwand - der Bereich, an dem die Kugel anliegt - vom unteren Bereich und dem Schatten deutlicher abzugrenzen, wurden verschiedene Bildbearbeitungs-Filter angewendet (Abbildung .16(b)). Das daraus resultierende Foto konnte abschließend in 3ds Max zur Modellierung verwendet werden. Der Einfachheit halber wurde die unregelmäßige Form der Labyrinthwände nicht berücksichtigt und eine symmetrischer Umriss modelliert (Abbildung .16(c)). Hierbei diente die linke Seite der Labyrinthwand als Referenz.

Spielfläche

Aus der zuvor erstellten Labyrinthwand wurde anschließend das gesamte Labyrinth zu-



(a) Ausschnitt (b) Segmentiert (c) Modelliert

Abbildung .16: Brio Labyrinthwand

sammengesetzt. Jede einzelne Wand verdickt sich dementsprechend am Ende leicht und ist ansonsten ideal gerade. Anschließend wurde der Rahmen und der Boden erstellt und texturiert. Aus dem Boden wurden zusätzlich die Löcher mithilfe eines Refernzzylinders ausgestanzt. In Abbildung .17(a) ist das alte Spielbrett zu erkennen. Wie in Abbildung .17(b) zu erkennen, wurden in der neuen Version des Spielbrettes alle Ecken durch die zuvor modellierte Rundung ersetzt. Schließlich wurde die Größe des Modells auf 2,905% reduziert, um es an die Größe des Originals anzupassen. Insgesamt konnte die Genauigkeit - auch aufgrund der neuen Fotovorlage - erheblich verbessert werden.

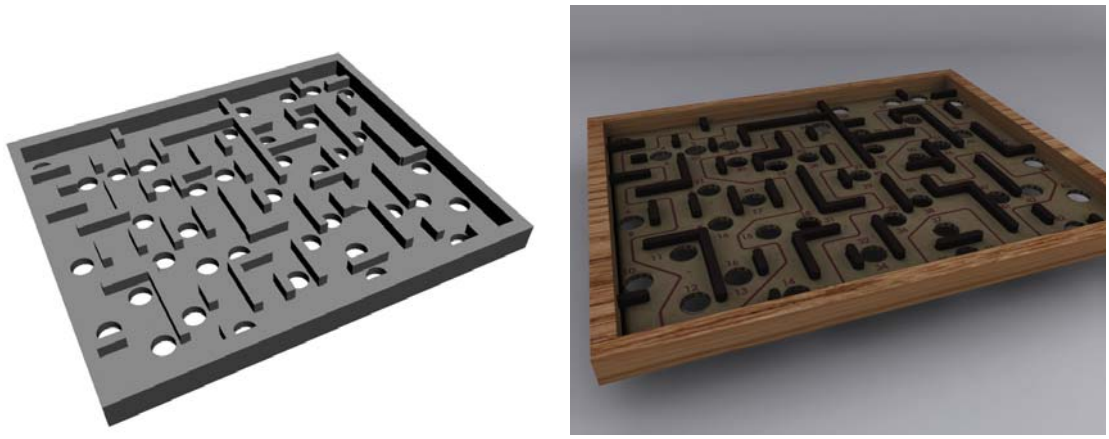
In diesem Arbeitspaket wurde das Brio-Labyrinth in der Simulation Umgebung MARS integriert und das physikalische Model manuell optimiert, Das Spielfeld wurde durch ein 3ds Max erstelltes Trimesh nachgebaut. Die Simulation wurde dann durch experimentelle Untersuchungen von drei gewählte physikalischen Parameters optimiert.

AP5: Echtzeit-Zustandsschätzung des Labyrinths

Die Arbeiten wurden in Q02-2008 abgeschlossen.

Im Berichtszeitraum wurde eine Echtzeits-Zustandsschätzung des Labyrinths realisiert. Der Zustand des Labyrinths setzt sich aus Kugelposition, Kugelgeschwindigkeit, und Brettorientierung zusammen. Die Orientierung des Brettes lässt sich mittels zweier Potentiometer messen. Die von den Potentiometern gelieferten Werte werden wie folgt in Winkel des Bretts umgerechnet: Während der Kalibrierung des Brio-Spiel (siehe AP2), werden die Werte der Potentiometer für die maximale Auslenkung des Bretts[1] in jeder

[1]Die maximale Auslenkung ist definiert als der Zeitpunkt, zu dem der jeweilige Endschalter aktiviert



(a) Alte Version (b) Neue Version

Abbildung .17: Brio Labyrinth

der vier Richtungen gemessen. Für jede dieser vier maximalen Auslenkungen wurde der jeweilige Winkel des Bretts (das heißt der Winkel relativ zu einer horizontalen Ausrichtung des Bretts) manuell bestimmt. Hierbei korrespondieren Änderungen der Brettorientierung gegen den Uhrzeigersinn zu Verringerungen des Winkels. Die später online gemessenen Potentiometerwerte werden mittels der so gewonnen Referenzzuordnungen (Poti-Wert $\rightarrow$ Winkel) in Winkel umgerechnet. Hierbei wird die vereinfachte Annahme gemacht, dass diese Abbildung linear ist. Die Gültigkeit dieser Annahme bleibt zu validieren.

