FH Westküste
Wirtschaft und Technik

HAW HAMBURG

Masterarbeit

Jonas Kempke

Entwicklung eines KI gesteuerten Tischkickers unter Verwendung von Reinforcement Learning

Fachhochschule Westküste
Fachbereich Technik

Fachhochschule Westküste - University of Applied Sciences
Faculty of Engineering

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Hamburg University of Applied Sciences
Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Jonas Kempke

Entwicklung eines KI gesteuerten Tischkickers unter Verwendung von Reinforcement Learning

Masterarbeit eingereicht im Rahmen der Masterprüfung
im gemeinsamen Masterstudiengang Mikroelektronische Systeme
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg
und
am Fachbereich Technik
der Fachhochschule Westküste

Betreuender Prüfer: Prof. Dr. Hensel
Zweitgutachter: Prof. Dr. Hußmann

Eingereicht am: 27. Februar 2023

Jonas Kempke

Thema der Arbeit

Entwicklung eines KI gesteuerten Tischkickers unter Verwendung von Reinforcement Learning

Stichworte

Tischkicker, Simulation, KI, Reinforcement Learning, Bildverarbeitung, Mikrocontroller, Motorsteuerung

Kurzzusammenfassung

In dieser Arbeit wird ein KI gesteuerter Tischkicker unter Verwendung von Reinforcement Learning entwickelt. Hierfür wird ein kleiner Tischkicker mit Motoren und Sensoren ausgestattet, sodass sowohl der Zustand des Kickers ermittelt als auch dieser verändert werden kann. Das Training der KI erfolgt innerhalb einer eigenen Simulation, während verschiedene Tests auf dem realen System die Qualität der Lösungen bestimmen. Auf dem entwickelten Tischkicker kann ein unterhaltsames Spiel mit der KI realisiert werden.

Jonas Kempke

Title of Thesis

Development of an AI controlled table football under the use of reinforcement learning

Keywords

Table football, simulation, AI, reinforcement learning, image processing, microcontroller, motor control

Abstract

This thesis deals with the development of an AI controlled table football, which is trained via techniques of reinforcement learning. The implemented system contains of motors and sensors, that are used to controll and observe the table football game. The AI training is carried out in a specially developed software simulation, whereas it is validated on the actual table football system. An entertaining game can be realized on the developed AI table football.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AABB	Axis Aligned Bounding Box
BDC	Brushed Direct Current
BLDC	Brushless Direct Current
ELU	Exponential Linear Unit
FOV	Field Of View
FPS	Frames Per Second
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HAW	Hochschule für Angewandte Wissenschaften
ID	Identification
KI	Künstliche Intelligenz
LED	Light-Emitting Diode
MDP	Markov Decision Process
MSE	Mean Squared Error
MOSFET	Metal Oxide Semiconductor Field-Effect Transistors
MTV	Minimum Translation Vector
OpenCV	Open Source Computer Vision Library
ReLU	Rectified Linear Unit
RFID	Radio-Frequency Identification

RGB	Red Green Blue
RL	Reinforcement Learning
RPM	Revolutions Per Minute
SAT	Separating Axis Theorem
USB	Universal Serial Bus

1 Einleitung

In dieser Arbeit wird die Entwicklung eines KI gesteuerten Tischkickers beschrieben. Hierbei wird ein besonderes Augenmerk auf die Methoden des Reinforcement Learnings gelegt.

1.1 Motivation

In den letzten Jahren schreiten die Entwicklungen auf dem Gebiet des maschinellen Lernens immer weiter voran. Mittlerweile kann eine einzelne künstliche Intelligenz (KI) einen Menschen in einer Vielzahl an Spielen schlagen [9]. Aber nicht nur in Spielen, sondern auch im Alltag finden sich immer mehr Anwendungen, die vom Einsatz künstlicher Intelligenzen profitieren. Es wird prognostiziert, dass künstliche Intelligenzen den Menschen in den nächsten 40 Jahren in fast allen Aufgaben überlegen sein wird [33]. Dies gelte z. B. auch für die Arbeit eines Chirurgen. Allerdings sind derzeit noch gewisse Grenzen gesetzt, die es bis dahin zu lösen gilt. Hierfür werden begeisterte junge Ingenieure benötigt, die sich diesen Herausforderungen stellen. Um interessierten Studierenden an der HAW Hamburg einen praktischen Einstieg ins maschinelle Lernen zu geben, soll in dieser Arbeit eine Plattform für das Training von künstlichen Intelligenzen geschaffen werden.

1.2 Zielsetzung

In dieser Arbeit soll ein automatisierter Tischkicker, der durch eine KI gesteuert wird, entwickelt werden. Hierfür wird insbesondere ein Fokus auf die Techniken des Reinforcement Learnings (RL) gelegt. Diese erlauben erst die Entwicklung von künstlichen Intelligenzen zur Steuerung von komplexen Robotern, die durch überwachtes Lernen nicht effizient trainiert werden können.

Der Tischkicker soll ein Spiel zwischen Mensch und KI möglich machen, sodass eine Seite des Tischs automatisiert und durch Aktoren gesteuert werden muss. Ziel soll es sein, dass die KI den Menschen in einem Spiel besiegen kann. Als Lernplattform sollen aber auch unterschiedliche KI-Modelle implementiert und verglichen werden können.

1.3 Aufbau der Arbeit

Diese Arbeit ist in mehrere Abschnitte unterteilt. Zunächst werden im Kapitel 2 die benötigten Grundlagen für die Entwicklung des automatisierten Tischkickers besprochen. Der aktuelle Stand der Technik wird in Kapitel 3 zusammengefasst und bietet einen Überblick zu bisherigen Arbeiten auf diesem Gebiet. Anschließend wird eine Anforderungsanalyse durchgeführt, um die genaue Zielsetzung der Entwicklung des Tischkickers zu ermitteln. Unter Berücksichtigung der Anforderungen und des Stands der Technik wird in Kapitel 5 das Konzept für den KI gesteuerten Tischkicker entwickelt. Auf der Grundlage des Konzepts kann im nächsten Schritt die Entwicklung erfolgen. Abschließend wird in Kapitel 7 die Evaluation der Ergebnisse vorgenommen und in Kapitel 8 ein Fazit mit Ausblick auf weitere Arbeiten gezogen.

2 Grundlagen

In diesem Kapitel werden die benötigten Grundlagen für die Entwicklung des automatisierten KI Tischkickers dargelegt. Zum einen werden hier die Grundlagen des maschinellen Lernens und zum anderen verschiedene Methoden in der Bildverarbeitung besprochen. Gleichzeitig wird eine Einführung in die 3-dimensionale Kollisionsdetektion gegeben, die im späteren Verlauf innerhalb einer Simulation des Tischkickers benötigt wird.

2.1 Tischkicker

Das Ziel dieser Arbeit ist die Entwicklung eines automatisierten und durch eine KI gesteuerten Tischkickers. Bevor die Entwicklung beginnt, lohnt sich eine kleine Einführung in das Thema.
Ein Tischkicker ist ein dem Volkssport Fußball nachempfundenes Gesellschaftsspiel, bei dem zwei Mannschaften gegeneinander antreten und versuchen den Spielball ins jeweils gegnerische Tor zu schießen. In der Regel sind die Tische wie in Abbildung 2.1 aufgebaut, sodass pro Mannschaft vier Reihen mit insgesamt elf Figuren verfügbar sind, die von einem oder zwei Spielern gesteuert werden. Die einzelnen Stangen haben dabei je zwei Bewegungsfreiheiten, um den Ball abzuwehren bzw. zu schießen. Sie können einerseits gedreht und andererseits hinein- bzw. herausgeschoben werden. In dieser Arbeit werden die Bewegungen als Rotation und Verschiebung referenziert.

Ein Spieler kann verschiedene Techniken einsetzen, die ihm helfen das Spiel zu gewinnen. Hierzu gehört der Pull Shot, bei dem der Ball zunächst seitlich verschoben und anschließend schnell geschossen wird, sodass er an den Spielfiguren der gegnerischen Verteidigung vorbeigeht, bevor der Gegner reagieren kann [36]. Auch ein Kantenschuss ist sehr beliebt. Hierbei wird der Ball leicht seitlich getroffen, sodass er eine schräge Bewegungsrichtung erfährt und z. B. auch über die Berührung einer Bande den Gegner ausspielt. Sogenanntes Trillern, bei dem eine Stange durchgehend in eine Richtung gedreht wird, ist hingegen in den meisten Fällen verboten oder zumindest unerwünscht [37].

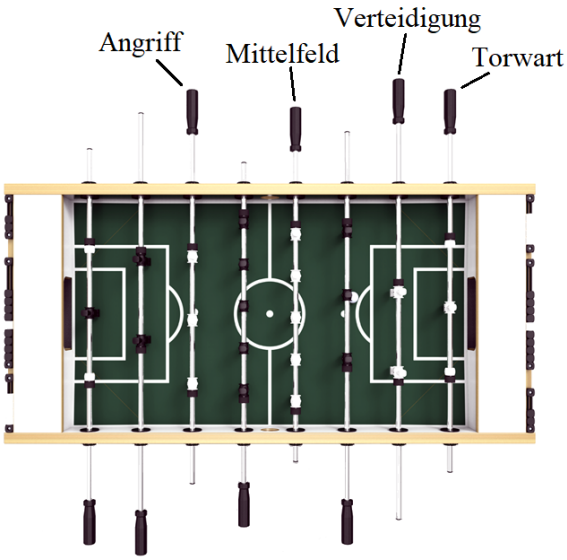


Abbildung 2.1: Aufbau eines typischen Tischkickers [97]

Ein Tischkickerspiel ist eine gute Kombination aus Geschicklichkeit, Geschwindigkeit, Technik und Strategie, wodurch es sich optimal für einen fairen Wettkampf eignet. Aus diesen Gründen ist auch die Steuerung einer Mannschaft über eine künstliche Intelligenz sehr interessant. Hierbei sind hohe Anforderungen an die KI, aber auch an die Steuerung und Sensordatenauswertung des Systems gestellt. Bei der Implementierung auf einem realen System können verschiedene Methoden des Reinforcement Learnings ausprobiert und miteinander verglichen werden. Zudem ist eine direkte Interaktion zwischen Mensch und KI möglich, wodurch mögliche Limitierungen der Algorithmen aufzeigbar werden.

2.2 Künstliche Intelligenz

In dieser Arbeit soll die Steuerung eines Tischkickers von einer KI übernommen werden. Diese wird dabei mittels Verfahren des Reinforcement Learnings trainiert. Für die Entwicklung einer möglichst spielstarken und intelligenten KI muss die Funktionsweise und das Training dieser verstanden werden. Im Folgenden wird ein grober Überblick über die Grundlagen der neuronalen Netze und des Reinforcement Learnings gegeben. Der Abschnitt 2.2.1 ist in Anlehnung an [50] entstanden.

2.2.1 Neuronale Netze

Für das Verständnis des maschinellen Lernens bzw. hier speziell des Reinforcement Learnings lohnt sich zunächst ein Blick auf die neuronalen Netze, die als Grundlage vieler Entscheidungsprozesse von künstlichen Intelligenzen dienen. Neuronalen Netze sind dem Nervensystemen des menschlichen Gehirns nachempfunden [31] und können durch ein Training an eine gewünschte Aufgabe angepasst werden. Sie werden häufig auch zur Klassifizierung von Objekten in Bildern eingesetzt.

Ein künstliches neuronales Netz besteht aus vielen Neuronen, welche mehrere gewichtete Eingänge besitzen können und diese über eine zuvor bestimmte Aktivierungsfunktion auswerten und als Ergebnis ausgeben. Die Neuronen sind wie in Abbildung 2.2 in verschiedenen Schichten aufgebaut. So gibt es immer eine Eingangsschicht, welche zum Beispiel den Pixeln eines Bilds entsprechen kann, und es gibt eine Ausgabeschicht, welche die Ergebnisse des Netzes widerspiegelt. Dazwischen können je nach Komplexität der Aufgabe viele weitere Schichten liegen, welche verborgene Schichten genannt werden. Die Ausgänge der Neuronen einer Schicht werden gewichtet als Eingänge der Neuronen der nächsten Schicht verwendet. Bei zwei oder mehr verborgenen Schichten wird von Deep Learning gesprochen [23].

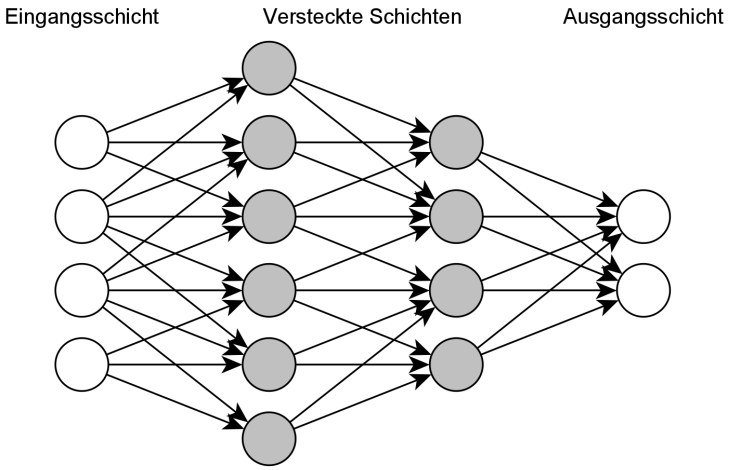


Abbildung 2.2: Aufbau eines sehr einfachen neuronalen Netzes mit zwei versteckten Schichten

Die einzelnen Neuronen lassen sich durch die Formel

$$a \;=\; f(\boldsymbol{w} \cdot \boldsymbol{x} + b) \tag{2.1}$$

beschreiben, wobei $\boldsymbol{w}$ die Gewichte sind, $\boldsymbol{x}$ die Eingänge und b das Bias, welches einem Offset entspricht. Das Ergebnis aus den gewichteten Eingängen mit dem Bias wird in die Aktivierungsfunktion gegeben und ergibt so den Ausgang des Neurons. Beim sogenannten Feedforward werden von der Eingangsschicht ausgehend die Ausgänge der Neuronen der jeweils nachfolgenden Schicht L wie folgt berechnet:

$$\boldsymbol{a}^L \;=\; f(\boldsymbol{W}^L \cdot \boldsymbol{a}^{L-1} + \boldsymbol{b}^L). \tag{2.2}$$

Durch eine Vektortransformationen kann die Berechnung des Ausgangs einer Schicht für beliebig viele Bilder bzw. Eingänge gleichzeitig erfolgen.

Aktivierungsfunktion

Die Aktivierungsfunktion ermöglicht nichtlineare Eigenschaften des Netzes, wodurch erst komplexe Berechnungen möglich werden, da ansonsten stets ein linearer Zusammenhang zwischen Ein- und Ausgangsschicht besteht [75].
Als Aktivierungsfunktion kommen mehrere Typen in Frage. Der einfachste Ansatz ist eine Sprungfunktion, welche einem binären Ausgang entspricht. Für die Verarbeitung ist diese jedoch nicht optimal, da sich der Ausgang des Neurons bei kleinen Änderungen am Eingang schlagartig ändern kann. Gleichzeitig ist das Training über den Gradientenabstieg nicht mehr direkt möglich, da der Gradient der Sprungfunktion Null entspricht [91].

Stattdessen wird deshalb häufig mit der Sigmoid-Funktion

$$\sigma(z) \;=\; \frac{1}{1+e^{-z}} \qquad \text{mit} \quad z \;=\; \boldsymbol{w} \cdot \boldsymbol{x} + b \tag{2.3}$$

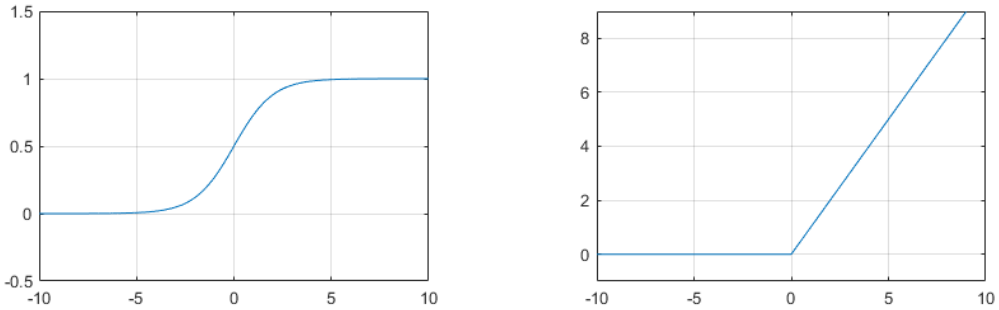
gearbeitet. Diese hat einen ähnlichen Verlauf wie die Sprungfunktion (vergleiche Abbildung 2.3), ist jedoch stetig und ermöglicht so beim Training einen Gradientenabstieg. Die Ableitung der Sigmoid-Funktion

$$\sigma'(z) \;=\; \sigma(z) \cdot (1 - \sigma(z)) \tag{2.4}$$

kann zudem leicht mithilfe der Funktion selbst berechnet werden.
Ein Problem der Sigmoid-Funktion ist jedoch, dass bei größeren absoluten Werten die Funktion in Sättigung geht. Gerade bei tiefen Netzwerken können die Gradienten zu klein werden, sodass sich das Training stark verlangsamen kann [79]. Die Lösung hierfür ist eine Funktion ohne Sättigung wie die ReLU (Rectified Linear Unit) [15]

$$f(z) = \max(0, z). \tag{2.5}$$

Auch diese Funktion lässt sich mit geringem Aufwand differenzieren und noch schneller als die Sigmoid-Funktion berechnen. Allerdings ist die ReLU-Funktion bei negativen Werten in der Sättigung, weshalb es noch weitere Aktivierungsfunktionen gibt, die dieses Problem lösen können, wie die Leaky ReLU oder ELU [91]. Allerdings bringen auch diese Probleme mit sich, da sie z. B. nicht stetig differenzierbar oder nur aufwändig zu berechnen sind. Deshalb wird häufig die ReLU-Aktivierungsfunktion verwendet [22].



(a) Sigmoid (b) ReLU

Abbildung 2.3: Häufig verwendete Aktivierungsfunktionen

Eine weitere wichtige Aktivierungsfunktion ist die Softmax-Funktion

$$\hat{p}_k = \frac{e^{x_k}}{\sum_i^K e^{x_i}}, \tag{2.6}$$

welche oft in der Ausgangsschicht verwendet wird, um das Ergebnis in einer prozentualen Sicherheit zu erhalten. Dabei entspricht K der Anzahl der Neuronen in der entsprechenden Schicht und x_k dem berechneten Eingang des Neurons k, mit welchem die Wahrscheinlichkeit der Zugehörigkeit zur Klasse k berechnet wird. Hohe Werte werden aufgrund der exponentiellen Berechnung stärker gewichtet.