Für die Schätzung der Ballposition wird die über dem Brett angebrachte Kamera benutzt. Im Kamerabild kann die (weiße) Spielkugel leicht detektiert werden, da sie in der kontrollierten Umgebung des Brio Labyrinth Spiels das hellste Objekt ist. Insofern reicht es zur Detektion der Kugel das Grauwertbild der Kamera stark zu glätten und dann das hellste Pixel in diesem geglätteten Bild zu finden. Dieses korrespondiert mit hoher Wahrscheinlichkeit zu einem Pixel das einen Teil der Kugel darstellt[2]. Um nun das Zentrum der Kugel zu finden werden im Umkreis um das gefundene Pixel alle Pixel gesammelt, deren Helligkeit um weniger als 10% unter der des Pixels mit maximaler Intensität liegt. Diese Pixel werden als Bildpunkte angesehen, die mit hoher Wahrscheinlichkeit einen Teil der Spielkugel darstellen. Der Mittelwert der Koordinaten aller dieser Pixel wird als geschätzte Kugelposition (in Bildkoordinaten) berechnet. Um das Verfahren zu beschleunigen, wird in jedem weiteren Schritt nur im unmittelbaren Umfeld der letzten detektierten Kugelposition nach der neuen Position gesucht.

Die Kugelposition liegt nun im Bildkoordinatensystem vor. Gewünscht ist jedoch eine Angabe der Position in einem von der relativen Lage/Ausrichtung von Kamera und

wird.

[2]Dieses Vorgehen hat sich für den Moment als hinreichend stabil und schnell erwiesen. Es wäre jedoch wünschenswert in Zukunft ein robusteres Verfahren zu entwickeln, das den Rechenaufwand nicht deutlich erhöht.

Brett unabhängigen Koordinatensystem. Dieses (zweidimensionale) Labyrinthkoordinatensystem wird wie folgt definiert: Die obere, linke Ecke des Spielbretts wird als $(0, 0)$ definiert. Die obere, rechte Ecke des Spielbretts wird als $(31.1, 0.0)$ definiert, die untere, linke Ecke des Spielbretts als $(0.0, 26.5)$. Dementsprechend besitzt die untere, rechte Ecke des Spielbretts die Koordinate $(31.1, 26.5)$. Die Einheit dieses Koordinatensystems ist Zentimeter.

Die Transformation von Bild- in Labyrinthkoordinaten wird als linear angenommen und während der Kalibrierung des Spielbretts berechnet. Hierzu werden die vier Eckpunkte des Spielbretts bei horizontaler Ausrichtung desselben einmalig im Kamerabild detektiert. Diese Detektion ist vergleichsweise einfach, da in der kontrollierten Umgebung des Brio-Systems außerhalb des Spielbretts alles schwarz bzw. sehr dunkel gefärbt ist. Die vier Eckpunkte des Labyrinth sind deswegen sehr starke Ecken (corners) im Grauwertbild der Kamera, da das Brett selbst sehr hell ist. Es genügt nun einmalig im Kamerabild eine Cornerdetection durchzuführen (mittels OpenCV), und nur die Corner mit sehr starker Intensität beizubehalten. Teilt man das Bild nun in vier Quadranten ein und sucht in jedem dieser Quadranten nach der Corner, deren Koordinate maximal stark vom Bildmittelpunkt abweicht, so kann man damit vergleichsweise zuverlässig die vier Eckpunkte des Spielbretts detektieren[3]. Diese vier Landmarken korrespondieren direkt zu den vier Referenzkoordinaten des Labyrinthkoordinatensystems. Somit hat man vier Beispielzuordnungen von Bild- zu Labyrinthkoordinate und die lineare Abbildung ist somit überbestimmt. Sie wird mittels eines Least-Squares Ansatzes bestimmt.

Die Kugelposition kann nun mittels des oben beschriebenen Verfahrens jederzeit im Labyrinthkoordinatensystem bestimmt werden. Um die Schätzung der Position unanfälliger gegenüber Rauschen zu machen und die Schätzung der Geschwindigkeit zu erlauben, wurde ein Kalman-Filter implementiert. Als gemessener Zustand wird die detektierte Kugelposition, eine Schätzung der Kugelgeschwindigkeit[4], sowie die gemessenen Brettwinkel an den Kalmanfilter bergeben. Der Prozess selbst wird als linear angenommen, wobei sich die Beschleunigung der Kugel aus den Brettwinkeln ableitet. Dieses Verhältnis wird eigentlich durch die Gleichung für die schiefe Ebene bestimmt ($a = g * sin(\alpha)$, wobei a die gesuchte Beschleunigung, g die Konstante für die Schwerebeschleunigung der Erde, und α die Brettneigung ist). Für kleine Winkel α (wie sie im Brio System stehts vorliegen) kann diese Beziehung durch $a \approx g * \alpha$ angenähert werden (vergleiche Taylorreihe des Sinus), was den Einsatz eines einfachen, linearen Kalmanfilters erlaubt.

Die durch diesen Kalmanfilter gelieferten Werte sind noch nicht systematisch analysiert worden[5], jedoch hat eine simultane Anzeige der Kugel in der Visualisierung der MARS-Simulation mit dem Spielen des realen Labyrinths gezeigt, dass zumindest die Ballposition sehr genau geschätzt wird und die Verzögerung sehr gering ist.

Es wurden einige kleine Verbesserungen an den im letzten Bericht vorgestellten Verfahren durchgeführt: Sofern die Kugel die Region of Interest verlässt (etwa weil sie durch ein Loch fällt), wird dies nun vom Algorithmus bemerkt und die Region of Interest

[3] Zuverlässigere Landmarken sind im Brio Spiel schwierig zu definieren und eine explizite Anbringung von künstlichen Landmarken soll vermieden werden.

[4] Differenz alter und neuer Kugelposition geteilt durch die Zeit zwischen diesen beiden Messungen.

[5] Diese systematische Analyse geschieht im Rahmen einer Masterarbeit.

wieder auf das Gesamtbild gesetzt. Dadurch kann die Kugel nun auch über mehrere Episoden hinweg verfolgt werden. Im Falle, dass keine Kugel detektiert wird, wird der Kalman-Filter deaktiviert und es wird keine Kugelposition geschtäzt.

AP6: Integration in das MML-Framework

Die Arbeiten an diesem AP wurden in Q4 2008 abgeschlossen.

Die Schnittstelle zwischen MARS Simulation und dem MMLF wurde festgelegt. Folgende Informationen werden von MARS an das MMLF übertragen: Die 3D Position der Simulationskugel, die 3D Geschwindigkeit der Simulationskugel sowie die Ist-Motorpositionen. Umgekehrt kann das MMLF die neuen Soll-Motorpositionen, einige Konfigurationsparameter (Reibung, PID Werte der Regler etc.) sowie ein Reset-Kommando an MARS senden. Sämtliche Kommunikation des MMLF mit dem in MARS simulierten BRIO System läuft über ein (Software-)Interface. Die Ansteuerung des realen BRIO Systems kann über ein analoges Interface erfolgen, das die selben Methoden bereitstellt. Somit stellt es aus Sicht des MMLF keinen Unterschied dar, ob das reale oder das simulierte System genutzt werden.

Das Erlernen einer Strategie für das BRIO-Spiel wurde in Form eines Markovschen Entscheidungsprozess (MDP) definiert. Hierbei besteht die Zustandsmenge aus 6-dimensionalen (kontinuierlichen) Zustandsvektoren, die sich aus den beiden Brettneigungswinkeln, den Kugelpositionen und den Kugelgeschwindigkeiten in beiden Richtungen zusammensetzen. Das MMLF kann optional Rauschen in den Sensoren simulieren, wodurch das Problem nur noch partiell beobachtbar ist (POMDP). Als mögliche Aktionen kann der Agent die beiden Soll-Brettneigungswinkel vorgeben; der Aktionsraum wurde hierzu künstlich diskretisiert, d.h. der Agent kann den momentanen Ist-Brettneigungswinkel nur um einen fest vorgegebenen Betrag (1.5°) in jede der beiden Richtungen ändern. Aufgrund der beiden unabhängigen Achsen stehen dem Agenten somit 4 verschiedene Aktionen zur Auswahl. Das Hinzufügen einer fünften Aktion ("behalte die momentane Brettneigung bei") hat sich als nicht vorteilhaft erwiesen. Die Zustandsübergangswahrscheinlichkeiten ergeben sich direkt aus der physikalischen Simulation (es handelt sich somit um ein deterministisches MDP). Die Rewardfunktion wurde wiefolgt definiert: Der Agent erhält nach jedem Zeitschritt einen Reward von -0.1, es sei denn die Kugel fällt in ein Loch (Reward -50) oder erreicht einen Wegpunkt zum ersten Mal (Reward +10). Passiert die Kugel einen Wegpunkt in die falsche Richtung, so wird der zuvor erhaltene Reward wieder abgezogen, d.h. ein Reward von -10 gegeben.