Kostenfunktion

Hierfür muss nach einer Vorhersage des neuronalen Netzes ein Soll-Ist-Vergleich durchgeführt werden, wobei die Kostenfunktion als Maß. Diese vergleichen die Ausgaben der letzten Schicht mit dem gewünschten Ergebnis. Häufig wird die mittlere quadratische Abweichung (MSE) als Kostenfunktion verwendet, welche über

$$C = \frac{1}{n}\sum_{x=1}^{n}(a_x^L - y_x)^2 \tag{2.7}$$

definiert ist. Dabei entspricht n der Anzahl der Neuronen der letzten Schicht, a_x^L deren Ausgängen und y_x dem gewünschten Ergebnis dieser Schicht.
Eine alternative Kostenfunktion, um Lernblockaden zu überwinden, ist die Cross-Entropy [67]

$$C = -\frac{1}{n}\sum_{x=1}^{n} y_x \cdot ln(a_x^L) + (1 - y_x) \cdot ln(1 - a_x^L), \tag{2.8}$$

welche abweichenden Ergebnissen höhere Kosten zuweist als die MSE-Kostenfunktion. Sie wird verwendet, wenn ein binärer Ausgang einer Schicht erwartet wird und das Ergebnis einer prozentualen Sicherheit entspricht. Diese Funktion arbeitet über eine Art Fallunterscheidung, bei der die Kosten an den Neuronen, die 0 entsprechen sollen, durch

$$C_x = -ln(1 - a_x^L) \tag{2.9}$$

berechnet werden, während bei den Neuronen, die 1 ergeben sollen,

$$C_x = -ln(a_x^L) \tag{2.10}$$

verwendet wird.

Gradientenabstieg

Damit das neuronale Netz für eine bestimmte Eingabe auch das zu erwartende Ergebnis liefern kann, muss es trainiert werden. Das Training basiert in der Regel auf dem Gradientenabstieg. Hierbei wird versucht, die Gewichte und Bias so zu verändern, dass die Kosten minimal werden und dementsprechend das erwartete Ergebnis dem tatsächlichen

Ergebnis entspricht. Die Änderung der Gewichte lässt sich aus

$$\Delta w \;=\; -\eta \cdot \nabla_w C \tag{2.11}$$

berechnen, wobei $\nabla_w C$ der nach dem einzelnen Gewicht abgeleiteten Kostenfunktion entspricht und η der Lernrate, welche die Geschwindigkeit des Abstiegs bestimmt. Gleiches gilt für die Änderungen des Bias

$$\Delta b \;=\; -\eta \cdot \nabla_b C. \tag{2.12}$$

Bei komplexen, tiefen Netzwerken ist die Berechnung unter Berücksichtigung aller Gewichte zu rechenaufwendig, sodass stattdessen die Änderungen schichtenweise berechnet werden. Angefangen bei der Ausgabeschicht wird das Netz so unter Berücksichtigung der Ergebnisse der hinteren Schichten und der jeweiligen Aktivierungsfunktion bis zur Eingangsschicht durchlaufen. Dieser Vorgang nennt sich Backpropagation [102].

Da die Kostenfunktion für sehr viele Trainingsdaten aufwendig zu berechnen ist und die Gefahr besteht, dass der Gradientenabstieg in lokalen Minima endet, werden diese in zufällige Batches einer zuvor festgelegten Größe eingeteilt und der Gradientenabstieg für die Batches einzeln berechnet. Dieser Vorgang wird für den gesamten Datensatz durchgeführt und wird als Epoche bezeichnet.

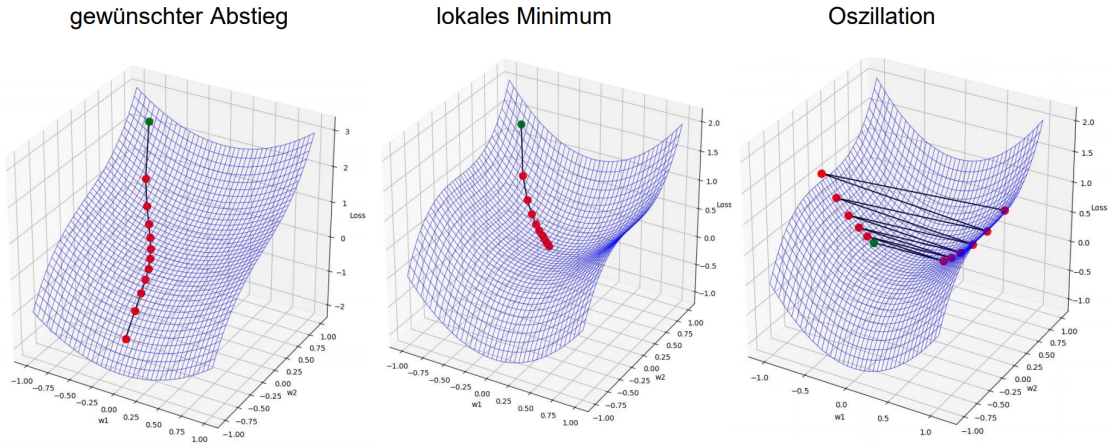


Abbildung 2.4: Gradientenabstieg mit gewünschtem Verlauf, mit zu kleiner und mit zu großer Lernrate [24]

Durch den Gradientenabstieg können so über mehrere Epochen die Gewichte und das Bias angepasst werden, sodass die Kosten minimal werden und somit das gewünschte

Ergebnis geliefert wird. Die Lernrate muss jedoch auf das Netz angepasst werden, da Oszillation entstehen kann oder sogar globale Minima übersprungen werden, wenn diese Rate zu groß gewählt ist. Andersherum kann bei einer zu kleinen Lernrate das Training wesentlich länger dauern oder der Gradientenabstieg in lokalen Minima enden. Abbildung 2.4 zeigt die unterschiedlichen Verläufe.

Um den Gradientenabstieg schneller zu gestalten und kleine lokalen Minima zu umgehen, kann ein Momentum eingesetzt werden, welches die Bewegung des Gradientenabstiegs beibehält. Dabei wird eine Geschwindigkeit

$$m' \;=\; \beta \cdot m + \eta \cdot \nabla_w C \tag{2.13}$$

errechnet, welcher den Gradienten als Beschleunigung interpretiert. Die Gewichte werden dann mit

$$w' \;=\; w + m' \tag{2.14}$$

berechnet. Das Moment β liegt dabei zwischen 0, was dem normalen Gradientenabstieg entspricht, und 1, wodurch eine reibungsfreie Bewegung simuliert wird.

Faltende Netze

Beim Aufbau der Netze gibt es unterschiedliche Ansätze. Eine mögliche Struktur stellen sogenannte Dense Layer dar, welche alle Neuronen der vorherigen Schicht mit allen Neuronen der nächsten Schicht verbinden (siehe z. B. Abbildung 2.2). Dieser Ansatz ist bei großen und komplexen Netzen jedoch nicht optimal, da

$$n_{\boldsymbol{W}^L} \;=\; n_{N^L} \cdot n_{N^{L-1}} \tag{2.15}$$

Gewichte sowie n_{N^L} Bias für eine Schicht L benötigt werden. Des Weiteren korrelieren nicht alle Pixel eines Bilds miteinander. So ist der gegenseitige Einfluss von zwei Pixeln, die sehr weit auseinander liegen, meist gering.
Eine Lösung hierfür sind faltende Schichten, die mit einem kleinen Filterkern (häufig 3×3, 5×5 oder 7×7) die vorige Schicht durchlaufen [77]. Dabei können die Filterkerne auch eine dritte Dimension besitzen, wodurch zum Beispiel Farbkanäle berücksichtigt werden können. Dieser Vorgang kann mit verschiedenen Filtern durchgeführt werden, sodass eine Vielzahl von sogenannten Feature Maps entstehen, die zusammen eine Schicht im Netz

ergeben. Auch die Anzahl der Bias kann so verringert werden, da lediglich eines pro Feature Map benötigt wird. Die Größe einer Feature Map ist abhängig von verschiedenen Einstellungen. Die Schrittweite der Filterkerne dient als Dividend der Dimension. Die Ränder sind abhängig vom sogenannten Padding, ohne welches sich die Dimensionen um maximal der Größe des Filterkerns verringern kann. Um bei faltenden Netzen die Daten zu reduzieren, werden zusätzlich sogenannte Pooling Layer eingesetzt, welche z. B. über einen 2×2 Filterkern und einer Schrittweite von zwei die vorige Schicht in ein Viertel ihrer Größe reduzieren können [77]. Dabei können die Werte im Filterkern gemittelt oder auch nur der maximale Wert verwendet werden.
Um am Ende eines Netzes aus den zweidimensionalen Schichten, in denen sich die extrahierten Merkmale befinden, eine Vorhersage für die verschiedenen Klassen zu treffen, werden über ein sogenanntes Flatten Layer diese Schichten in eine eindimensionale Schicht umgewandelt. Abschließend werden meist wenige Dense Layer eingesetzt, um die Klassifizierung zu übernehmen. U dabei robust gegenüber kleinen Änderungen zu sein, werden hier Dropout Layer eingebaut, welche einen zuvor festgelegten Anteil ihrer Neuronen zufällig ausschalten. Dadurch werden dieselben Funktionen von mehreren Neuronen übernommen. Bei der Klassifikation werden dann alle Neuronen verwendet. Die Struktur eines solchen Netzes ist in Abbildung 2.5 festgehalten.

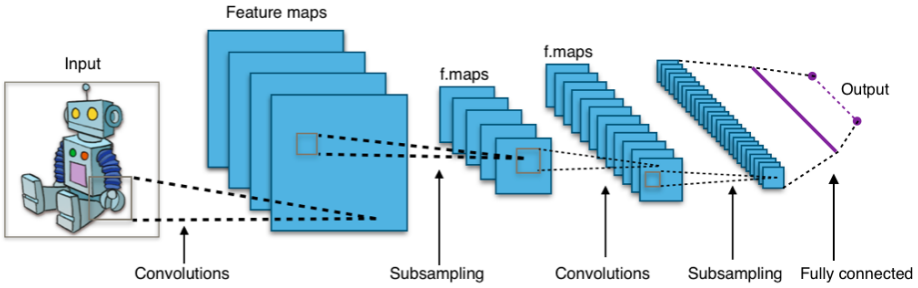


Abbildung 2.5: Exemplarischer Aufbau eines faltenden Netzes [103]

2.2.2 Reinforcement Learning

Mit den beschriebenen Methoden lassen sich neuronale Netze entwickeln, die eine Klassifizierung von generellen Daten oder Bildern übernehmen. Bei Eingabe der Daten wird in diesen Fällen das Ergebnis der Ausgangsschicht beobachtet und interpretiert. So kann das Netz z. B. eine Klassifizierung von mehreren Objekten vornehmen und anschließend anhand der vom Menschen zuvor zugewiesenen Klasse die eigene Vorhersage bewerten und entsprechend seine Gewichte anpassen.

Dieser Ansatz kommt jedoch an seine Grenze, wenn keine einfache Klassifizierung eines Bilds oder Zustands vorgenommen werden soll. Eine KI, die wie in dieser Arbeit ein Tischkickerspiel erlernen soll, kann die verschiedenen Zustände, die im Spiel auftreten können, aufgrund ihrer extrem hohen Anzahl nur schwierig unterscheiden. Bei einem Training müsste jeder Zustand mit einer passenden Aktion vorher angegeben werden. In diesem Fall könnte auch direkt eine Programmierung über klassische Algorithmen erfolgen, die wahrscheinlich besser und schneller ist, da die KI maximal so gut werden kann, wie die eintrainierten Aktionen. Hierzu kommt noch, dass die Bewegungsrichtung und Geschwindigkeit des Balls nicht aus einem einzelnen Bild hervorgehen.
Die Lösung diese Problems ist das Training über Reinforcement Learning (RL). Das Konzept beim RL ist, dass ein Agent (entspricht hier der KI) mithilfe seiner eintrainierten Taktik eine Aktion auswählt, die anschließend ausgeführt wird [34]. Dadurch beeinflusst er die (Spiel-)Umgebung, aus welcher der aktuelle Zustand ermittelt und an den Agenten ausgegeben wird. Damit der Agent trainiert werden kann, muss er wissen wie gut seine letzte Handlung war. Hierfür wird in der Umgebung eine Belohnung anhand des beobachteten Zustands bestimmt. Je höher diese Belohnung ist, desto besser war die Aktion des Agenten. So kann auch eine schlechte Aktion mit einer niedrigen Belohnung bestraft werden. Die Belohnung bzw. Bestrafung und der aktuelle Zustand werden dem Agenten mitgeteilt. Anhand des erhaltenen Zustands und der Belohnung muss der Agent seine Taktik anpassen, um langfristig eine möglichst hohe Belohnung zu erhalten. Abbildung 2.6 zeigt den Ablauf von RL.

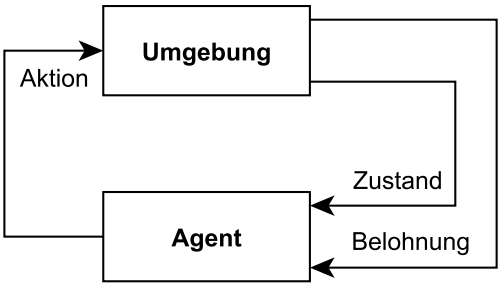


Abbildung 2.6: Konzept des Reinforcement Learnings nach [52]

Für RL muss somit eine Umgebung geschaffen werden, mit der ein Agent interagieren kann und von der er den Zustand und die Belohnung erhält. Gleichzeitig muss ein Agent generiert werden, der eigene Grundregeln festlegt, anhand welcher er eine Aktion auswählt.

Markov Decision Process

Die Grundlage für das Training der meisten RL Algorithmen bildet der sogenannte Markov Decision Process (MDP). Dieser dient als mathematische Beschreibung des Zustandsübergangs in einem Tupel ($\boldsymbol{S}$, $\boldsymbol{P}$, $\boldsymbol{R}$, A) [49].

- $\boldsymbol{S}$ entspricht allen möglichen Zuständen des Systems.
- A entspricht allen möglichen Aktionen.
- $\boldsymbol{P}$ entspricht der Wahrscheinlichkeit für den Übergang aus dem aktuellen Zustand s mit der Aktion a in den nächsten Zustand s'.
- $\boldsymbol{R}$ entspricht der zu erwartenden Belohnung beim Übergang aus dem aktuellen Zustand s in den nächsten Zustand s'.

Der MDP ist ein zeitdiskreter Prozess für die Beschreibung zufallsbedingter Übergänge eines Systems in den nächsten Zustand bei einer gegebenen Aktion. Der Zustand eines Systems muss dabei immer eindeutig sein, sodass alle Übergänge in die nächsten Zustände ausschließlich von der aktuellen Beobachtung und Aktion und nicht von früheren Ereignissen abhängig sind [108].
Neben der ausgewählten Aktion kann ein Zustandsübergang auch durch stochastische Prozesse beeinflusst sein, wodurch der nächste Zustand bei gleichen Bedingungen nicht immer gleich sein wird. Dies erlaubt z. B. auch die Berücksichtigung von äußeren Einflüssen auf das System.
Für das Training einer KI ist vor allem die zu erwartende Belohnung beim Übergang interessant, da mit dieser eigene Grundregeln $\boldsymbol{\pi}$ (Policy) für das Maximieren der Belohnung und somit für das Lösen der gewünschten Aufgabe festgelegt werden können. Die Grundregeln $\boldsymbol{\pi}$ können beispielsweise so gewählt sein, dass immer die Aktion mit der größten zu erwartenden Belohnung ausgewählt werden. In vielen Fällen kann es jedoch langfristig zu höheren Belohnungen kommen, wenn kurzfristig Entscheidungen getroffen werden, die zunächst eine geringere Belohnung versprechen. Deshalb ist die Ermittlung der richtigen Policy wichtig, was durch verschiedene Einstellungen ermöglicht wird. Beispielsweise kann ein sogenannter Discount Factor γ angegeben werden, der bestimmt in welchem Maße in den nächsten Schritten folgende Belohnungen bei der Auswahl der Aktion berücksichtigt werden sollen [63].

Q-Learning

Ein wichtiger Algorithmus für RL ist Q-Learning. Hierbei wird die $Q(s,\ a)$-Funktion aufgestellt, welche für jeden Zustand s und die möglichen Aktionen a die zu erwartende Belohnung berechnet. Initial ist diese Funktion für alle Kombinationen 0. Beim Durchlaufen der Zustände werden die Werte der Funktion über die Formel

$$Q(s,\ a) \ \leftarrow \ Q(s,\ a) + \eta[r + \gamma V(s') - Q(s,\ a)] \tag{2.16}$$

aktualisiert [34]. Die Belohnung für die ausgewählte Aktion a ist als r festgelegt. Weiter entspricht $V(s')$ je nach Definition der maximal bzw. der mittleren zu erwartenden Belohnung vom aktuellen Zustand s bei einer gewählten Aktion a [55]. Über den Discount Factor γ kann der Einfluss der folgenden Zustände angepasst werden. Der aktuelle Wert $Q(s,\ a)$ wird von der Summe der aktuellen und folgenden zu erwartenden Belohnungen subtrahiert, damit eine Differenz zwischen dem tatsächlichen und dem aktuellen Wert $Q(s,\ a)$ gebildet wird. Über die Lernrate kann diese Differenz mit dem aktuellen Wert addiert werden, sodass eine langsame Anpassung an den tatsächlichen Wert erfolgt. Die Lernrate ist hierbei nötig, da auch $V(s')$ in der Regel noch fehlerbehaftet ist und es sonst wie in Abschnitt 2.2.1 beschrieben zu Oszillation kommen kann, sodass die Werte nicht konvergieren. Wenn $r + \gamma V(s') \ = \ Q(s,\ a)$ ist, wird der Wert von $Q(s,\ a)$ nicht verändert, sodass die Funktion nach einigen Anpassungen optimalerweise konvergiert.

Die Auswahl der auszuführenden Aktion kann nach der Policy $\boldsymbol{\pi}$ so ausgelegt sein, dass immer die Aktion mit der maximal zu erwartenden Belohnung $\max_a Q(s,\ a)$ ausgewählt wird. Ein Problem hierbei ist jedoch, dass zunächst ermittelt werden muss, welche Zustandsübergänge die höchste Belohnung geben. Hierfür müssen bei der Auswahl der Aktionen auch zufällige Entscheidungen getroffen werden, da sonst ein großer Teil des Zustandsraums nie untersucht wird. Um diesen Problem entgegenzuwirken, kann ein Wert ϵ festgelegt werden, der die Wahrscheinlichkeit für die Wahl einer zufälligen Aktion bestimmt [63]. Beim Training muss dieser Wert entsprechend hoch sein, damit alle Zustände durchlaufen werden. Für den Einsatz im fertigen Produkt sollte ϵ eher klein sein, damit das erlernte Wissen auch angewendet wird. Solange der Wert von $\epsilon \neq 0$ ist, untersucht der Algorithmus noch weitere Zustände und kann sich so stetig verbessern.

Damit die optimalen Grundregeln gefunden werden können, müssen sehr viele Zustandsübergänge beobachtet werden. Die meisten Vorgänge sind jedoch kontinuierlich, wobei die Speicherung im Computer jedoch diskret, aber hier nicht weiter zu beachten ist,

und durch verschiedene Parameter ergeben sich extrem viele Zustände. Aus genannten Gründen sowie dem Fakt, dass auch die Folgezustände von äußerlichen Einflüssen abhängig sind, können in der Regel nicht alle Zustandsübergänge eines Systems überprüft werden. Deshalb müssen z. B. Zustandsräume diskretisiert werden, sodass eine endliche Menge an Zuständen entsteht [34]. Selbst mit dieser Maßnahme sind nicht immer alle Zustandsübergänge überprüfbar und in der Q-Funktion, die eigentlich einer großen Tabelle entspricht [106], speicherbar.

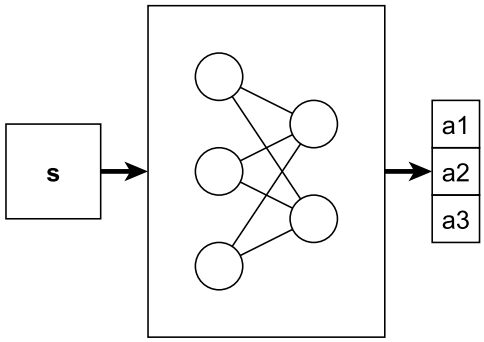


Abbildung 2.7: Neuronales Netzwerk als $Q(s,\ a)$-Funktion nach [34]

In der Regel wird eine solche Aufstellung für jeden einzelnen Fall nicht benötigt, da ähnliche Zustände meist die gleichen Aktionen erfordern. Es gilt also nach Möglichkeit die Zustände zu generalisieren. Die Lösung hierfür sind Deep Q-Networks (DQN), die über neuronale Netze funktionieren [64]. Als Eingang nutzen sie wie in Abbildung 2.7 den aktuellen Zustand und als Ausgang geben sie die jeweilige Aktion aus. Dazwischen befinden sich die versteckten Schichten. Ein DQN nutzt die Differenz zwischen der eigentlichen und aktuell zu erwartenden Belohnung

$$\Delta Q \ = \ r + \gamma V(s') - Q(s,\ a) \tag{2.17}$$

in der Kostenfunktion für den Gradientenabstieg [99], da die Kosten gegen 0 konvergieren, wenn die optimale Lösung gefunden wird [34].
Weiter wird auch ein Puffer eingesetzt, in welchem bekannte Daten im $(s,\ a,\ r,\ s')$-Format gespeichert werden. Für das Training können dann diese Daten in zufälligen Gruppen ausgewählt werden, was eine Verbesserung der Stabilität der neuronalen Netze bewirkt [76].

2.3 Bildverarbeitung

Für die Auswertung des Zustands des Tischkickers werden verschiedene Algorithmen entwickelt. Diese verwenden unterschiedliche Verfahren zur Verarbeitung von Bildern und zur Nachverarbeitung der gewonnenen Daten. Im Folgenden werden diese Verfahren und wichtige Bibliotheken vorgestellt.

2.3.1 OpenCV

OpenCV (Open Source Computer Vision Library) [74] ist eine Programmbibliothek für die Bildverarbeitung und für maschinelles Lernen. Sie bietet eine umfangreiche Sammlung von verschiedenen gängigen Algorithmen zur Anwendung in der Bild und Videoverarbeitung. Hierbei stehen auch Methoden für die Objekterkennung und -verfolgung zur Verfügung. Die implementierten Algorithmen sind hoch optimiert und erlauben die einfache und schnelle Ausführung komplexer Verarbeitungsschritte.
OpenCV ist in C++ geschrieben und auch in Programmiersprachen wie Python und Java verfügbar.

2.3.2 Otsu-Binarisierung

In der klassischen Bildverarbeitung wird häufig eine Segmentierung eines Bilds in verschiedene Bereiche vorgenommen, um z. B. Formen und so auch Objekte in den Bildern erkennen zu können. Hierbei können beispielsweise helle Objekte vom dunklen Hintergrund unterschieden werden. Die Segmentierung kann in diesem Fall bei einem Graustufenbild durch ein Schwellwertverfahren erfolgen, welches Pixel mit größeren Werten weiß einzeichnet, während Pixel mit kleineren Werten als der Schwellwert schwarz werden. Ein Problem hierbei kann das Finden des optimalen Schwellwerts sein, da sich dieser Wert je nach Helligkeit des Bilds verändern kann.
Hier kommt die Otsu-Binarisierung ins Spiel, welche die Graustufen des Bilds so in zwei Klassen eingeteilt, dass die Streuung innerhalb der Klassen minimal und zwischen den beiden Klassen so groß wie möglich ist [69]. Einfacher beschrieben bedeutet dies, dass in einem Histogramm wie in Abbildung 2.8 die zwei Bereiche der unterschiedlichen Segmente möglichst passend getrennt werden.

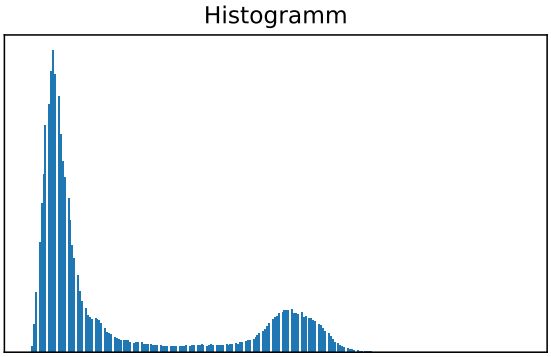


Abbildung 2.8: Beispiel eines Histogramms eines Bilds mit einem hellen Objekt auf dunklem Hintergrund

Die mathematische Beschreibung des Algorithmus wird im Folgenden in Anlehnung an [73] und [50] beschrieben.
Für die Grauwerte $0 \leq g \leq G$ mit $G = 255$ als Maximalwert kann die Wahrscheinlichkeit der Zugehörigkeit zu den Klassen K_1 und K_2, welche durch den Schwellwert t getrennt sind, durch

$$P_0(t) = \sum_{g=0}^{t} p(g) \quad \text{und} \quad P_1(t) = \sum_{g=t+1}^{G} p(g) \tag{2.18}$$

berechnet werden, wobei $p(g)$ der Wahrscheinlichkeit des jeweiligen Grauwertes entspricht.
Mit den mittleren Grauwerten $\bar{g_0}$ und $\bar{g_1}$ können die Varianzen innerhalb der einzelnen Klassen

$$\sigma_0^2(t) = \sum_{g=0}^{t} (g - \bar{g_0})^2 \cdot p(g) \quad \text{und} \quad \sigma_1^2(t) = \sum_{g=t+1}^{G} (g - \bar{g_1})^2 \cdot p(g) \tag{2.19}$$

berechnet werden, aus welchen sich die Gesamtvarianz innerhalb der Klassen

$$\sigma_{in}^2(t) = P_0(t) \cdot \sigma_0^2(t) + P_1(t) \cdot \sigma_1^2(t) \tag{2.20}$$

zusammensetzt. Die Varianz zwischen den Klassen kann durch

$$\sigma_{zw}^2(t) = P_0(t) \cdot (\bar{g_0} - \bar{g})^2 + P_1(t) \cdot (\bar{g_1} - \bar{g})^2 \tag{2.21}$$

berechnet werden, wobei $\bar{g}$ dem mittleren Grauwert des gesamten Bilds entspricht.
Der Schwellwert t ist für ein Bild optimal, wenn der Quotient

$$Q(t) = \frac{\sigma_{zw}^2(t)}{\sigma_{in}^2(t)} \tag{2.22}$$

maximal wird.
So lässt sich effizient ein an das jeweilige Bild angepasster Schwellwert errechnen, der dann für das Schwellwertverfahren eingesetzt wird.

2.3.3 Kalman Filter

Für eine spätere Auswertung der Position des Balls über die Daten einer Kamera, kann die Verwendung von Filtern für bessere Ergebnisse sorgen [82]. Im Folgenden wird der Kalman-Filter vorgestellt, der ein potentieller Lösungsansatz hierfür ist.

Der Kalman-Filter [47] ist ein weit verbreiteter Algorithmus zur Filterung und Vorhersage von Messdaten. Er berücksichtigt dabei vorangegangene Messungen und bildet so den Zustand des Systems ab. Gleichzeitig wird auch die Messgenauigkeit mit einbezogen, um der eigenen Vorhersage eine Art Genauigkeit zuzuweisen.

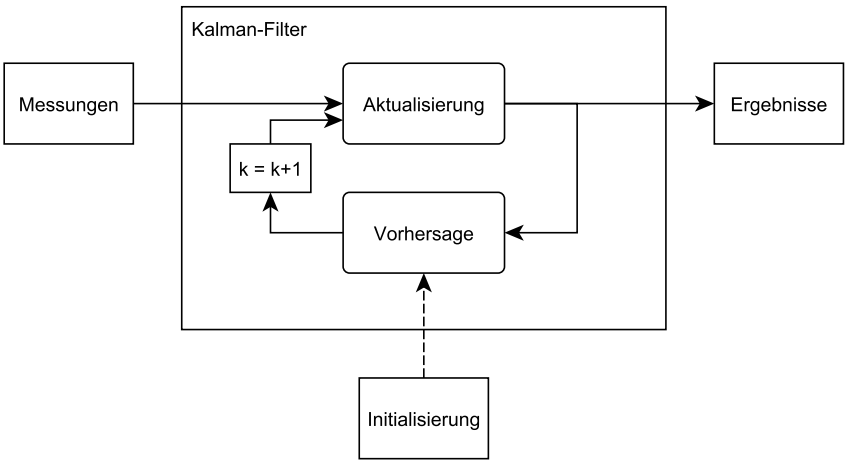


Abbildung 2.9: Grober Ablaufplan des Kalman-Filters nach [12]

Die Berechnung des Filters besteht im Grunde aus zwei Schritten. Einerseits wird eine Messung vorgenommen, mit deren Ergebnis eine Anpassung und Korrektur des eigenen

Zustands durchgeführt wird. Andererseits wird im Wechsel mit der Messung eine Vorhersage gemacht, welche eine Schätzung des nächsten Zustands berechnet.
Der grobe Ablaufplan des Kalman-Filters ist in Abbildung 2.9 zu sehen. Nach der Initialisierung wird mit einer Vorhersage gestartet und anschließend erst das System an die erste Messung angepasst. Der Ausgang des Filters entspricht in der Regel der korrigierten Messung, wobei auch auf die Systemeigenschaften zugegriffen werden kann.

Tabelle 2.1: Beschreibung der Parameter des Kalman-Filters

Name	Dimensionen	Beschreibung
k	1	Aktuelle Abtastung zum Zeitpunkt t_k.
n	1	Anzahl der Werte im Systemzustand.
m	1	Anzahl der Messwerte.
$\boldsymbol{z}_k$	$(m \times 1)$	Aktuelle Messwerte der Abtastung k.
$\boldsymbol{y}_k$	$(m \times 1)$	Korrigierte Messwerte nach der Filterung.
$\boldsymbol{x}_{k,k-1}$	$(n \times 1)$	Systemzustand vor der Berücksichtigung der aktuellen Abtastung $\boldsymbol{z}_k$ (a-priori).
$\boldsymbol{P}_{k,k-1}$	$(n \times n)$	Kovarianzmatrix der Genauigkeit von $\boldsymbol{x}_{k,k-1}$, also vor der Berücksichtigung der aktuellen Abtastung $\boldsymbol{z}_k$ (a-priori).
$\boldsymbol{x}_{k,k}$	$(n \times 1)$	Systemzustand nach der Berücksichtigung und Einberechnung der Abtastung $\boldsymbol{z}_k$ (a-posteriori).
$\boldsymbol{P}_{k,k}$	$(n \times n)$	Kovarianzmatrix der Genauigkeit von $\boldsymbol{x}_{k,k}$, also nach der Berücksichtigung der aktuellen Abtastung $\boldsymbol{z}_k$ (a-posteriori).
$\boldsymbol{F}$	$(n \times n)$	Übergangsmatrix zur Vorhersage des nächsten Systemzustands $\boldsymbol{x}_{k,k+1}$ aus dem aktuellen Zustand $\boldsymbol{x}_{k,k}$.
$\boldsymbol{H}$	$(m \times n)$	Beobachtungsmatrix für die Berechnung des aktuellen korrigierten Messwerts $\boldsymbol{y}_k$ über den Systemzustand $\boldsymbol{x}_{k,k}$.
$\boldsymbol{R}$	$(m \times m)$	Messrauschkovarianzmatrix zur Bestimmung der Genauigkeit der Messwerte.
$\boldsymbol{Q}$	$(n \times n)$	Prozessrauschen für die Beschreibung natürlicher Abweichungen im System.
$\boldsymbol{K}_k$	$(n \times m)$	Kalman-Gain-Matrix zur Korrektur des aktuellen Systemzustands $\boldsymbol{z}_k$ aus dem letzten Zustand $\boldsymbol{x}_{k,k-1}$.
$\boldsymbol{u}_k$	$(l \times 1)$	Dauerhafte und unvermeidbare Störung auf das System (wie z. B. Gravitation).
$\boldsymbol{B}$	$(n \times l)$	Einfluss der Störung $\boldsymbol{u}$ auf den Systemzustand.
$\boldsymbol{I}$	$(n \times n)$	Einheitsmatrix dessen Hauptdiagonale 1 ist, während die restlichen Werte 0 sind.

Die Zustandsraumdarstellung ist die Grundlage des Kalman-Filters. Diese ermöglicht einem Benutzer die Berechnung des Ausgangs eines Übertragungssystems anhand seiner internen Zustände und der Eingänge. Die Berechnung des Systems ist effizient über verschiedene Matrizen möglich. Im Tabelle 2.1 werden zunächst die verschiedenen Vektoren und Matrizen beschrieben, die für die Berechnung eines Kalman-Filters notwendig sind.

Die Berechnung der gefilterten Ausgangswerte erfolgt aus der Multiplikation des Systemzustands $\boldsymbol{x}_{k,k}$ mit der Beobachtungsmatrix $\boldsymbol{H}$ über

$$\boldsymbol{y}_k \;=\; \boldsymbol{H} \cdot \boldsymbol{x}_{k,k}. \tag{2.23}$$

Hierfür wird zunächst jedoch der aktuelle Systemzustand benötigt. Dieser wird wie bereits angedeutet in zwei Schritten berechnet [53].

Im ersten Schritt wird eine Vorhersage für den nächsten Zustand gemacht. Dies kann einfach mittels des Zustands und der Übergangsmatrix $\boldsymbol{F}$ durch

$$\boldsymbol{x}_{k,k-1} \;=\; \boldsymbol{F} \cdot \boldsymbol{x}_{k-1,k} + \boldsymbol{B} \cdot \boldsymbol{u}_k \tag{2.24}$$

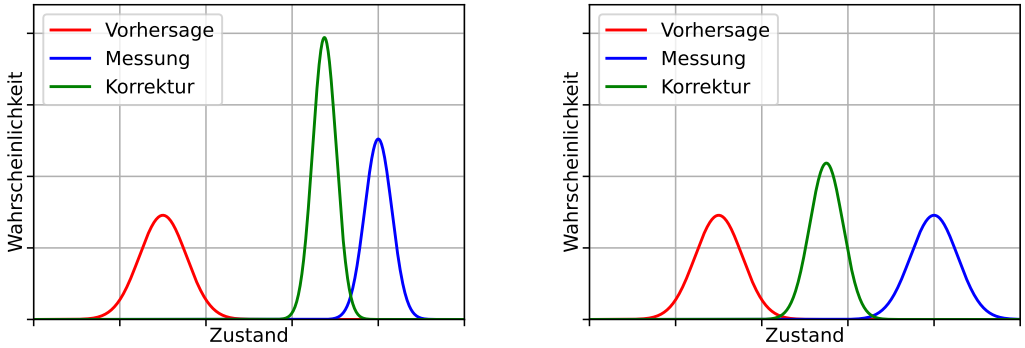
erfolgen. Die Störung $\boldsymbol{u}_k$ wird hierbei mittels Kontrollmatrix auf das System angewendet. Neben der Berechnung des nächsten Zustands muss im ersten Schritt auch die Unsicherheit des Zustands berechnet werden. Diese lässt sich durch die Formel

$$\boldsymbol{P}_{k,k-1} \;=\; \boldsymbol{F}\boldsymbol{P}_{k-1,k}\boldsymbol{F}^T + \boldsymbol{Q} \tag{2.25}$$

berechnen. Dabei wird die Unsicherheit nach der letzten Korrektur berücksichtigt und mit der Übergangsmatrix verrechnet. Bei diesem Schritt erhöht sich stets die Unsicherheit [12]. Zudem kann das Prozessrauschen addiert werden, welches dabei hilft besser auf natürliche Änderungen des Systems zu reagieren.

Der zweite Schritt besteht aus der Messung des tatsächlichen Zustands und der anschließenden Korrektur der Vorhersage. Die Korrektur berücksichtigt dabei die Unsicherheit der Vorhersage $\boldsymbol{P}_{k,k-1}$ und der Messungen $\boldsymbol{R}$. Die Unsicherheit kann als Varianz des jeweiligen Zustands betrachtet werden. Je größer sie ist, desto mehr kann der tatsächliche Status vom aktuellen System abweichen. Der berechnete Zustand entspricht immer nur dem Erwartungswert dieser Normalverteilung [10].
Bei der Korrektur der gemachten Vorhersage ist das Ziel, einen Zustand zu finden, der

eine möglichst geringe Unsicherheit besitzt. Dieser Zustand liegt zwischen Messung und Vorhersage. Je höher die Messgenauigkeit $\boldsymbol{R}$, desto näher liegt der Erwartungswert des aktuellen Zustands am Messwert. Wenn wie in Abbildung 2.10 die Messgenauigkeit gering ist, kann der tatsächliche Zustand näher an der Schätzung liegen. Dabei verliert dieser jedoch auch an Genauigkeit.



(a) Hohe Messgenauigkeit (b) Niedrige Messgenauigkeit

Abbildung 2.10: Korrektur des Zustands des Kalman-Filters

Die Berechnung des korrigierten Zustands ist über

$$\boldsymbol{x}_{k,k} = \boldsymbol{x}_{k,k-1} + \boldsymbol{K}_k \cdot (\boldsymbol{z}_k - \boldsymbol{H} \cdot \boldsymbol{x}_{k,k-1}) \tag{2.26}$$

möglich. Dabei wird die letzte Schätzung $\boldsymbol{x}_{k,k-1}$ mit der Differenz der Ergebnisse der Schätzung und der gemessenen Werte addiert. Hierbei wird die Differenz jedoch zunächst mithilfe des Kalman-Gains

$$\boldsymbol{K}_k = \boldsymbol{P}_{k,k-1}\boldsymbol{H}^T(\boldsymbol{P}_{k,k-1}\boldsymbol{H}^T + \boldsymbol{R}_n)^{-1} \tag{2.27}$$

verrechnet. Dieser kann Werte zwischen $0 \leq \boldsymbol{K}_k \leq 1$ annehmen. Ein geringer Gain entspricht einer geringen und ein hoher Gain einer hohen Messgenauigkeit [12].
Zuletzt muss noch die Kovarianzmatrix

$$\boldsymbol{P}_{k,k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_{k,k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})^T + \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{K}_k^T \tag{2.28}$$

des korrigierten Zustands berechnet werden.