Die Wegpunkte (Subgoals) wurden manuell bestimmt, dazu wurde das Labyrinth vermessen. Die Wegpunkte wurden wiefolgt festgelegt: Auf dem Brio-Labyrinth ist eine Ideallinie eingezeichnet. An den Stellen, an denen diese Ideallinie ihre Richtung ändert, wurde ein Wegpunkt gesetzt (vergleiche auch Figur .19). Ein Wegpunkt gilt als erreicht, wenn die Kugel sich entweder sehr nah zu diesem befindet oder sich die Entfernung zu diesem vergrößert und zugleich die Entfernung zum nächsten Wegpunkt abnimmt. Hierdurch wird der Agent nicht gezwungen, einen Weg genau durch diesen Wegpunkt zu nehmen.

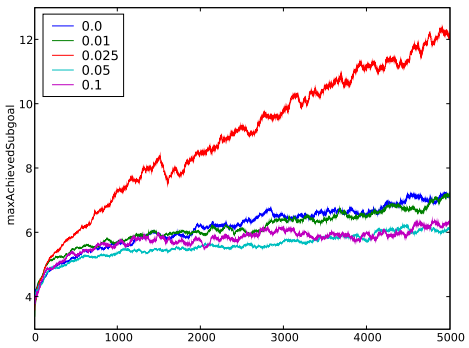
Im Rahmen dieses Arbeitspakets wurden einige Untersuchungen mit dem im MM-

LF enthaltenen TD-Lambda-Agenten (der sowohl für Watkins Q-Learning also auch für SARSA genutzt werden kann) durchgeführt, sowie der Einfluss verschiedener Parameter auf die Lerngeschwindigkeit[6] bestimmt. Da es sich bei dem oben definierten BRIO-MDP um ein MDP mit kontinuierlichem (d.h. insbesondere unendlich großem) Zustandsraum handelt, können nur Agenten genutzt werden, die einen Funktionsapproximator einsetzen, um Value-Funktion oder Policy zu repräsentieren. In diesen Untersuchungen wurde der CMAC Funktionsapproximators eingesetzt. In Figur .18 sind die Ergebnisse für einen Teil der untersuchten Parameter dargestellt; sämtliche Kurven sind Mittelwerte über 10 unabhängige Durchläufe. In den Untersuchungen wurden jeweils alle Parameter mit Ausnahme des angegebenen festgehalten. In Figur .18a ist der Einfluss des Explorationsparameters ϵ dargestellt, der bestimmt wie oft der Agent eine Aktion ausprobiert, die er für suboptimal hält. Es hat sich gezeigt, dass die Wahl von ϵ sehr kritisch für die Lerngeschwindigkeit ist; von allen geprüften Werten hat nur $\epsilon = 0.025$ zu guten Ergebnissen geführt. Aufgrunddessen sollen weitere Untersuchungen durchgeführt werden. Der Einfluss des Diskontierungsfaktors γ (Figur .18b), der bestimmt ob der Agent eher versucht, den kurz- oder langfristigen Reward zu optimieren, ist ebenfalls relevant für die Lerngeschwindigkeit. Hierbei haben sich Werte nahe dem maximal möglichen Wert $\gamma = 1.0$ als optimal erwiesen, d. h. ein Agent der möglichst weit vorausschauend agiert. Ein Parameter des CMAC-Funktionsapproximator ist die Anzahl der Tilings, die beeinflusst wie glatt die repräsentierbaren Funktionen sind. In Figur .18c ist zu zu erkennen, dass eine höhere Anzahl Tilings zu einem schnelleren Lernen führt (in Hinblick auf die notwendigen Trainingsepisoden). Allerdings geht eine Erhöhung der Anzahl der Tilings mit einer linearen Erhöhung der Rechenzeit einher. Daher kann die Anzahl Tilings nicht beliebig erhöht werden. Aufgrunddessen könnte es sinnvoll sein, in Zukunft andere Funktionsapproximatoren auszuprobieren, in denen man die gewünschte Glätte bestimmen kann ohne den Rechenaufwand zu erhöhen. In Figur .18d sind die off-policy Lernregel WatkinsQ und die on-policy Lernregel SARSA verglichen worden. Die vorläufigen Ergebnisse deuten darauf hin, dass initial das off-policy verfahren Q-Learning schneller lernt, später jedoch das on-policy Verfahren SARSA. Dies könnte daran liegen, dass ein on-policy Verfahren in der Lage ist, zu lernen, dass der Explorationsmechanismus dazuführen kann, das in der Nähe eines Loches eine suboptimale Aktion durchgeführt wird, und somit Wege mit einem großem Sicherheitsabstand zu den Löchern besser sind.