Die beiden Schritte können fortan iterativ durchlaufen werden. So kann der Kalman-Filter bei guter Abstimmung an das System eine Verbesserung der Messdaten erzeugen und z. B. einfaches Messrauschen filtern. Gleichzeitig kann der Filter auch verwendet werden, um fehlende Messungen durch eine zusätzliche Schätzung auszugleichen. So kann beispielsweise die Position eines Objekts bei einer fehlgeschlagenen Detektion aus Kameradaten geschätzt werden, sodass weitere Verarbeitungsschritte mit dieser Vorhersage arbeiten können. Ebenfalls wäre eine Verdoppelung der Abtastwerte über eine zusätzliche Schätzung möglich. Hierbei muss jedoch immer bedacht werden, dass die Genauigkeit des gefilterten Zustands abnimmt, je häufiger eine Bestimmung des Zustands ohne Messung durchgeführt wird.

2.4 Kollisiondetektion

In der Simulation eines Tischkickers, welche im späteren Verlauf dieser Arbeit für das Training der KI genutzt werden kann, wird eine Kollisionsdetektion benötigt. Wenn der Ball beispielsweise eine Figur berührt, muss ein Impuls auf diesen übertragen werden, sodass er geschossen oder gehalten werden kann. Hierfür muss jedoch erst überprüft werden, ob der Ball irgendein Objekt berührt. Im Folgenden werden kurz zwei gängige Verfahren beschrieben, die diese Detektion durchführen.

2.4.1 AABB

Das erste Verfahren ermittelt eine Kollision durch die den Achsen angepassten Bounding Boxen der Objekte. Sie beschreibt das kleinstmögliche Rechteck um einen Körper entlang der Achsen des Koordinatensystems [94]. Eine solche Bounding Box ist in Abbildung 2.11 in rot zu sehen.

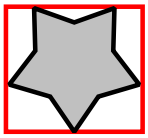
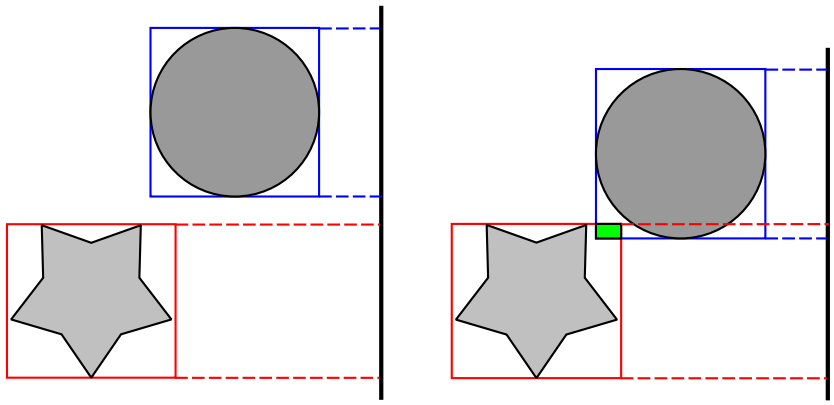


Abbildung 2.11: Achsenangepasste Bounding Box um einen Körper

Die Überprüfung der Kollision zweier Objekte kann alleine an den Rechtecken ausgemacht werden. Wenn sich in einer der Dimensionen die Bounding Boxen der zwei Körper nicht überschneiden, liegt keine Kollision vor. Wenn hingegen in keiner der vorgegebenen Dimensionen eine Trennung der Boxen möglich ist, berühren sich die Objekte [80]. Dabei kann durch die grobe Annäherung wie in Abbildung 2.12 eine Kollision erkannt werden, obwohl die tatsächlichen Objekte sich gar nicht berühren.



(a) Keine Berührung wird erkannt (b) Berührung wird erkannt

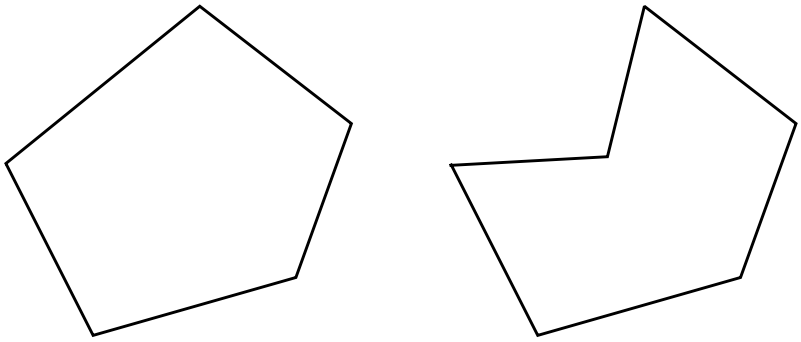
Abbildung 2.12: AABB Kollisionsdetektion zweier Körper

Das Verfahren über AABB ist zwar sehr einfach und schnell zu berechnen, jedoch sind die Ergebnisse eher ungenau. Deshalb ist diese Kollisionsdetektion nicht besonders gut geeignet für eine genaue Simulation. Häufig wird der Ansatz über AABBs allerdings eingesetzt, um im ersten Schritt eine Auswahl von Objekten zu treffen, die sich potentiell berühren [62].

2.4.2 SAT

Ein zweiter weit verbreiteter Ansatz zur Detektion von Kollisionen zweier Objekte im 3-dimensionalen Raum ist über das Separating Axis Theorem (SAT). Dabei wird überprüft, ob zwei Objekte durch eine beliebige lineare Achse trennbar sind. Wenn dies der Fall ist, liegt keine Kollision vor, ansonsten berühren sich die Objekte.
Durch die Trennung über eine lineare Achse ist die Detektion nur für konvexe Körper möglich. Ein konvexer Körper hat ausschließlich Innenwinkel $\alpha_c \leq 180°$, wodurch eine ge-

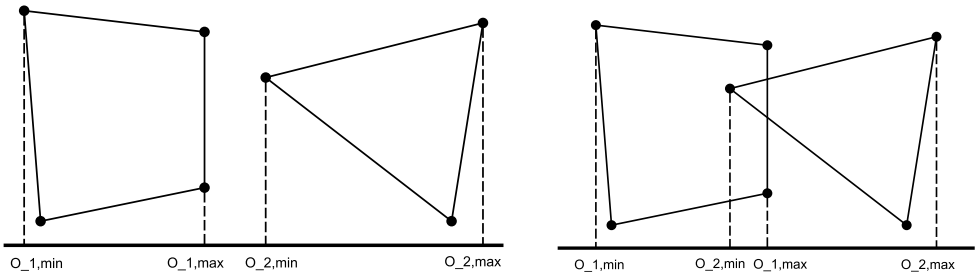
rade Linie durch einen Körper maximal zweimal seine Hülle durchstößt [58]. Ein Beispiel für einen konvexen und einen konkaven Körper ist jeweils in Abbildung 2.13 zu sehen.



(a) Konvexer Körper (b) Konkaver Körper

Abbildung 2.13: Vergleich von konvexen und konkaven Körpern

Die Kollisionsdetektion mittels SAT basiert auf dem Vergleich der Position der Eckpunkte eines Körpers mit dem zweiten Objekt. Wenn mindestens einer der Eckpunkte eines Körpers innerhalb des Zweiten liegt, berühren sie sich. Anderenfalls ist eine Trennung durch eine lineare Achse möglich [29]. Abbildung 2.14 zeigt den Zusammenhang der Position der Punkte zweier Objekte bei einer Kollision.



(a) Keine Überlappung der Eckpunkte (b) Eckpunkt innerhalb eines anderen Körpers

Abbildung 2.14: Eckpunkte zweier Körper bei einer Kollision

Das SAT ist im Grunde eine 1-dimensionale Überprüfung von den minimalen und maximalen Positionen der Ecken zweier Objekte entlang einer Achse. Da je nach Form und Lage der Körper die Rotation einer trennenden Ebene im Raum unterschiedlich sein kann, ist es wichtig alle möglichen Achsen zu überprüfen. Eine kontinuierliche Berechnung aller

Rotationen im Raum ist nicht möglich, weshalb eine Auswahl auf wenige interessante Kandidaten getroffen werden muss.
Die beste Möglichkeit hierfür ist über die Normalenvektoren der verschiedenen Flächen eines Körpers [92]. Durch die konvexen Eigenschaften der Objekte muss parallel zu einer der äußeren Flächen eine trennende Achse ermittelbar sein, wenn keine Kollision auftritt. Abbildung 2.15 zeigt eine zu untersuchende Achse entlang des Normalenvektors der Fläche. Die Überprüfung entlang dieser Achsen muss für alle Flächen der zwei Objekte durchgeführt werden.

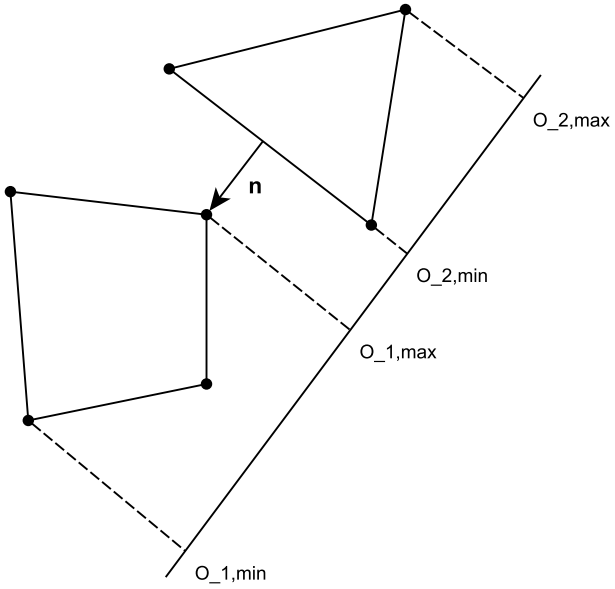


Abbildung 2.15: Rotation der Achse für die 1-dimensionale Überprüfung entlang des Normalenvektors

Für die Überprüfung müssen die minimalen und maximalen Werte der Objekte $O_{1,min}$, $O_{1,max}$, $O_{2,min}$ und $O_{2,max}$ entlang des Normalenvektors berechnet werden, um eine Aussage über die Kollision treffen zu können. Dies kann durch die Berechnung des Skalarproduktes des Normalenvektors $\boldsymbol{n}_L$ mit den Vektoren zu den einzelnen Punkten $\boldsymbol{p}_x$ über

$$O_x \;=\; \boldsymbol{n}_L \cdot \boldsymbol{p}_x \tag{2.29}$$

erfolgen, denn das Skalarprodukt stellt die Projektion eines Vektors auf einen anderen dar [19]. Anschließend müssen die minimalen und maximalen Werte der jeweiligen Objekte miteinander verglichen werden. Wenn alle Werte für die Eckpunkte eines Körpers kleiner

sein sollten als für den Anderen, tritt keine Kollision auf. Erst wenn diese Überprüfung für alle Flächen und Punkte erfolglos ist, berühren sich die Objekte.
Eine Herausforderung bei diesem Verfahren sind runde Körper. Der Sonderfall einer Kugel kann recht leicht durch die zusätzliche Berücksichtigung des Radius gelöst werden. Bei komplexeren abgerundeten Formen wird dies jedoch schwerer. Eine Lösung hierfür ist die Erstellung eines künstlichen Netzes aus Eckpunkten und kleinen Flächen, die den jeweiligen Körper nachstellen. So kann die Berechnung auch hier mit dem SAT durchgefürt werden. Dabei erhöht sich jedoch auch der Rechenaufwand, da sehr viele Punkte abgeglichen werden müssen.

Bei einer Kollision der zwei Objekte kann beim SAT zusätzlich der der sogenannte Minimum Translation Vector (MTV), also der kürzeste Vektor für die Trennung der Körper, berechnet werden [92]. Hierfür muss einfach die kleinste Überlappung $O_{diff,min}$ auf dem jeweiligen Normalenvektor der Ebene L gespeichert werden. Der MTV ist dann

$$\overrightarrow{MTV} \;=\; O_{diff,min} \cdot \frac{\boldsymbol{n}_L}{|\boldsymbol{n}_L|^2}. \tag{2.30}$$

Dieser Wert kann dabei helfen den Zeitpunkt einer Kollision zu finden.

2.5 JupyterLab

In dieser Arbeit wird die Entwicklung der KI mithilfe von JupyterLab durchgeführt. Dies ist eine webbasierte Umgebung für die Entwicklung und Dokumentation von Code in verschiedenen Programmiersprachen, wobei jedoch Python der Standard ist [83].
Ein Nutzer kann in JupyterLab ganze Programme schreiben, interaktive Grafiken erstellen und auch die Dokumentation der Ergebnisse durchführen. Dabei ist es möglich einzelnen Abschnitte anzupassen und einzeln auszuführen. Aus diesem Grund ist JupyterLab sehr beliebt in der Datenverarbeitung und KI-Entwicklung, denn hier können zeit- und rechenintensive Abschnitte von der weiteren Verarbeitung getrennt werden [66]. So kann beispielsweise ein langes Training eines neuronalen Netzes vorgenommen und anschließend getestet und dokumentiert werden. So werden z. B. nicht mehrere einzelne Skripte benötigt.

3 Stand der Technik

In diesem Kapitel wird ein Überblick zum Stand der Technik gegeben. Hierbei wird zum einen auf bisherige Arbeiten zum automatisierten Tischkicker eingegangen, zum anderen werden ähnliche vorangegangene Forschungen auf dem Gebiet analysiert. Aus dem Stand der Technik können sinnvolle Implementierungen abgeleitet und unerforschte potentielle Ansätze erkannt werden.
Zunächst muss diese Arbeit im Gesamtkontext des automatisierten KI Tischkickers eingeordnet werden.

3.1 Automatisierter KI Tischkicker

Das Projekt der Automatisierung eines Tischkickers mithilfe einer KI ist gänzlich neu an der HAW Hamburg. Es gibt weder genaue Anforderungen an die Funktionalität des fertigen Systems, noch feste Vorgaben für die Vorgehensweise der Entwicklung. Es steht einzig die Idee zur Automatisierung eines Tischkickers. Dabei ist das Hauptziel der Idee die Entwicklung eines werbewirksamen Tischkickers, welcher zur Veranschaulichung verschiedener Techniken und Möglichkeiten dienen soll. Dabei soll die Steuerung von einer oder zwei Mannschaften flexibel über Verfahren des maschinellen Lernens erfolgen, um einen möglichen Einsatzbereich und die erreichbare Qualität einer KI im Duell mit einem Menschen oder einer anderen KI zu zeigen. Gleichzeitig kann der automatisierte Tischkicker als Plattform für Projekte und Abschlussarbeiten künftiger Studierender, die sich z. B. mit Themen wie der Bildvararbeitung oder dem Reinforcement Learning befassen wollen, nutzbar gemacht werden.

Es gilt also, das gesamte Projekt, von der Idee bis zum fertigen Produkt zu durchdenken, zu skalieren und in einzelne Arbeitspakete zu unterteilen. Eine genaue Analyse der durch die Stakeholder und verschiedenen Anwendungsfälle gegebenen Anforderungen an den Tischkicker wird im nächsten Kapitel durchgeführt.

3.2 Vorangegangene Forschungen

Die Idee eines automatisierten Tischkickers, welcher teilweise von einer Maschine gesteuert wird, ist nicht gänzlich neu. In verschiedenen Arbeiten und Projekten wurden bereits ähnliche Ansätze vewendet [87, 26, 45]. Dabei wurden sehr unterschiedliche Ansätze zur Steuerung verfolgt. Einige Tischkicker werden über spezielle Algorithmen gesteuert, während Andere mittels einer KI die bestmögliche nächste Aktion berechnen. Im Folgenden werden die wichtigsten Optionen genannt.

3.2.1 Steuerung

Der wichtigste Teil der automatisierten Tischkicker ist die Steuerung. Einerseits gibt es viele verschiedene Möglichkeiten zur Bewegung der Stangen, andererseits muss auch eine Berechnung der nächsten Aktion vom Computer ausgeführt werden.

Aktoren

Die Steuerung der Stangen ist ein zentraler Anwendungsfall des automatisierten Tischkickers. Sie muss kraftvolle, schnelle und präzise Bewegungen durchführen können, damit die computergesteuerte Seite mit einem menschlichen Spieler mithalten kann.
In vorangegangenen Arbeiten, die sich mit der Automatisierung von Tischkickern befasst haben, werden verschiedene Antriebsarten für die Steuerung der Spielfiguren verwendet. Die meisten Kicker werden hierbei durch Elektromotoren, in der Regel Servomotoren, angetrieben [78, 81]. Diese sind präzise und schnell und können verhältnismäßig leicht gesteuert werden.
Als weitere Optionen werden in [65] Hydraulik- und Pneumatik-Systeme vorgeschlagen, die beide über Druck funktionieren. Hierbei wird Hydraulik jedoch bereits ausgeschlossen, da keine passenden Optionen für diesen Einsatzzweck auf dem Markt vorhanden sind.

Die Übertragung der Bewegung der Motoren auf die Stange werden unterschiedliche gelöst. Der einfachste Ansatz ist wie in [87] beschrieben über einen Schlitten samt Motor für die Rotation, der auf einer Linearführung verschoben wird. Hierfür werden nur die Motoren und die besagte Linearführung benötigt. Statt einen Schlitten zu verwenden, kann z. B. wie in [20] mit Zahnrädern gearbeitet werden.

Bei anderen Lösungen werden speziell gefertigte Teile benötigt. Dadurch kann wie in [81, 101] verhindert werden, dass einer der Motoren bewegt werden muss (siehe Abbildung 3.1). Da die Motoren recht schwersind und durch ihr Wegfallen die Trägheit der zu verschiebenden Stangen verringert wird, sind dynamischere Bewegung möglich.

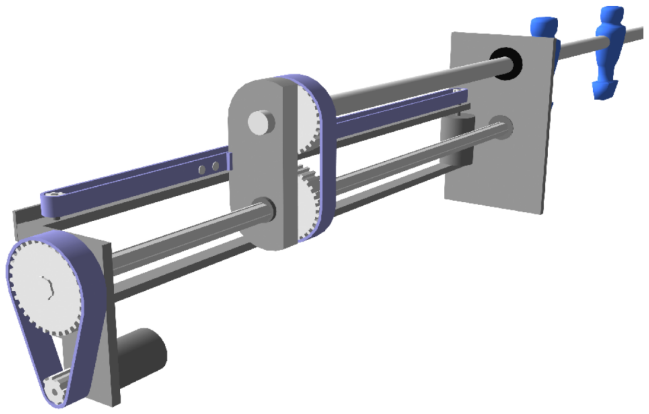


Abbildung 3.1: Antrieb der Stangen ohne beweglichen Motor [101]

Berechnung der nächsten Aktion

Die Berechnung der nächsten Aktion kann über verschiedene Algorithmen gelöst werden. Ihre Spielstärke ist je nach Aufwand der Programmierung gegeben. Ein potentieller Algorithmus kann beispielsweise versuchen nur der Position des Balls mit den eigenen Spielern zu verfolgen. Verbesserungen können hier wie in [98] über das Verteidigen mit versetzten Stangen erfolgen oder durch die Berechnung eines möglichen Kontaktpunkts. Viel interessanter ist jedoch der Ansatz über eine KI. Diese kann wie in [43, 26] durch Reinforcement Learning trainiert werden. Durch das Spielen gegen einen menschlichen Benutzer kann die KI, anders als bei der festen Programmierung eines Algorithmus, auch eigene Taktiken entwickeln und so potentiell besser werden als ein Mensch. Die KI erhält in diesen Arbeiten kein direktes Kamerabild, sondern nur die Metadaten, die durch eine Sensordatenverarbeitung ermittelt werden.

3.2.2 Sensorik

Für die Berechnung der nächsten Aktion müssen in den verschiedenen Arbeiten die Position des Balls und der Spielfiguren eingelesen werden. Auch hier gibt es mehrere Ansätze für die verschiedenen Aufgaben.

Balldetektion

Die Balldetektion liefert mit der Position des Balls die wohl wichtigste Information. Hierfür werden in [65] verschiedene Ansätze vorgeschlagen. Die Lokalisierung könne demnach über einen großen Touchscreen am Boden des Kickers oder über ein Bluetooth-System mit mehreren Komponenten für die Positionsbestimmung mittels Triangulation erfolgen. Diese Methoden haben jedoch den Nachteil, dass keine passenden Produkte auf dem Markt verfügbar sind.

Bei allen tatsächlich realisierten Implementierungen wird daher auf eine Kamera gesetzt [44, 90, 87, 26]. Ihr Bild wird in der Regel auf die bekannten Maße des Tischs projiziert, um die Verzerrung der Kamera zu umgehen.
Da durch die Spielfiguren auf dem Kicker die Sicht auf den Ball beschränkt ist, werden mehrere Optionen verwendet, um eine optimale Detektion zu erhalten. Mithilfe der Position der Spielfiguren kann z. B. wie in [44] eine Maske generiert werden, die verdeckte Bereiche erkennt, sodass ein teilweise verdeckter Ball von den Figuren getrennt werden kann. Diese Art der Verarbeitung ist in Abbildung 3.2 dargestellt.

(a) Originales Bild (b) Maske (c) Gefiltertes Bild

Abbildung 3.2: Balldetektion über das Filtern der Spielfiguren durch eine Maske [44]

Eine weitere Möglichkeit die Detektion zuverlässiger zu machen, ist über die Verwendung mehrerer Kameras wie in [90]. Hierbei müssen jedoch auch mehr Daten verarbeitet werden, wodurch die Detektion seltener durchgeführt werden kann.

Um die Figuren gänzlich zu umgehen, kann auch eine Kamera unterhalb des Spielfelds angebracht werden [87, 98]. Da der Ball in der Regel nur auf dem Boden rollt, sollte so immer freie Sicht auf diesen sein. Natürlich muss hierfür zunächst der Boden des Tischs durch eine Glasscheibe ausgetauscht werden.

Erkennung der Rotation und Verschiebung der Stangen

Die Erkennung der Rotation und Verschiebung der Stangen kann ebenfalls über eine Kamera erfolgen. Hierbei lässt sich die Verschiebung und Rotation wie in [13] durch die Verwendung spezieller Marker realisieren. Es kann aber auch ohne Marker einfach über die Farben der Figuren eine Wahrscheinlichkeit und somit einen Erwartungswert ihrer Position berechnet werden [11].

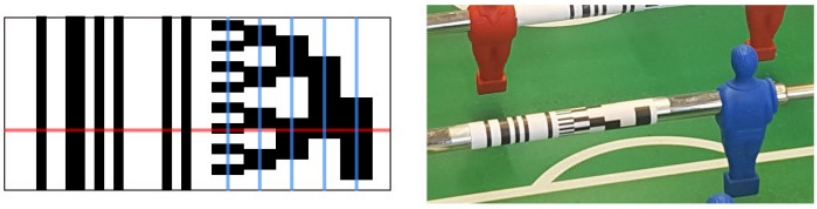


Abbildung 3.3: Erkennung der Verschiebung und Rotation der Stangen über spezielle Marker [13]

Anders als bei der Balldetektion gibt es verschiedene Implementierungen, der Zustandserkennung der Stangen, bei denen keine Kamera verwendet wird. Stattdessen wird mithilfe von Laser-Abstandssensoren die Verschiebung gemessen [78]. Die Rotation kann dabei auf die gleiche Weise ermittelt werden, wenn wie in [87] eine Spirale am Ende der Stange befestigt ist. So kann die Rotation aus der Subtraktion der beiden Messwerte errechnet werden.
Bei diesem Verfahren werden die Positionen der computergesteuerten Mannschaft über die Position der Motoren bestimmt.

3.2.3 Qualität

Die verschiedenen Implementierungen sind unterschiedlich ausgereift. Generell scheint jedoch die Auswertung der Sensordaten über die verschiedenen genannten Methoden sehr gut möglich zu sein. In [44] kann beispielsweise eine Detektionsrate von 100 % erreicht

werden.
Die Qualität der KI bzw. des Algorithmus ist hingegen schwieriger zu ermitteln. In [43] kann jedoch ein Torwart trainiert werden, der den Ball in etwa 80 % der Fälle abwehrt. Im Gegensatz dazu kann die KI aus [26] in der Simulation mit einer Wahrscheinlichkeit von etwa 75 % das Tor treffen. Die Ergebnisse verschlechtern sich jedoch deutlich beim Übergang auf das reale System.
Mit der Verwendung eines intelligenten Algorithmus kann in [87] ein Tischkicker gänzlich automatisiert werden, der einen menschlichen Benutzer im regulären Spiel besiegen kann.

4 Anforderungsanalyse

Das Ziel dieses Kapitels ist es, die Anforderungen für den durch künstliche Intelligenz gesteuerten Tischkicker aus dem realen Anwendungsfall abzuleiten, mögliche Probleme zu identifizieren und einen Rahmen für diese Arbeit zu definieren. In einem ersten Schritt muss abgeschätzt werden, wie das optimale System aussehen kann und welche Funktionen es beinhaltet.
Bisher sind bei klassischen Tischkickern immer mindestens zwei Spieler erforderlich, alleine lässt sich kein vernünftiges Spiel realisieren. Das Ziel ist deshalb, einen Tischkicker zu entwickeln, der auch ein Spiel eines einzelnen Spielers erlaubt. Dabei wird mindestens eine Mannschaft vom Computer und angeschlossenen Motoren gesteuert. Um ein möglichst realistisches Spielerlebnis zu ermöglichen, soll die vom Computer übernommene Mannschaft über eine künstliche Intelligenz gesteuert werden.

4.1 Stakeholder

Die Anforderungen an den Tischkicker ergeben sich aus den Interessen verschiedener Stakeholder. Deshalb ist eine Analyse der Stakeholder samt ihrer Anliegen sinnvoll.

4.1.1 Nutzer

Ein wichtiger Stakeholder bei dieser Arbeit sind die Benutzer des Tischkickers.
Diese möchten ein realistisches und unterhaltsames Spielerlebnis erfahren. Dies bedeutet, dass das gegnerische Team ein ernstzunehmender Konkurrent sein muss und die künstliche Intelligenz die Spieler dementsprechend gut steuern kann. Allerdings sollte stets ein faires Spiel möglich sein, damit der Benutzer auch Spaß am Spiel hat. Das bedeutet also, dass die KI auch nicht zu stark optimiert sein darf, sondern im besten Fall anpassbar an die Fähigkeiten des Benutzers ist.

Für ein angenehmes Spielerlebnis ist auch die Reaktionszeit und Spielweise der vom Computer gesteuerten Mannschaft entscheidend, da eine geringere Reaktionszeit der gegnerischen Mannschaft die Schwierigkeit für den Nutzer erhöht. Zudem kann es für einen Menschen unangenehm sein, wenn die automatisch gesteuerten Spieler durchgehend in Bewegung sind und z. B. komplett rotiert werden (hierbei spricht man von Trillern).
Zuletzt hat ein menschlicher Benutzer natürlich das Interesse nicht von den automatisch gesteuerten Spielen verletzt zu werden. Eine Gefahr bilden dabei die Griffstangen, an denen die Spieler befestigt sind, welche beim Verschieben der Spieler auf einer Seite des Tisches herausgeschoben werden. Eine weitere Gefahr entsteht durch die sich bewegenden Figuren, wenn ein Mensch seine Hand in den Spielbereich hält, um z. B. einen liegengebliebenen Ball zum Rollen zu bringen. Außerdem ist auch die Gefahr durch fliegende Bälle zu beachten, auch wenn diese in der Regel nicht besonders hoch ist.

4.1.2 Auftraggeber und Prüfer

Als weiterer Stakeholder ist Prof. Dr. Hensel als Auftraggeber und Prüfer aufzuführen.

Als Auftraggeber besteht das Interesse einen funktionstüchtigen Tischkicker zu erhalten, welcher mithilfe einer KI gesteuert wird. Dabei soll der Tischkicker ein werbewirksames Projekt zum Präsentieren der Möglichkeiten von künstlicher Intelligenz im Zusammenspiel mit dem Menschen in einer realen Umgebung sein. Im Rahmen dessen steht besonders die einwandfreie Funktionalität der automatischen Steuerung einer Mannschaft durch die KI im Fokus. Gleichzeitig soll der Tischkicker als flexible Plattform für Studierende dienen und für Projekte und Abschlussarbeiten genutzt werden.
Weiter sollte der Tischkicker möglichst platzsparend gelagert und transportiert werden können, um gegebenenfalls bei Ausstellungen präsentiert werden zu können. Hierfür ist eine einfache Installation des Systems notwendig, um einen schnellen Aufbau zu ermöglichen. Dies bedeutet, dass nach Möglichkeit nur ein Netzstecker verwendet werden muss, um den Tischkicker einsatzbereit zu machen. Hierfür ist dann auch die Verwendung einer mobilen Plattform nötig, welche die Sensordatenauswertung und Steuerung übernimmt. Gleichzeitig muss eine Schnittstelle für den Zugriff auf die Plattform vorhanden sein, sodass beispielsweise verschiedene künstliche Intelligenzen implementiert und miteinander verglichen werden können.
Ein weiteres Interesse besteht auch an der Robustheit des Gesamtsystems, sodass keine softwareseitigen Probleme beim fertigen Tischkicker auftreten und gleichzeitig die Hard-

ware auch in Form des Kickers selbst möglichst lange unbeschädigt bleibt. Da bei einer solchen Anlage mit einem gewissen Maß an Verschleiß zu rechnen ist, sollte der gesamte Tischkicker leicht wartbar sein. Hierfür ist ein modularer Aufbau sinnvoll, sodass einzelne Elemente ausgetauscht oder repariert werden können. Gleichzeitig ist eine flexible Anbindung weiterer Komponenten wünschenswert, damit das System auf Wunsch erweitert werden kann.
Zuletzt ist auch das Einhalten des Budgets von Interesse sowie die zeitige Fertigstellung des automatisch gesteuerten Tischkickers.

Als Prüfer besteht vor allem ein Interesse an der Komplexität der verwendeten Lösung und der eingesetzten Methodik. Die Arbeit soll in sich schlüssig und abgeschlossen sein.

4.1.3 Ersteller der Arbeit

Für den Ersteller der Arbeit besteht vor allem ein hohes Interesse am erfolgreichen Abschluss dieser. Des Weiteren stehen auch das Erlernen neuer Fähigkeiten und das Aneignen von weiterem Wissen im Vordergrund. Es besteht ein hohes Interesse an der Fertigstellung des Tischkickers, wobei gleichzeitig ein zeitlicher Rahmen für die Arbeit nicht überschritten werden soll, sodass ein geeignetes Arbeitspaket definiert werden muss. Dabei ist zumindest eine teilweise Implementierung der KI auf dem realen Tischkicker wünschenswert.

4.2 Anwendungsfälle

In diesem Abschnitt werden die spezifischen Anwendungsfälle genauer beleuchtet. Eine Übersicht dieser ist in dem UML-Anwendungsfall-Diagramm in Abbildung 4.1 zu sehen. Grob unterteilen sich die Anwendungsfälle in zwei Kategorien, die Verarbeitung der Sensordaten und die Ansteuerung des Systems. Die Sensordatenverarbeitung unterteilt sich in verschiedene Unterpunkte, welche dazu dienen, den aktuellen Zustand des Tischkickers optimal aufzunehmen. Die KI soll dann mit diesen vorverarbeiteten Daten als eine Art Schnittstelle zwischen der Verarbeitung der Sensordaten und der Ansteuerung der Motoren dienen.

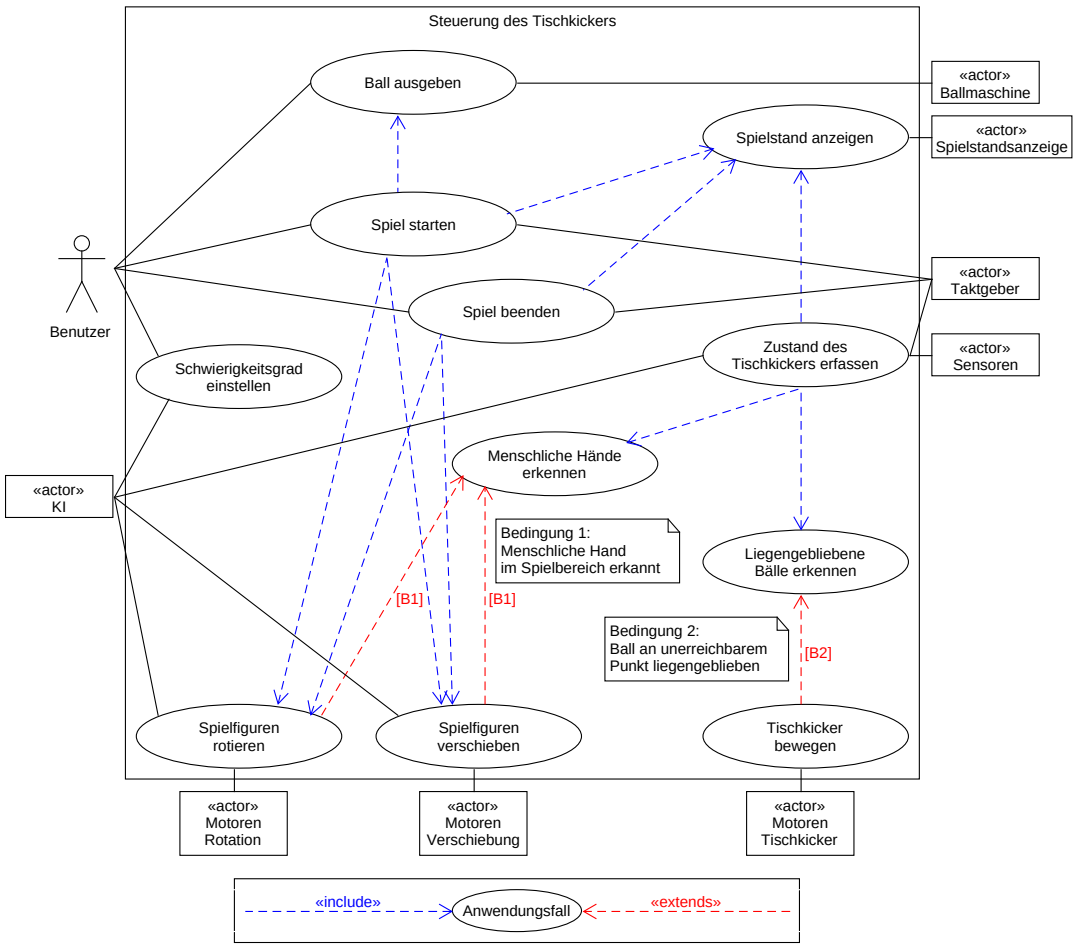


Abbildung 4.1: Anwendungsfall-Diagramm des automatisierten Tischkickers

4.2.1 Zustand des Tischkickers erfassen

Zunächst gilt es den Zustand der Umgebung, also hier des Tischkickers, zu erfassen, damit die Daten zur Steuerung an den Computer weitergegeben werden können. Erst eine möglichst genaue Erfassung des Ist-Zustands ermöglicht der KI die Berechnung der optimalen Reaktion. Deshalb ist die präzise Verarbeitung der Sensordaten essentiell. Die Erfassung teilt sich in zwei Bereiche auf. Einerseits muss der Spielball und andererseits die Spielfiguren erkannt werden.

Spielball erfassen

Am wichtigsten ist die Erfassung des Spielballs, denn dieser entscheidet letztendlich das gesamte Spiel. Zum einen ist die genaue Position dessen auf dem Spielfeld von hoher Bedeutung, zum anderen muss jedoch auch die Geschwindigkeit und Bewegungsrichtung berücksichtigt werden, um vorausschauend agieren zu können. Die Position wird nicht nur zum Verteidigen und Spielen des Balls benötigt, sondern auch entscheidend für den aktuellen Spielstand, indem erzielte Tore erkannt werden.

Spielfiguren erfassen

Neben dem Ball ist auch die Erfassung der Position und Rotation der Spielfiguren wichtig. Zum einen muss die KI natürlich wissen, wo sich die eigenen Spieler befinden und wie weit diese eingedreht sind, um Bälle abwehren und selbst schießen zu können. Zum anderen ist aber auch die Lokalisierung der gegnerischen Mannschaft sinnvoll, damit sich der Einsatz einer KI überhaupt lohnt, da so Lücken in der gegnerischen Verteidigung ermittelt und gezielt angespielt werden können, anstatt den Ball nur „blind“ nach vorne zu schlagen. Die Rotation der gegnerischen Spielfiguren ist nicht unbedingt von hohem Interesse, da diese in der Regel orthogonal zum Boden stehen und nur als Hindernis wahrgenommen werden müssen. Allerdings kann hier eine schräge Stellung der Spieler auch den Treffpunkt vom Ball und den Spielfiguren leicht verändern, sodass die zusätzliche Erfassung eine weitere Verbesserung darstellen kann.

Die Erfassung des gesamten Zustands des Tischkickers muss regelmäßig und so oft wie möglich erfolgen. In Folge dessen müssen die Sensordaten eingelesen und verarbeitet werden und anschließend der KI zur Weiterverarbeitung zur Verfügung gestellt werden. Der Taktgeber, welcher diesen Anwendungsfall auslöst, muss an die Geschwindigkeit der Datenverarbeitung, der KI und nicht zuletzt an die Abtastrate der Sensoren angepasst werden. Je häufiger die Daten erfasst werden, desto genauer kann der tatsächliche Zustand des Kickers abgebildet werden.