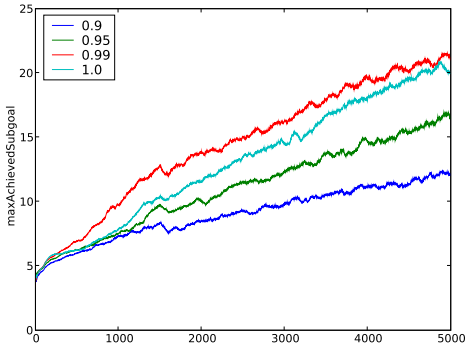
Neben dem CMAC-Funktionsapproximator könnten prinzipiell auch eine Reihe anderer Funktionsapproximatoren, die im MMLF implementiert sind, wie beispielsweise künstliche neuronale Netze (ANNs) oder QCON, oder direkte Policysuchverfahren wie die neuroevolutionäre Methode EANT im BRIO Szenario getestet werden. Allerdings hat sich bei Tests herausgestellt, dass im oben definierten BRIO-MDP der Funktionsapproximator lokal sein sollte, d. h. das eine Änderung von $Q(s, a)$ (bzw.$\pi(s)$) für einen Zustand s nur $Q(s', a)$ (bzw.$\pi(s')$) für Zustände s' ändern sollte, für die gilt $d(s, s') < \epsilon$.[7]

[6] Die Lerngeschwindigkeit wurde hierbei in "Maximal erreichter Wegpunkt vs. dafür benötigte Lernepisoden" gemessen.

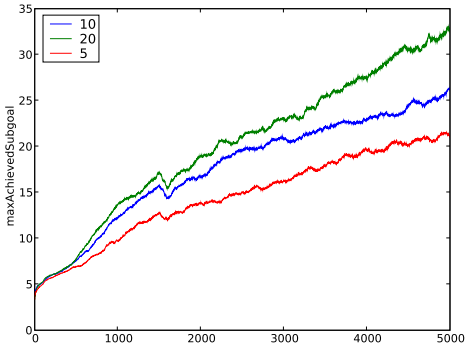
[7] Dies liegt darin begründet, dass die optimale Policy/Valuefunction über dem Zustandsraum stark zerklüftet ist und an vielen Stellen nicht stetig ist, da der Zustandsraum durch Wände und Löcher unterteilt wird. Die Repräsentation einer solchen Funktion durch einen monolithischen, globalen



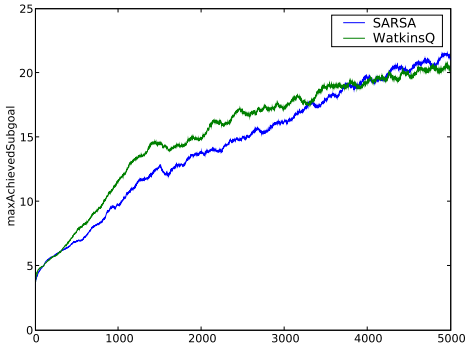
(a) Explorationsparameter ϵ



(b) Diskontierungsfaktor γ



(c) Anzahl Tilings



(d) Lernregel

Abbildung .18: Untersuchung des Einflusses verschiedener Einstellungen des TD-Agenten auf die Lerngeschwindigkeit. Die X-Achse gibt jeweils die Anzahl der absolvierten Trainingsepisoden an, die Y-Achse die Nummer des erreichten Wegpunkte (72 entspräche dem Ziel).

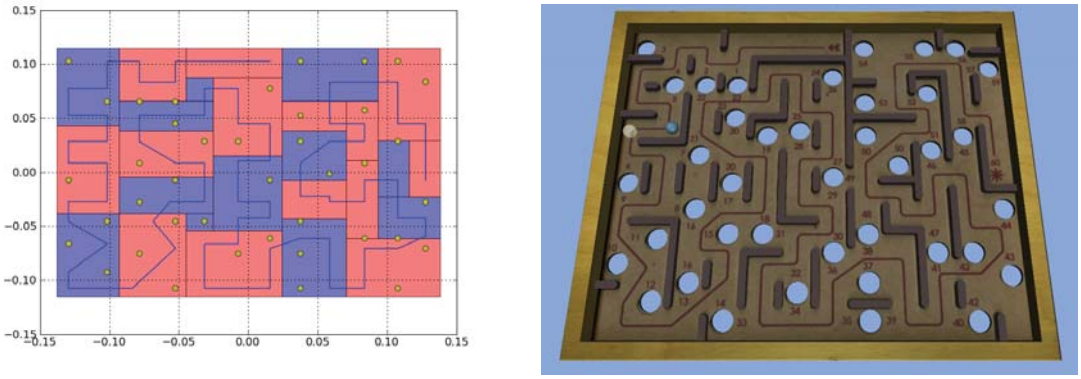


Abbildung .19: Die linke Graphik zeigt die Löcher des Spielbretts (gelb), den optimalen Pfad (blau) und die Zerlegung des Bretts in Regionen. Die rechte Graphik zeigt zum Vergleich das Labyrinth selbst.