Menschliche Hände im Spielbereich erkennen

Die von den Motoren betriebene Mannschaft stellt ein gewisses Sicherheitsrisiko da, wenn der Benutzer z. B. einen liegen gebliebenen Ball anstoßen möchte. Um das Versetzungsrisiko des Benutzers zu verringern, muss verhindert werden, dass die Spieler vom Computer

angesteuert werden, sofern sich eine menschliche Hand über dem Spielfeld befindet. Das bedeutet, dass eine Erkennung von Fremdkörpern bzw. Händen im Spielbereich erfolgen muss.

Auch dieser Anwendungsfall muss wie die Erfassung des Zustands des Tischkickers möglichst häufig und regelmäßig ausgelöst werden, damit keine Verletzungen entstehen. Deshalb wird er durch den Anwendungsfall der Erfassung des Zustands des Tischkickers ausgelöst. Anschließend werden die eingelesenen Sensordaten verarbeitet. Wenn eine Hand erkannt wird, muss ein Stopp-Befehl an die Steuerung weitergeleitet werden. Sobald keine Hand mehr erkannt wird, kann eine erneute Freigabe der Motorsteuerung erfolgen.

Liegengebliebene Bälle erkennen

Ein Problem bei vielen Tischkickern ist, dass der Ball an bestimmten Positionen von keiner Spielfigur mehr erreicht werden kann. Wenn im Verlauf der Partie der Ball an einer dieser Stellen zum Stehen kommt, muss normalerweise ein Spieler den Ball mit der Hand wieder ins Rollen bringen. Da dies hier durch die vom Computer gesteuerten Figuren ein gewisses Verletzungsrisiko birgt und zudem diese Methode häufig nicht gerade fair ist, soll dieser Prozess automatisiert werden.

Dieser Anwendungsfall wird durch die Erfassung des Zustands des Tischkickers ausgelöst und ermittelt, ob der Ball außerhalb der mit den Spielfiguren erreichbaren Regionen liegengeblieben ist. Wenn dies der Fall ist, können z. B. Maßnahmen ergriffen werden um den Ball wieder in Bewegung zu setzen.

4.2.2 System steuern

Nach der Erfassung der Umgebung kann mit der Verarbeitung der Daten begonnen werden. Hierbei werden dann die bestmöglichen Aktionen für den aktuellen Zustand berechnet und anschließend angesteuert.

Nächste Aktion berechnen

Zunächst muss ermittelt werden, welche Aktion von der automatisierten Mannschaft als nächstes durchgeführt werden soll. Hierbei müssen die ermittelten Positionen und Zustände der Spielfiguren und des Balls berücksichtigt werden, um ein optimales Ergebnis

zu erhalten. Durch klassische Algorithmen und Entscheidungsbäume kann relativ leicht ein durchaus passabler Gegner simuliert werden [4], allerdings ist durch die Anforderungen der Stakeholder gegeben, dass die Entscheidungen des Computers über eine KI gesteuert werden sollen. Speziell soll dabei das Verfahren des Reinforcement Learning angewendet werden. Hierdurch kann ein sehr flexibler Spieler simuliert werden, der auch eigene Taktiken entwickeln kann.
Dieser Anwendungsfall wird entsprechend der Anforderungen der Stakeholder durch den Einsatz einer KI gelöst. Diese wird nach der Erfassung des Zustands aufgerufen und erhält die ermittelten Daten. Die KI selber berechnet den bestmöglichen Spielzug und ruft anschließend die Anwendungsfälle zur Ansteuerung der Spielfiguren durch die verschiedenen Motoren auf. Diese unterteilen sich in die Steuerung der Verschiebung und der Rotation der Spielfiguren.

Spielfiguren verschieben

Die Verschiebung der Spielfiguren bildet einen der zwei Anwendungsfälle zur Steuerung und ist dafür verantwortlich die Berechnung der KI in eine seitliche Verschiebung der einzelnen Stangen zu übertragen. Eine schnelle und präzise Verschiebung ist der Schlüssel für ein erfolgreiches Spiel, da vor allem die Verteidigung von Schüssen auf das eigene Tor durch diese Bewegung realisiert wird. Gleichzeitig muss auch ein Stopp-Befehl während der Ausführung der Verschiebung verarbeitet werden können, wenn entweder eine Hand im Spielbereich erkannt wird, oder das Spiel beendet werden soll. Wichtig ist auch die Anpassung an die Eigenschaften des Motors, sodass eine konstant zuverlässige Ansteuerung möglich wird.

Ausgelöst durch die KI muss in diesem Anwendungsfall der Ausgang jener verarbeitet und die Ansteuerung der verschiedenen Motoren zur Verschiebung der Stangen reguliert werden. Bei dem Aufruf durch einen Stopp-Befehl muss die Steuerung bis zur erneuten Freigabe eingefroren werden.

Spielfiguren rotieren

Ähnlich wie bei der Verschiebung funktioniert auch die Rotation der Spielfiguren. Sie wird ebenfalls durch die KI aufgerufen, wobei die Rotation eher ein Mittel zum Schießen von Toren ist. Auch hier müssen Stopp-Signale verarbeitet werden, sodass keine Verletzung entstehen, bzw. das Spiel beendet oder pausiert werden kann.

Analog zur Verschiebung wird im Regelfall auch der Anwendungsfall der Rotation durch die KI ausgelöst und behandelt die Verarbeitung der Ausgangsdaten der KI und die Ansteuerung der verschiedenen Motoren zur Rotation der Stangen. Bei dem Aufruf durch einen Stopp-Befehl muss auch hier die Steuerung bis zur erneuten Freigabe eingefroren werden.

Tischkicker bewegen

Um einen liegengebliebenen Ball, welcher nicht erreichbar für die Spielfiguren ist, wieder in Bewegung zu bringen, kann ein Motor am Kicker befestigt werden. Wenn dieser Angesteuert wird, soll der Tisch z. B. leicht gerüttelt oder gekippt werden, sodass der Ball sich langsam wieder in Bewegung versetzt. Wichtig ist hierbei wieder der Sicherheitsaspekt, d.h. eine Gefahr für den Nutzer durch bewegliche Teile muss verhindert werden.

Dieser Anwendungsfall wird nach dem Erkennen eines liegengebliebenen Balls aufgerufen und sorgt dann für die Ansteuerung des Motors/der Motoren für das Ballrollen.

4.2.3 Spiel starten

Zu Beginn eines Spiels muss der Benutzer das Spiel starten. In diesem Anwendungsfall wird dann der Spielstand zurückgesetzt und diese Information an die Displaysteuerung weitergegeben. Gleichzeitig soll automatisch ein Ball über die Ballmaschine ausgegeben werden, sodass der Anwendungsfall zur Ausgabe eines Balls aufgerufen wird. Weiter müssen auch die Motorsteuerungen und der Taktgeber aktiviert werden.

4.2.4 Spiel bewegen

Wenn der Benutzer das Spiel beenden möchte, kann er diesen Anwendungsfall aktivieren. Der Aufruf wird dann sofort bearbeitet und das System abgestellt. Das bedeutet, dass die Anwendungsfälle für die Rotation und die Verschiebung der Spielfiguren aufgerufen werden, die dann die Motoren blockieren. Zudem muss der Taktgeber für die Erfassung des Zustands abgeschaltet werden und die Spielstandsanzeige soll den Endstand anzeigen.

4.2.5 Schwierigkeitsgrad einstellen

Da die KI im Optimalfall wesentlich besser als ein menschlicher Benutzer spielen kann, dies jedoch für den Spielspaß des Benutzers nicht vom Vorteil ist, macht es Sinn einen Regler zum Einstellen des Schwierigkeitsgrades hinzuzufügen.

Der Benutzer kann durch eine Eingabe diesen Anwendungsfall aufrufen und den Schwierigkeitsgrad entsprechend seines Könnens anpassen. Diese Eingabe muss zunächst verarbeitet werden und anschließend an das System weitergegeben werden.

4.2.6 Spielstand anzeigen

Über die Erfassung des Zustands können Tore ermittelt werden, welche dann auf einer Spielstandsanzeige angezeigt werden. Außerdem kann eine fortschrittliche KI möglicherweise auch anhand des Spielstands ihre Taktik anpassen.

Beim Start des Spiels und nach jedem Tor wird dieser Anwendungsfall aktiviert. In diesen Fällen muss dann ein Displaycontroller informiert werden, welcher anschließend die Ausgabe auf dem Display regelt.

4.2.7 Ball ausgeben

Ein weiteres Feature des Tischkickers wäre eine automatische Ballmaschine, die nach einem geschossenen Tor, den Ball automatisch dem Spiel zurückführt. Zwar könnte eine Ausgabe immer sofort nach einem Tor erfolgen, sinnvoller ist hier jedoch, dass die Ausgabe erst nach einer Aufforderung des Benutzers ausgeführt wird, damit dieser ausreichend Zeit hat, um sich auf einen neuen Spielzug vorzubereiten.

Die Ausgabe eines neuen Balls erfolgt also immer nach Anforderung des Benutzers oder beim Start eines Spiels. Diese Anfrage muss dementsprechend erkannt und verarbeitet werden. Anschließend muss ein Signal an die Ballmaschine weitergeleitet werden, welche die Ausgabe auf der Hardware übernimmt.

4.3 Formulierung der Anforderungen

Aus den Anwendungsfällen und der Anforderungen der Stakeholder können konkrete Anforderungen und Ziele an das System formuliert werden. Diese unterteilen sich dabei in funktionale und nicht-funktionale Anforderungen. Da der Umfang der gesamten Arbeiten am Tischkicker den zeitlichen Rahmen dieser Arbeit überschreitet, werden die einzelnen Anforderungen priorisiert. Eine hohe Priorisierung bedeutet, dass die entsprechende Anforderung für das fertige Gesamtsystem essentiell sind und zumindest teilweise in dieser Arbeit gelöst werden müssen. Mit einer niedrigen Priorität werden hingegen verschieden Zusatzfunktionen markiert, welche keinen Einfluss auf die Basisfunktionen des automatisierten Tischkickers haben. Diese Anforderungen müssen für einen erfolgreichen Abschluss dieser Arbeit nicht zwingend erfüllt sein.

4.3.1 Funktionale Anforderungen

Die funktionalen Anforderungen an das System sind in Tabelle 4.1 aufgeführt.

Tabelle 4.1: Funktionale Anforderungen des Tischkickers

ID	Priorität	Anforderung
F1	hoch	Erkennung der Position und Bewegung des Balls
F2	hoch	Erkennung der Verschiebung und Rotation der Spielfiguren
F3	hoch	Berechnung der nächsten Aktion durch eine KI
F4	hoch	Die KI kann gegen einen menschlichen Gegner Tore verhindern
F5	hoch	Die KI kann gegen einen menschlichen Gegner Tore schießen
F6	niedrig	Einstellbarkeit des Schwierigkeitsgerades der KI
F7	hoch	Ansteuerung der Rotation der Spielfiguren
F8	hoch	Ansteuerung der Verschiebung der Spielfiguren
F9	hoch	Anhalten aller elektrisch beweglichen Teile
F10	niedrig	Anzeige des Spielstands
F11	niedrig	Automatische Ausgabe eines neuen Balls nach Knopfdruck
F12	niedrig	Erkennung eines liegen gebliebenen Balls
F13	niedrig	Automatisches ins Rollen bringen eines Balls
F14	hoch	Zugriff auf das System durch ein externes Gerät

Für die Realisierung des automatisierten Tischkickers muss der aktuelle Zustand dieses ermittelt werden. Hierzu gehören die Position und Bewegung des Spielballs (F1), sowie die Verschiebung und Rotation der Spielfiguren (F2). Erst mit diesen Informationen können die bestmöglichen Aktionen zum jeweiligen Zeitpunkt ermittelt werden. So ergibt

sich hierfür auch eine entsprechend hohe Priorität.

Der Fokus dieser Arbeit ist auf die künstliche Intelligenz gerichtet, welche für die Berechnung der nächsten Aktion verantwortlich ist. Da diese Arbeit ein komplexes Zusammenspiel aus vielen verschiedenen Aufgabenpaketen ist, muss ein geeigneter Rahmen gefunden werden. Deshalb hat die Anforderung F3 die höchste Priorität, sodass im Zweifel Abstriche in anderen Bereichen in Kauf genommen werden müssen, um eine ausreichende Tiefe in diesem Gebiet zu erreichen. Das Ziel ist gegen einen Menschen spielen zu können, sodass die KI Angriffe des gegnerischen Spielers abwehren und auch selber Tore schießen kann. Für ein verbessertes Spielerlebnis soll zudem der Schwierigkeitsgrad der KI einstellbar sein. Diese Anforderung ist allerdings nicht essentiell.

Für die Übertragung der von der KI berechneten Aktionen auf ein reales System müssen die Spielfiguren entsprechend der Vorgaben angesteuert werden. So ergeben sich die Anforderungen F7 für die Rotation und F8 für die Verschiebung der Spielfiguren. Gleichzeitig muss jedoch auch die Sicherheit des Benutzers gewährleistet sein, sodass alle elektrisch beweglichen Teile bei Bedarf gestoppt werden.

Weitere funktionale Anforderungen wie eine Spielstandsanzeige, einer automatischen Ballmaschine und die Möglichkeit einen unerreichbaren, zum Stehen gekommenen Ball zu erkennen und wieder ins Rollen zu bringen, sind für das Erste nicht von einem hohen Stellenwert.

Durch die Stakeholder wird ein möglicher Zugriff von einem externen Rechner auf das System gefordert, um beispielsweise die Implementation von verschiedenen KI-Modellen zu testen. Die Anforderung F14 wird benötigt um den Tischkicker als Lernplattform für Studenten zu Reinforcement Learing einzusetzen.

4.3.2 Nicht-funktionale Anforderungen

Neben den funktionalen Anforderungen müssen auch die nicht-funktionalen Anforderungen analysiert und festgehalten werden. Sie stellen dabei spezifische Ansprüche an die einzelnen Verarbeitungsschritte und werden in der Tabelle 4.2 aufgelistet und priorisiert.

Tabelle 4.2: Nicht-funktionale Anforderungen des Tischkickers

ID	Priorität	Anforderung
N1	hoch	Bestimmung der Position des Balls mit ±2 mm Genauigkeit
N2	hoch	Bestimmung der Verschiebung der Stangen mit ±1 mm Genauigkeit
N3	hoch	Bestimmung der Rotation der Stangen mit ±1° Genauigkeit
N4	hoch	Verschiebung der Stangen mit einer maximalen Abweichung von ±1 mm
N5	hoch	Rotation der Stangen mit einer maximalen Abweichung von ±1°
N6	hoch	Verschiebung der Stangen mit einer Geschwindigkeit von bis zu $v_{Stange,max} = 2\ \frac{\mathrm{m}}{\mathrm{s}}$
N7	hoch	Schießen des Balls mit einer Geschwindigkeit von bis zu $v_{Ball,test} = 3\ \frac{\mathrm{m}}{\mathrm{s}}$
N8	hoch	Minimale Abtastrate von $f_{s,min} = 50\ \frac{1}{\mathrm{s}}$ zur Erfassung des Zustands
N9	hoch	Beschleunigung der Stangen auf $v_{Reak} = 1,5\ \frac{\mathrm{m}}{\mathrm{s}}$ innerhalb von $T_{Verschiebung} = 200$ ms
N10	hoch	Stoppen der Motoren innerhalb $T_{Krit} = 200$ ms bei der Erkennung einer menschlichen Hand
N11	hoch	Maximale Rechenzeit der KI kleiner als $T_{KI} = 20$ ms
N12	hoch	Die KI kann 80 % der Torschüsse halten
N13	hoch	Die KI kann 80 % der Schüsse auf ein leeres Tor treffen
N14	niedrig	Die KI kann 50 % der Spiele gegen einen Menschen gewinnen
N15	niedrig	Aufrecht positionierbarer Tisch
N16	niedrig	Leichte Hardware mit einem Gewicht unter $m_{max} = 60$ kg
N17	niedrig	Netzanschluss über einen Stecker
N18	niedrig	Tischkicker als Standalone-System
N19	niedrig	Einhaltung des Budgets von 500 €

Für die Realisierung auf dem Tischkicker müssen gewisse Anforderungen an die Qualität der einzelnen Schritte eingehalten werden, um eine Übertragung der von der KI berechneten Aktionen zu erlauben. Zum einen ist die Genauigkeit der Sensordaten entscheidend, da beim Spielen häufig feine Bewegungen benötigt werden, um z. B. den Ball anhalten zu können. Zum Anderen müssen auch die Bewegungen durch die Motoren in einer ähnlich feinen Ausführung möglich sein.
Die Position des Balls sollte mit einer Richtigkeit und Präzision von ±2 mm bestimmt werden können. Eine so genaue Lokalisierung ist notwendig, damit eine KI auch komplexere Taktiken ausführen kann, bei denen beispielsweise der Ball mit der Ecke einer Spielfigur und somit schräg nach vorne geschossen wird. Da der Fehler der Verschiebung noch zu der Ungenauigkeit der Position des Balls addiert wird, muss auch sie einen klei-

nen Fehler haben. Durch die leichtere Messbarkeit soll sie mit einer Abweichung von ±1 mm bestimmbar sein. Ähnlich ist es auch bei der Rotation, welche mit ±1° Genauigkeit erfassbar sein muss, denn so lässt sich bei einer Figur mit 10 cm Abstand zur Stange eine Genauigkeit von angenäherten $\Delta s_{Stange,Fuss} \approx 0.1 \cdot \sin(\frac{2 \cdot 2\pi}{360}) \approx 1.75$ mm erreichen.
Analog muss die Ansteuerung der Figuren ähnlich genau sein, da ein möglicher Fehler sich mit den anderen Abweichungen addiert. So wird hier ebenfalls eine Ansteuerung mit einer maximalen Abweichung von ±1 mm in der Verschiebung und ±1° in der Rotation gefordert. So ergeben sich die nicht-funktionalen Anforderungen N1 bis N5.

Die Motoren müssen nicht nur präzise, sondern auch noch sehr schnell sein, damit auch schnelle Bälle gespielt bzw. abgewehrt werden können. So kann ein Mensch im Selbsttest die Verschiebung mit rund $v_{Stange,max} = 2\ \frac{\mathrm{m}}{\mathrm{s}}$ bewältigen und den Ball durch die Rotation auf $v_{Ball,test} = 3\ \frac{\mathrm{m}}{\mathrm{s}}$ beschleunigen. Dies entspricht auch etwa den Werten die ein durchschnittlicher Spieler laut [107] erreicht. Die Anforderungen N6 und N7 erwarten das Erreichen dieser Grenzwerte auch von der Computer gesteuerten Mannschaft.