Dies ist für CMAC der Fall, für Standard-ANNs jedoch nicht. Um globale Funktionsapproximatoren einsetzbar zu machen, musste also entweder die Zustandsrepräsentation des BRIO-MDP geändert oder das BRIO-Problem in kleinere Teilprobleme zerlegt werden. Der zweite Ansatz wurde im Rahmen dieses APs realisiert.

Diese Zerlegung des BRIO-Problems in Teilprobleme läuft darauf hinaus, statt eines einzelnen Funktionsapproximators für das ganze Brett mehrere zu nutzen, von denen jeder nur für einen kleinen Teilbereich des Zustandsraums verantwortlich ist. Dieser Ansatz war schon in der Diplomarbeit von Larbi Abdenebaoui verfolgt worden, in der mehrere Instanzen des QCON-Funktionsapproximators jeweils für einen Teilbereich des BRIO-Spiels verantwortlich waren. Basierend auf diesem Ansatz ist im MMLF ein Meta-Funktionsapproximator implementiert worden, der speziell auf das BRIO-Spiel zugeschnitten ist: Für jede Region des Bretts delegiert der Meta-Funktionsapproximator die Berechnung des Q-Wertes an den für diesen Bereich zuständigen Basis-Funktionsapproximator. Der Vorteil dieser Implementierung ist, dass nicht nur ein spezieller Funktionsapproximator, sondern jeder im MMLF implementierte Funktionsapproximator die Zerlegung nutzen kann. Die Zerlegung des Bretts in Teilbereiche ist in Figur .19 dargestellt. Die Kombination von EANT und darauf aufbauenden Verfahren mit dieser Zerlegung wird momentan im Rahmen der Masterarbeit von Tchando Kongue durchgeführt.

Die Untersuchung der Übertragbarkeit erlernter Verfahren aus der Simulation auf das reale System wird im Rahmen der Masterarbeit von Constantin Bergatt erfolgen. Ziel dieser Arbeit ist die Untersuchung und Quantifizierung des Simulation-Reality Gaps und seines Einflusses auf die Übertragbarkeit von erlernten Policies von Simulation auf das reale System.

Funktionsapproximator würde erfordern, dass dieser ein hohes Maß an Komplexität besitzt (d.h. sehr viele Neuronen im Falle eines ANNs). Momentane neuroevolutionäre Verfahren mit einer direkten genetischen Kodierung des ANNs können dies noch nicht leisten.

Gegeben sei

- Eine Menge von zu optimierenden Parametern $P = \{p_o, ..., p_n\}$. Diese werden in AP4 festgelegt.
- Eine Reihe von Sensorwerten $S = \{s_0, ..., s_m\}$, die sowohl am realen System als auch an der Simulation gemessen werden können
- Eine Reihe von Aktionsfolgen $A = \{A_1, ..., A_l\}$, wobei jede Aktionsfolge aus eine Folge von Aktionen besteht: $A_i = [a_{i0}, ..., a_{it_i}]$
- Eine Zielfunktion $F(P)$
- Ein Optimierungsalgorithmus O, der genutzt werden kann um ein (lokales) Optimum einer Zielfunktion zu finden

Gesucht ist

- Ein Wert P, der $F(P)$ minimiert/maximiert.

Basierend auf dieser Formalisierung soll das (vorläufige) Vorgehen wie folgt aussehen:

1. Jede Aktionsfolge $A_i \in A$ wird auf dem realen Labyrinth (mehrmals) durchgeführt und für jede Folge eine Reihe von Sensordaten $S^{real,i} = [(s_{00}^{real,i}, ..., s_{m0}^{real,i}), ..., (s_{0t_i}^{real,i}, ..., s_{mt_i}^{real,i})]$ aufgezeichnet.
2. Für jeden Wert Parametervektor P werden dieselben Aktionsfolgen in der Simulation durchgeführt, woraus eine Reihe von Sensordaten $S^{sim,i} = [(s_{00}^{sim,i}, ..., s_{m0}^{sim,i}), ..., (s_{0t_i}^{sim,i}, ..., s_{mt_i}^{sim,i})]$ resultiert.
3. Die Zielfunktion $F(P)$ wird als $F(P) = \frac{1}{l} \sum_{i=1}^{l} \frac{1}{t_i} \sum_{j=1}^{t_i} \frac{1}{m} \sum_{k=1}^{m} |s_{kj}^{sim,i} - s_{kj}^{real,i}|$ definiert, also als mittlere Abweichung der Sensordaten auf simulierten und realem System.
4. Ein Optimierungsalgorithmus O (bspw. Gradient-Descent oder CMA-ES) wird eingesetzt, um ein optimales P zu finden.