Neben der Genauigkeit und der Bewegungsgeschwindigkeit ist auch die Reaktionszeit bzw. Rechengeschwindigkeit ein wichtiger Faktor. Ein durchschnittlicher Mensch kann innerhalb von $T_{Mensch} \approx 220$ ms auf ein Ereignis reagieren [40], sodass hier das Ziel sein muss, dass die KI ähnlich schnell oder sogar schneller reagieren kann. Die Reaktion unterteilt sich in die Abtastzeit T_{Sensor} und den benötigten Zeitraum bis die Verschiebung einer Reihe auf eine gewisse Geschwindigkeit beschleunigt wird. Die Abtastrate ergibt sich aus der maximalen Geschwindigkeit des Balls, wenn sich dieser maximal $s_{Ball,max} = 10$ cm zwischen zwei Abtastungen bewegen soll. Diese Strecke ist etwas geringer, als der Abstand zwischen zwei Reihen bei einem durchschnittlichen Tischkicker. Es zeigt sich, dass $f_{s,min} = \frac{v_{Ball,test}}{s_{Ball,max}} = 50\ \frac{1}{\mathrm{s}}$ sein muss. Um die Reaktionszeit eines Menschen zu imitieren, muss die Beschleunigung der Stangen auf $v_{Reak} = 0.5 \cdot v_{Stange,max} = 1,5\ \frac{\mathrm{m}}{\mathrm{s}}$ innerhalb $T_{Verschiebung} = T_{Mensch} - \frac{1}{f_{s,min}} = 200$ ms erfolgen.
Um die Sicherheit der Benutzer zu gewährleisten, müssen die Motoren innerhalb kurzer Zeit abschaltbar sein. Dieser Zeitraum wird mit $T_{Krit} = T_{Verschiebung} = 200$ ms genau wie die Reaktionszeit bemessen. Gleichzeitig ist auch ein Schutz vor den herausfahrenden Stangen sinnvoll, damit diese kein Verletzungsrisiko für einen Spieler darstellen.
Zwar werden bei der KI keine 50 verschiedenen Ausgaben pro Sekunde an die Steuerung benötigt, da die Steuerung zwangsläufig etwas langsamer ist als die Berechnung, dennoch muss die KI ebenfalls mit einer hohen Abtastung arbeiten, um so schnell wie möglich auf die neuen Informationen zu reagieren. So muss die Berechnung in unter

$T_{KI} = T_{Sensor} = \frac{1}{f_{s,min}} = 20$ ms möglich sein, wenn sie parallel zur Abtastung erfolgt.

Damit ein Spiel zwischen KI und Mensch möglich ist, muss die KI gewissen qualitativen Anforderungen entsprechen. Optimalerweise gewinnt die KI jedes zweite Tischkickerspiel gegen einen Menschen, da dies einem ausgeglichenem Duell entspricht. Die Anforderungs N14 ist jedoch niedrig gesetzt, da in dem gegebenen zeitlichen Rahmen eine perfekte Entwicklung des gesamten Tischkickers nur schwer möglich ist. Eine höhere Priorität bekommen stattdessen die abgeschwächten Anforderungen N12 und N13, die eine 80 prozentige Erfolgsrate der KI bei der Ausführung einfacher Grundlagen des Spiels einfordern. Mit 80 % ist ein realistischer und in vergleichbaren Projekten erreichter Wert gewählt [43].

Aus den Anforderungen der Stakeholder lässt sich weiter auch ableiten, dass der Tischkicker ein möglichst kompaktes und transportables Format haben sollte, damit er bei Bedarf an gewünschter Stelle aufgebaut und präsentiert werden kann. Damit der Tisch leicht lager- und transportierbar ist, sollte dieser nach Anforderungen N15 und N16 hochklappbar sein und ein Gewicht von über $m_{max} = 60$ kg nicht überschreiten, da dies die empfohlene maximale Last einer Frau und eines Mannes kombiniert ist [7]. Gleichzeitig sollte der Aufbau des Systems möglichst einfach sein, sodass im Idealfall nur ein Netzstecker benötigt wird, um den Tischkicker anzuschließen. Daraus leitet sich auch ab, dass der Tischkicker als Standalone-System entworfen werden muss und so eine rechenstarke Plattform zur Sensordatenverarbeitung und Berechnung der nächsten Aktionen benötigt. Zunächst ist allerdings auch ein erster Aufbau möglich, welcher diese Anforderung nicht erfüllt, um den Einsatz der KI gesteuerten Mannschaft auf dem realen System zu testen. Dementsprechend sind die genannten Anforderungen von niedriger Priorität.

Da der Tischkicker ein physisches System ist, mit welchem interagiert werden soll, muss auch eine gewisse Zuverlässigkeit und Robustheit gewährleistet sein. So muss das System für die Zeit eines ganzen Spiels fehlerfrei funktionieren. Außerdem muss die Hardware auch langfristig halten und seltene Reparaturen möglichst leicht möglich machen. Einzelne Teile des Kickers sollten verhältnismäßig leicht austauschbar sein.

Die letzte Anforderung N19 ist die gesetzte Budgetgrenze, die für diese Arbeit zur Verfügung steht. Die Einhaltung dieser Grenze ist zwar erwünscht, kann bei Bedarf jedoch auch überschritten werden, da der Kicker später als Lernplattform eingesetzt werden soll und so die Ausgaben als Investition angesehen werden können.

5 Konzept

In diesem Kapitel wird das Konzept für den KI-gesteuerten Tischkicker erarbeitet. Hierbei werden die verschiedenen Anforderungen berücksichtigt und mögliche Probleme analysiert. Daraus lässt sich ein Rahmen für die weitere Entwicklung der einzelnen Arbeitspakete ableiten. Weiter wird auch eine Auswahl verschiedener Hardwarekomponenten anhand des aufgestellten Konzepts getroffen.

5.1 Training der KI

Der Fokus dieser Arbeit liegt auf der durch Reinforcement Learning trainierten KI, welche die zentrale Schnittstelle zwischen der Sensordatenerfassung und der Ansteuerung der Spielfiguren ist. Die KI entscheidet, welche Aktionen bei einem gegebenen Zustand des Tischkickers auszuführen sind, sodass für ein möglichst realistisches Spielerlebnis die KI gute Entscheidungen treffen können muss.

5.1.1 Trainingsumgebung

Damit eine KI auch wirklich intelligent wird und sinnvolle Entscheidungen treffen kann, muss diese zunächst trainiert werden. Das Training benötigt je nach Komplexität des Modells und der auszuführenden Aufgabe viele Iterationen und ist somit zum Teil sehr rechen- und zeitintensiv. Problematisch wird dies auch, wenn die KI mit der realen Umgebung interagieren soll, da die zu beobachtenden Vorgänge nur in Echtzeit durchlaufen werden können und so nur eine begrenzte Anzahl von Trainingseinheiten möglich ist. Zudem wird dieses Problem in dem vorliegenden Fall dadurch verschärft, dass nur eine Mannschaft des Tischkickers vom Computer gesteuert werden soll, während der menschliche Benutzer die gegenüberliegende Seite verwendet. Somit wäre das Training nicht nur zeit-, sondern auch recht ressorcenintersiv.

Dieses Problem könnte zwar für das Training durch den Umbau des Tischkickers gelöst werden, sodass zwei künstliche Intelligenzen gegeneinander spielen, allerdings bleibt das Problem der maximalen Spielgeschwindigkeit vorhanden. Zudem muss der Tischkicker hierfür bereits optimal vollautomatisiert sein, damit automatisch Bälle eingeworfen werden und diese auch ins Rollen gebracht werden können, sofern sie liegen bleiben. Dennoch würde hier eine Gefahr bestehen, dass das Training unerwartet unterbrochen wird, da z. B. ein Ball durch einen Querschläger aus dem Spielbereich befördert werden könnte.

Eine alternative Möglichkeit zum Training auf dem realen System ist deshalb ein Training der KI mithilfe einer genauen Simulation. Erst wenn in der Simulation hinreichend gute Ergebnisse erreicht werden, sollte diese auf dem realen System implementiert und weiter trainiert werden. So können viele Trainingsdurchläufe ohne Aufsicht eines Menschens und ohne Pause hintereinander ausgeführt werden. Durch ein Training über die Simulation ist es auch möglich mehrere Ansätze gleichzeitig zu verfolgen und die Trainingsmethode gegebenenfalls anzupassen. Für den Erfolg dieser Methode muss die Simulation möglichst genau und auf die Eigenschaften des Tischkickers und der Steuerung anpassbar sein. Anderenfalls verliert das Modell seinen Lernfortschritt, wenn die erlernten Methoden nicht auf dem realen Modell umsetzbar sind.

5.1.2 Trainingsdaten

Für ein erfolgreiches Spiel zwischen Mensch und Maschine muss der Zustand des Tischkickers erfasst und der KI bereitgestellt werden. Eine gängige Methode hierfür ist beim Machine Learning auch speziell beim Reinforcement Learning die Verwendung eines Kamerabilds [63]. Dies scheint auch für den Einsatz beim Tischkicker eine gute Lösung zu sein, da ein Mensch ebenfalls das Spiel visuell analysiert.
Um jedoch ein möglichst gutes Trainingsergebnis zu erhalten, wird der Traingsprozess zum großen Teil in einer Simulation durchgeführt. Bei der Verwendung der Kameradaten müsste also die Simulation so angepasst sein, dass die visuelle Ausgabe dem tatsächlichen Kamerabild sehr ähnlich ist. Dies ist zwar auf verschiedene Weisen möglich, jedoch ist die Implementierung der genauen grafischen Ausgabe sehr aufwändig, sodass hierfür wesentlich mehr Zeit eingeplant werden müsste. Zusätzlich wird die Simulationsgeschwindigkeit durch die visuelle Ausgabe weiter limitiert, sodass auch das Training um einen hohen Faktor langsamer wird.
Zuletzt ist auch zu bedenken, dass trotz genauer Anpassung der Simulation, Unterschiede

zwischen dieser und der Realität zu erwarten ist. Wie gut die mit synthetischen Daten trainierte KI generalisierbar ist, sodass der Wechsel auf die Hardware ohne größere Verluste in der Performance erfolgen kann, lässt sich nicht zuverlässig vorhersagen. Da schon die pyhsikalische Simulation die Realität nicht perfekt abbilden kann und an verschiedenen Stellen Kompromisse eingehen muss, besteht die Gefahr, dass ein Wechsel von der Simulation zum realen Aufbau nicht mit dem erhofften Erfolg durchgeführt werden kann [100]. Aus diesen Gründen ist die direkte Verwendung eines Kamerabilds nicht optimal.

Statt des Kamerabilds kann die KI auch mit bereits vorverarbeiteten Informationen arbeiten. Als Eingang würden in diesem Fall die Eigenschaften der relevanten Objekte, also z. B. die Geschwindigkeit und Bewegungsrichtung des Balls, verwendet werden. Diese Methode lässt sich bei der Simulation besonders leicht umsetzen, da die benötigten Daten ohnehin vorliegen, weil anhand dieser Informationen erst das weitere Spielgeschehen bestimmt wird. Gleichzeitig ist die Berechnung der grafischen Ausgabe in diesem Fall nicht für das Training erforderlich. Erst zur Überprüfung der korrekten Funktionalität der KI durch einen Menschen müsste diese visualisiert werden. In diesem Fall reicht jedoch eine sehr einfache Darstellung, welche lediglich einen groben Überblick liefert.
Durch die Verwendung der Metadaten kann die Trainingszeit stark verringert werden, da in der Regel wesentlich weniger Eingänge als bei der Verarbeitung eines Bilds benötigt werden. Hierdurch kann auch das gewählte Netz weniger komplex sein. Statt einer hohen Anzahl an faltenden Schichten, die für die Bildverarbeitung innerhalb einer KI nötig sind, können bei der Verwendung von Metadaten wenige und kleine Schichten verwendet werden. So kann das Training einer KI für die CartPole Umgebung [35] innerhalb weniger Minuten durchgeführt werden [96], während für das Atari-Spiel Pong dieser Prozess über zehn Stunden dauern kann [70], da hier nicht die Metadaten, sondern die visuelle Ausgabe als Eingang der KI dient.

In einem ersten Schritt bietet es sich also an, das Training über die Metadaten einer Simulation durchzuführen. Damit die Ergebnisse der Simulation auch auf dem realen System anwendbar sind, muss die Simulation, wie zuvor beschrieben, anpassbar und möglichst genau sein. Da ein Programm, welches reale physikalische Eigenschaften beschreiben soll, recht schnell sehr komplex werden kann, ist es sinnvoll Anforderungen an dieses Programm aufzustellen.

5.2 Simulation

Da die physikalische Simulation des Tischkickers ein komplexer Schritt ist und einen hohen zeitlichen Aufwand benötigt, damit sie so genau wie möglich die Realität beschreibt, scheint auch hier eine Aufstellung der Anforderungen an diese sinnvoll.

5.2.1 Stakeholder

Zunächst werden die verschiedenen Stakeholder der Simulation samt ihrer Interessen analysiert. Hieraus lassen sich in den weiteren Schritten die genauen Anforderungen an die Simulation ableiten.

Ersteller der Arbeit

Der primäre Stakeholder der Simulation ist der Ersteller dieser Arbeit, denn seine Aufgabe ist die Programmierung der Simulation und der spätere Einsatz dieser für das Training.

Einerseits besteht aus Sicht des Programmierers das Interesse einer schnellen Lösung der Aufgabe ohne einen unnötig hohen zeitlichen Einsatz. Andererseits muss die erarbeite Lösung eine möglichst genaue und korrekte Beschreibung des realen Aufbaus sein. Anderenfalls verfehlt der Einsatz der Simulation seinen Zweck, da die erlernten Methoden der KI nicht auf dem echten Tischkicker übertragbar sind. Es muss also eine Abschätzung getroffen werden, wie komplex die Simulation sein muss, damit sie die Realität möglichst genau abbildet, aber nicht zu zeitintensiv in der Entwicklung ist.

Gleichzeitig ist der Ersteller dieser Arbeit auch die erste Person, die mit der Simulation ein Training der KI durchführt. Hierfür muss diese die nötigen Schnittstellen liefern, welche die benötigten Informationen weitergeben. Eine einfache Anwendbarkeit der Simulation ist also von hohem Interesse. Dazu gehört auch die Möglichkeit zur Anpassung verschiedener Parameter, welche abhängig von der später verwendeten Hardware sind.
Des Weiteren ist die Geschwindigkeit der Simulation essentiell für den Erfolg des Trainings, da mehr Durchläufe in geringerer Zeit simuliert werden können und so schneller eine möglichst optimale Lösung eintrainiert wird. Um die Funktionalität der Simulation und der KI zu überprüfen, muss der aktuelle Zustand dieser darstellbar sein. Dabei ist

eine perfekte Visualisierung jedoch nicht nötig, da nur die generelle Funktionsweise überprüft werden muss, während die optische Ansprechbarkeit kein wichtiges Kriterium ist. Zusätzlich sollte die grafische Ausgabe deaktivierbar sein, damit Ressourcen eingespart werden können und so der Trainingsprozess beschleunigt wird.

Spätere Benutzer

Da der Tischkicker später als Plattform für Studenten genutzt werden soll, um Techniken des Reinforcement Learnings zu erlernen, wird die Simulation auch für weitere Nutzer interessant. Natürlich bleiben die gleichen Interessen an Geschwindigkeit, Genauigkeit und Anpassbarkeit der Simulation, wie beim Ersteller der Arbeit. Gleichzeitig besteht aber ein Bedarf an eine gute Dokumentation und einfache Anwendbarkeit, damit der Fokus möglichst nur auf dem Teil des maschinellen Lernens gelegt werden kann.

Auftraggeber und Prüfer

Für Prof. Dr. Hensel besteht auch ein Interesse an der Simulation, dass diese für spätere Arbeiten noch verbessert oder erweitert werden kann. Dementsprechend muss diese möglichst modular und flexibel programmiert sein, um Änderungen leicht zu ermöglichen.
Ein kleines Interesse besteht auch an einer hochwertigen grafischen Ausgabe für Demonstrationszwecke. Zwar erfüllt eine einfache Ausgabe hier auch den Zweck, allerdings ist eine verbesserte Darstellung werbewirksamer. Die Priorität hierfür ist jedoch entsprechend gering, da eine verbesserte Darstellung auch nachträglich hinzugefügt werden kann.

5.2.2 Anwendungsfälle

Neben den Stakeholdern müssen auch die einzelnen Anwendungsfälle der Simulation analysiert werden. Aus diesen können ebenfalls wichtige Funktionalitäten und Anforderungen an die Simulation abgeleitet werden. Die verschiedenen Anwendungsfälle der Simulation sind in Abbildung 5.1 dargestellt.

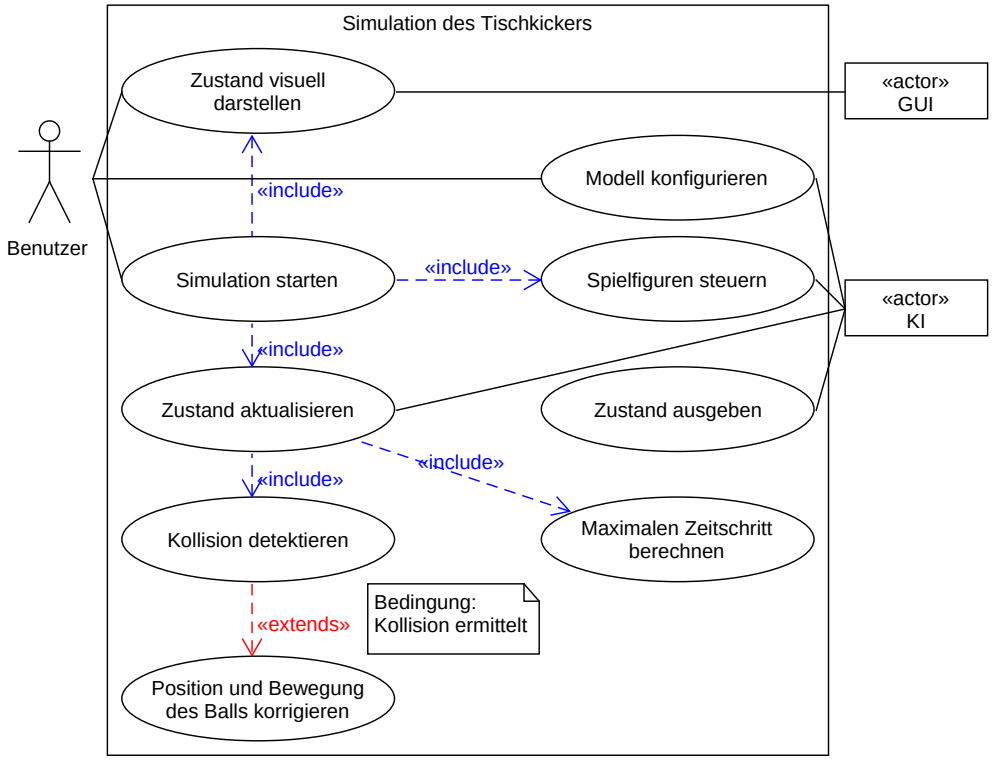


Abbildung 5.1: Anwendungsfall-Diagramm der Tischkickersimulation

Modell konfigurieren

Da die realen pyhsikalischen Vorgänge eines Tischkickerspiels präzise simuliert werden sollen, ist die genaue Anpassung der Simulation an die Hardware nötig. Dementsprechend muss die Form und Größe, sowie weitere Besonderheiten, wie angeschrägte Ecken einstellbar sein. Ähnlich gilt dies auch für die Stangen und Spielfiguren, welche in der richtigen Position und im besten Fall auch in der korrekten Form eingespeichert werden müssen. Da es generell sehr schwer ist, die realen Spielfiguren zu digitalisieren, kann hier auch eine angemessene Annäherung verwendet werden. Weiter muss auch der Ball anpassbar sein, indem Faktoren wie Radius, Gewicht und Material einstellbar sind.
Zuletzt müssen auch Eigenschaften wie die maximale Verschiebe- und Rotationsgeschwindigkeit der Motoren im fertigen Aufbau mit einberechnet werden. Anderenfalls funktionieren die eintrainierten Methoden möglicherweise nur in der Simulation.