Abbildung .20: Konzept für die automatische Kalibrierung

AP7: Automatische Kalibrierung der Simulation

Die Arbeiten an diesem AP wurden in Q4 2008 abgeschlossen.

Ziel der automatischen Kalibrierung ist es, eine Reihe von Parametern der Simulation so zu wählen, dass das Verhalten der Simulation möglichst nahe an dem des realen System liegt. Als erster Schritt wurde eine Formalisierung der Problemstellung für die automatische Kalibrierung der Labyrinth-Simulation entwickelt (vergleiche Figur .20).

Dieses Konzept, das auch in Figur .21 visualisiert ist, wurde im Rahmen eines Python-Projekts im MMLF realisiert. Für die Kommunikation mit dem simulierten und realen BRIO-System wurden die in AP6 entwickelten Schnittstellen benutzt. Als Optimierungsalgorithmus wurde CMA-ES ausgewählt.

Die Implementierung ist modular und erweiterbar ausgelegt, sodass man die zu optimierenden Parameter, die relevanten Sensoren, die zu testenden Aktionsfolgen, die Zielfunktion und den Optimierungsalgorithmus frei wählen und kombinieren kann. Eine Schwierigkeiten stellten bei der Implementierung die unterschiedlichen Eigenschaften von simulierter und realer Zeit dar, insbesondere die verschiedenen Abtastraten des rea-

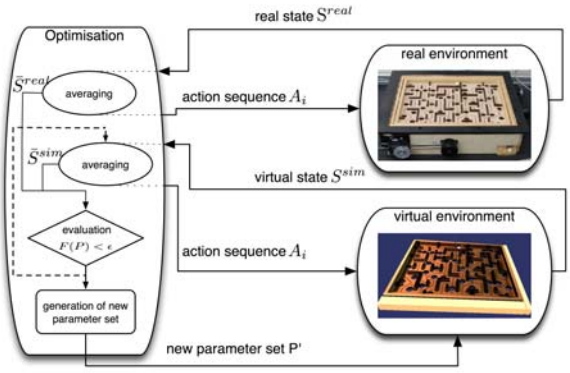
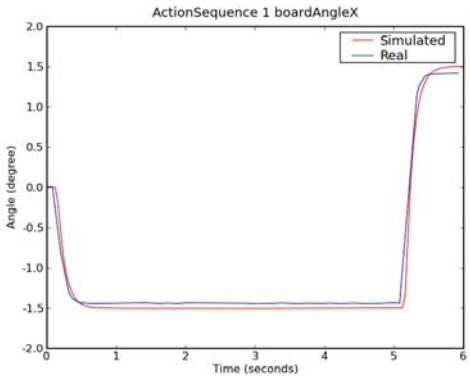


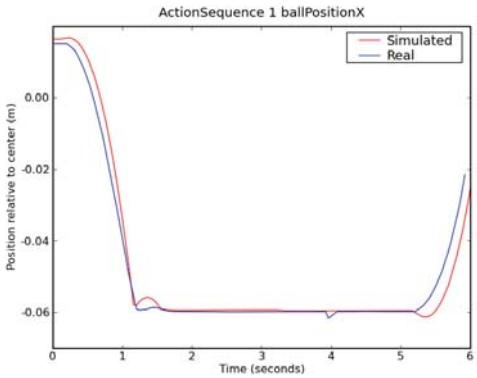
Abbildung .21: Visualisierung des Datenflusses in der automatischen Kalibrierung

len und simulierten Systems. Das Konzept geht bei der Definition der Zielfunktion davon aus, dass für einen Zeitpunkt t jeweils genau ein Sensorwert $s_{kt}^{sim,i}$ des simulierten Systems und ein Sensorwert $s_{kt}^{real,i}$ des realen Systems vorliegt. Dies kann aus zwei Gründen für das reale System nicht eingehalten werden: Zum einen sind die Abtastraten nicht beliebig hoch wählbar und werden auch ansonsten nicht exakt eingehalten, zum anderen sind die realen Sensoren verrauscht, sodass es nicht einen klar definierten, "richtigen" Sensorwert gibt. Daher wurden die notwendigen $s_{kt}^{real,i}$ wie folgt bestimmt: Zunächst wird jede Aktionsfolge A_i mehrmals auf dem realen System ausgeführt. Die Sensoraufzeichnungen, die aus der jeweiligen Ausführung resultieren, werden daraufhin jeweils linear interpoliert, sodass für jeden Zeitpunkt t ein interpolierter Sensorwert bestimmt werden kann. Um $s_{kt}^{real,i}$ zu bestimmen werden sodann die interpolierten Werte aller zu A_i gehörenden Sensorkurven gemittelt, um das Rauschen in den Sensoren zu verringern.

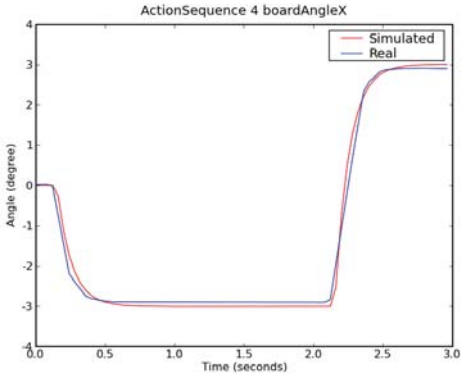
Im Rahmen des Arbeitspakets wurde das Verfahren im Rahmen der Anpassung der PID Werte des Reglers, der in der Simulation die Servos ansteuert, validiert. Die zu optimierenden Parameter waren hierbei die P, I und D Werte des Reglers sowie ein "Delay" Parameter, der steuert wie stark verzögert die simulierten Regler auf ein Kommando reagieren. Dieser Parameter war notwendig, da die realen Servos verzögert reagieren, vermutlich aufgrund des Kommunikationsoverhead. Die aufgezeichneten "Sensorwerte" sind die Auslenkung des Bretts in beiden Richtungen sowie die Ballposition. Durchgeführt wurden 4 verschiedene Aktionssequenzen: In diesen wurde veranlasst, dass der Servo sich zunächst verschieden weit gegen den Uhrzeigersinn dreht und dann nach unterschiedlichen Zeiträumen in die andere Richtung gegengesteuert. Nachdem die Daten auf dem realen System für diese Kurven aufgezeichnet wurden, wurde mittels des oben skizzierten Verfahrens unter Nutzung des CMA-ES Optimierungsverfahrens die optimalen Simulationsparameter bestimmt. Die resultierenden Kurven der sind in Figur .22 dargestellt. Wie die Graphiken zeigen stimmen nicht nur die Motorkurven gut überein, sondern auch die Bewegungen von realer und simulierter Kugel. Die geringen Abweichungen sind unter anderem darin begründet, dass der reale Servo die gewünschten Zielpositionen nicht genau anfährt, es also einen nicht vorhersagbaren Fehler im realen System gibt, der sich



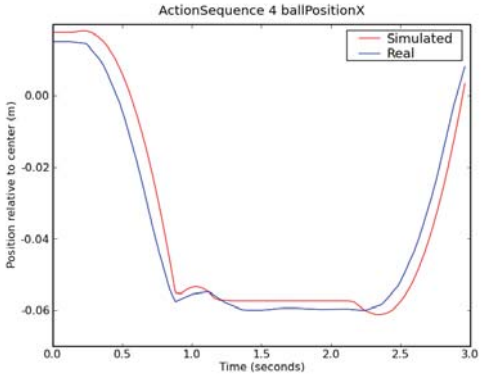
(a) Entwicklung des Brettwinkels in X-Richtung



(b) Entwicklung der Ballposition in X-Richtung



(c) Entwicklung des Brettwinkels in X-Richtung



(d) Entwicklung der Ballposition in X-Richtung

Abbildung .22: Die Figuren zeigen die Entwickluing von Brettwinkel und Ballposition in X-Richtung für zwei verschiedene Aktionssequenzen auf simuliertem und realem BRIO-System.

prinzipiell nicht simulieren lässt. Somit hat die automatische Kalibrierung für dieses Szenario wie erhofft funktioniert. Ein analoges Vorgehen kann in Zukunft bei Bedarf auch für weitere Simulationsparameter durchgeführt werden.

Literaturverzeichnis

[1] Dynamixel. *Dx Series*. Datenblatt der Modelle DX-113, DX-115 und DX-117.

[2] Phil Hassey. Pygame - python game development. Website. `http://www.pygame.org` Abruf am 19.10.2008.

[3] Phil Hassey. Pygame - python game development, joystick reference. Website. `http://www.pygame.org/docs/ref/joystick.htm` Abruf am 19.10.2008.