Dieser Anwendungsfall kann vom Benutzer oder der KI mit den entsprechenden Eigen-

schaften aufgerufen werden. Die eingelesenen Werte werden dann angewendet und die Simulation entsprechend angepasst.

Simulation starten

Damit der Benutzer eine Simulation durchführen kann, muss er lediglich einen Start initialisieren. Durch die Auslösung dieses Anwendungsfalls soll eine Simulation automatisch durchgeführt werden. Hierbei werden die verschiedenen Anwendungsfälle der Simulation automatisch aufgerufen und ausgeführt.

Ausgelöst durch den Benutzer, wird in diesem Anwendungsfall die Steuerung der Simulation eines ganzen Spiels übernommen. Hierbei werden wiederholt die Anwendungsfälle Zustand aktualisieren, Spielfiguren steuern und Zustand visuell darstellen ausgeführt.

Spielfiguren steuern

Um ein Spiel realisieren zu können, müssen die Reihen von Spielfiguren von der KI gesteuert werden. Dabei müssen sowohl die Verschiebung als auch die Rotation der Reihen veränderbar sein. Gleichzeitig ist für eine feine Steuerung erforderlich, dass auch die Geschwindigkeit der Bewegung anpassbar ist. Erst bei Erfüllung dieser Anforderungen können verschiedene Taktiken angewendet werden, statt ein einfaches nach vorne Schlagen des Balls.

Dieser Anwendungsfall muss natürlich von der KI aufrufbar sein, damit sie die Spielfiguren von ihr gesteuert werden können. Es wird dabei die Geschwindigkeit der Reihen verändert, sodass bei der Aktualisierung des Zustands die gewünschte Bewegung ausgeführt wird.

Zustand aktualisieren

Das Herzstück der Simulation ist die Aktualisierung des Zustands. Hierbei werden für einen gewissen Zeitschritt die Position des Balls und der Spielfiguren entsprechend des aktuellen Zustands aktualisiert.

Durch die Simulation oder die KI kann die Aktualisierung eines Zustands angefragt

werden. Diese berechnet aus dem aktuellen Zustand den nächsten Zustand. Anschließend wird die Kollisionsdetektion und zuletzt die Berechnung des nächsten Zeitschritts aufgerufen.

Kollision detektieren

Beim Tischkicker ist ein wichtiger Aspekt die Kollision des Balls mit den Spielfiguren bzw. den Banden, denn erst so kann der Ball bewegt und auch abgewehrt werden. Bei einer Berührung des Balls mit dem restlichen Tischkicker muss dementsprechend eine Korrektur der Bewegungsrichtung und Geschwindigkeit vorgenommen werden. Zu der Kollisionsdetektion gehört auch die Erkennung von geschossenen Toren, da hier im Grunde eine Kollision berechnet wird.

Dieser Anwendungfall wird bei der Aktualisierung des Zustands aufgerufen und berechnet mögliche Kollisionen. Wenn eine Kollision erkannt wird, soll die Korrektur der Position und Bewegung des Balls aufgerufen werden.

Position und Bewegung des Balls korrigieren

Falls der Ball sich in der Simulation mit anderen Objekten überlappt, ist eine Kollision aufgetreten. In diesem Fall muss die Position und auch die Bewegungsrichtung, sowie die Geschwindigkeit des Balls angepasst werden.

Aufgerufen durch die Kollisionsdetektion korrigiert dieser Anwendungsfall die Position und Bewegungsrichtung des Balls nach einer Kollision und berücksichtigt auch Verluste durch den Aufprall bzw. Reibung.

Maximalen Zeitschritt berechnen

Für die Kollisionsdetektion muss der erlaubte Zeitschritt zwischen zwei Aktualisierungen des Zustands berechnet werden. So kann eine gewisse Genauigkeit gewährleistet werden, während die Simulation nicht unnötig langsam ist, beispielsweise wenn der Ball sehr langsam rollt.

Die Aktualisierung des Zustands ruft diesen Anwendungsfall immer nach der Kollisionsdetektion auf. In diesem Fall wird der maximal erlaubte Zeitschritt der nächsten Aktualisierung berechnet.

Zustand ausgeben

Damit die KI in das Spielgeschehen eingreifen kann, muss diese die Daten des aktuellen Zustands des Kickers erhalten. Die Daten sollen wie in Abschnitt 5.1 beschrieben nicht in Form eines Bilds, sondern als Metadaten der relevanten Objekte vorliegen. Eine Ausgabe dieser muss in regelmäßigen Abständen von der KI angefordert werden können.

Dieser Anwendungsfall wird von der KI aufgerufen und liefert die benötigten Daten des Zustands des Tischkickers. Die Daten werden zurück an die KI geschickt.

Zustand visuell darstellen

Ein Anwendungsfall ist die grafische Ausgabe der Simulation. Auf Wunsch eines Benutzers muss diese ein- bzw. ausschaltbar sein, damit eine Überprüfung der Funktionsweise der KI erfolgen kann, während beim Training Rechenzeit eingespart wird. Die Darstellung des Tischkickers muss jederzeit aufrufbar sein und den aktuellen Zustand ausgeben.

Der Benutzer kann diesen Anwendungsfall aufrufen um eine einfache grafische Darstellung des aktuellen Zustands zu bekommen. Die Ausgabe wird durch eine GUI realisiert.

5.2.3 Anforderungen

Die Anforderungen an die Simulation werden im folgenden Abschnitt aufgestellt, beleuchtet und priorisiert, damit ein klarer Rahmen für die Entwicklung dieser vorgegeben wird.

Funktionale Anforderungen

Die funktionalen Anforderungen an die Simulation sind in der Tabelle 5.1 aufgeführt.

Tabelle 5.1: Funktionale Anforderungen der Simulation

ID	Priorität	Anforderung
SF1	hoch	Simulation der Ballbewegung
SF2	hoch	Kollisionsdetektion zwischen Ball und Spielfiguren
SF3	hoch	Kollisionsdetektion zwischen Ball und Spielfeld
SF4	hoch	Erkennung von Toren
SF5	hoch	Steuerung der Rotation der Spielfiguren
SF6	hoch	Steuerung der Verschiebung der Spielfiguren
SF7	niedrig	Einstellbarkeit der Größe, Beschaffenheit und des Gewichts des Spielballs
SF8	niedrig	Einstellbarkeit der Größe und Form des Tischkickers
SF9	niedrig	Einstellbarkeit der Anzahl und Position der Reihen und Figuren
SF10	niedrig	Einstellbarkeit der maximalen Verfahrstrecke der Verschiebung
SF11	niedrig	Einstellbarkeit der maximalen Geschwindigkeit der Rotation und Verschiebung
SF12	hoch	Anbindungsmöglichkeit zur Steuerung durch die KI
SF13	hoch	Ausgabe relevanter Informationen an die KI
SF14	hoch	Einstellbare visuelle Ausgabe des aktuellen Zustands

Die Simulation eines Tischkickerspiels ist vor allem von der Bewegung des Balls und wie dieser mit dem Tischkicker interagiert abhängig. So sind die Anforderungen SF1-4 an den Spielball gebunden. Kollisionen mit Spielfiguren bzw. dem Tischkicker müssen erkannt werden und die Bewegung des Balls entsprechend geändert werden. Gleichzeitig können auf diese Weise auch Tore erkannt werden, welche letztendlich das Spiel entscheiden.
Um in das Spielgeschehen eingreifen zu können, müssen die Verschiebung und Rotation der Reihen entweder durch eine manuelle Eingabe oder durch die KI steuerbar sein. Dies spiegelt sich in den Anforderungen SF5-6 wieder. Die bisher genannten Anforderungen bilden die Grundfunktionen des Tischkickers und sind somit unverzichtbar für die Simulation.
Damit die Simulation nicht nur für einen speziellen Fall funktioniert, ist eine Möglichkeit zur Einstellung der verschiedenen Eigenschaften vom Tischkicker und der eingesetzten Hardware erforderlich. Dabei ist die Größe und Form (also ob es z. B. angehobene Ecken gibt) des Tisches, sowie der Durchmesser und die Beschaffenheit des Balls entscheidend. Denn z. B. beim Einsatz von Bällen mit verschiedenen Materialien ergeben sich auch unterschiedliche Verhaltensweisen bei der Kollision mit Spielfiguren oder dem Boden. Wichtig ist auch, dass die richtige Anzahl und die Positionen der Reihen und Spielfiguren anpassbar an den jeweiligen Einsatzfall ist. Hierbei ist auch der maximale Bewegungsraum der Verschiebung interessant. Damit die ausgeführten Aktionen in der

Simulation auch realistisch auf dem realen System ausführbar sind, müssen die Bewegungsgeschwindigkeiten an die der Motoren anpassbar sein. Alle genannten Funktionen zur Einstellbarkeit der Simulation werden in den Anforderungen SF7-11 festgehalten und erhalten eine niedrige Priorität.
Eine hohe Priorität wird hingegen der Möglichkeit zur Anbindung der KI zugeschrieben, da ohne diese Funktion kein Training möglich ist. Hierzu gehört auch die Ausgabe der relevanten Informationen, als Eingang der KI.
Um die Trainingsfortschitte sichtbar zu machen, soll zudem eine visuelle Ausgabe realisiert werden, welche allerdings auch nach Bedarf abgestellt werden kann, damit der rechenintensive Prozess des Trainings nicht durch die Ausgabe verlangsamt wird.

Nicht-funktionale Anforderungen

In der Tabelle 5.2 werden die nicht-funktionalen Anforderungen an die Simulation aufgelistet und nach Prioritäten eingestuft.

Tabelle 5.2: Nicht-funktionale Anforderungen der Simulation

ID	Priorität	Anforderung
SN1	niedrig	Objektorientierte Programmierung
SN2	niedrig	Programmierung in Python
SN3	hoch	Rechenzeit der Simulation kleiner als simulierte Zeit
SN4	hoch	Kollisionsdetektion nach maximal $\Delta s = 2\ mm$ Bewegung des Balls
SN5	hoch	3-dimensionale Simulation
SN6	niedrig	Berücksichtigung von Verlusten durch Reibung

Die nicht-funktionalen Anforderungen SN1 und SN2 beziehen sich auf die Art der Programmierung der Simulation. Durch den objektorientierten Ansatz in Python soll die Entwicklung möglichst einfach und zeitsparend sein. Zudem ermöglicht dies einen leichten Einstieg einer dritten Person in die Weiterentwicklung der Simulation bzw. der KI. Die Priorität ist allerdings eher niedrig, da die Funktionalität im Vordergrund steht.
Ein wichtiger Faktor neben der Funktionalität ist die Geschwindigkeit der Berechnung. Nach SN3 soll die Rechenzeit kleiner als die simulierte Zeit sein. So lohnt sich der Einsatz der Simulation noch mehr im Gegensatz zu dem Training auf der Hardware. Die Priorität hierfür ist dementsprechend hoch.
Ebenfalls eine hohe Priorität hat die regelmäßige Kollisionsdetektion des Balls. Dieser darf sich zwischen den Abtastzeitpunkten nicht weiter als $\Delta s = 2$ mm bewegen, denn nur so kann gewährleistet werden, dass kein Kontakt übersehen wird.

Für die Kollision ist ebenfalls wichtig, dass die Simulation im 3-dimensionalen Raum durchgeführt wird, denn nur so kann eine genaue Kollisionsdetektion stattfinden. Anderenfalls wäre beispielsweise das Durchlassen eines nach vorne geschossenen Balls unter den Spielfiguren nicht möglich. Deshalb muss die Simulation drei Dimensionen besitzen. In der Realität ist die Geschwindigkeit des Balls begrenzt durch verschiedene Verluste wie z. B. Reibung. Für eine möglichst genaue Nachbildung der Realität sollten in der Simulation diese Verluste ebenfalls berücksichtigt werden. Eine gute Annäherung kann allerdings auch ohne Verluste erreicht werden, wodurch die Priorität eher gering ist.

5.2.4 Konzept der Simulation

Nachdem die Anforderungen aufgestellt sind, kann ein Konzept bzw. ein Ablaufplan für die Simulation aufgestellt werden. Die Simulation ist der Grundstein für das erfolgreiche Training einer KI und muss dementsprechend hohen Anforderungen genügen. In Abbildung 5.2 ist der grobe Ablaufplan der Simulation geschildert.

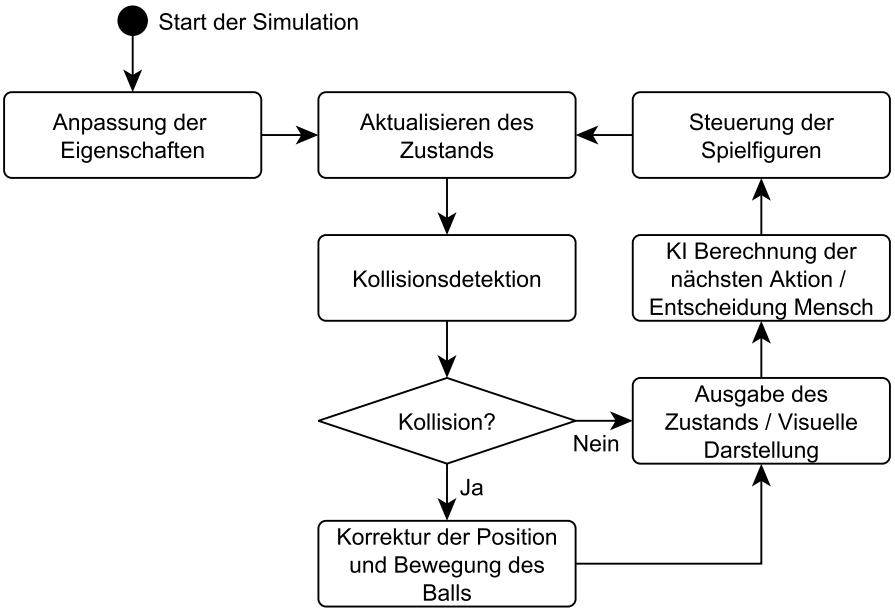


Abbildung 5.2: Ablaufdiagramm der Tischkickersimulation

Die Simulation beginnt mit dem Einstellen der Parameter des Tischkickers, der Spielfiguren und des Balls. Ein gewünschter Anfangszustand wird so hergestellt.

Anschließend beginnt die eigentliche Simulation in einer Schleife, die nach einer gewissen Anzahl an Durchläufen oder nach einem erzielten Tor abgebrochen wird.
Wie schon in den Anwendungsfällen der Simulation angedeutet, muss mit jedem Zeitschritt die Position des Balls und der Spielfiguren aktualisiert werden. Zusätzlich muss hier die Berechnung des maximalen Zeitschritts erfolgen. Zusammen ist dies der Hauptschritt der Simulation, welcher jedoch noch um die Kollisionsdetektion erweitert ist. Bei einer erfolgreich detektierten Kollision muss zusätzlich eine Korrektur der Position und Bewegung des Balls erfolgen.
Anschließend wird der Zustand des Spiels an den Benutzer bzw. die KI weitergegeben, indem eine grafische Ausgabe oder im Fall der KI die relevanten Daten weitergegeben werden. Der Benutzer bzw. die KI kann dadurch die bestmögliche nächste Aktion bestimmen und die Steuerung der Spielfiguren übernehmen.
Damit die Anforderung SN4 erfüllt wird, müssen möglicherweise die Schritte der Ausgabe der Daten, die Berechnung der nächsten Aktion und die Steuerung der Spielfiguren übersprungen werden. Dies liegt daran, dass bei hohen Geschwindigkeiten des Balls eine niedrige Abtastrate benötigt wird und diese bei der Abtastung des realen Systems nicht erreicht werden kann. Auch die visuelle Ausgabe ist durch eine hohe Laufzeit beschränkt, sodass hier die Geschwindigkeit der Simulation erhöht wird, wenn die FPS nach unten angepasst werden.

Damit die Entwicklung möglichst wenig Zeit in Anspruch nimmt, soll die Simulation möglichst einfach gehalten werden und erst bei Bedarf angepasst werden. Dieser Bedarf kann geprüft werden, indem die KI auf die Hardware übertragen wird und ein Vergleich zwischen der Leistung in der Simulation und auf der Hardware aufgestellt wird. So kann beispielsweise ohne Berücksichtigung der Reibung und Gravitation gearbeitet werden, wenn die Übertragung der KI ohne größere Verluste in der Qualität möglich ist.

5.3 Auswahl der Hardware

Mit dem Konzept der KI und dem Training dieser in einer Simulation, kann eine geeignete Auswahl der Hardware für einen ersten Aufbau getroffen werden. Dabei müssen die verschiedenen verfügbaren Optionen miteinander hinsichtlich ihrer Eigenschaften und auch ihres Preises verglichen werden.

5.3.1 Aufbau

Zunächst lohnt sich ein Blick auf das geplante System. Hierbei soll im ersten Aufbau ein Tischkicker entwickelt werden, auf welchem die grundsätzliche Funktionalität der Hard- und Software überprüft werden kann. So können nötige Anpassungen gemacht werden, bevor ein endgültiger Aufbau realisiert wird, der dann auch den aufgestellten Anforderungen entspricht.
In diesem ersten Aufbau soll lediglich eine der Stangen automatisiert werden, sodass die KI die Steuerung übernehmen kann. So kann beispielsweise das Verteidigen und auch das Tore Schießen getestet werden. Ein komplettes Tischkickerspiel kann so jedoch nicht realisiert werden.
Hierfür werden zwei Aktoren benötigt, die einerseits die Verschiebung und andererseits die Rotation ausführen.
Damit die Steuerung auf der Hardware mit dem Ball interagieren kann, muss der Zustand des Kickers ermittelt werden. Für die Verarbeitung der Sensordaten wird eine Plattform benötigt, die zudem auch die Ansteuerung der Motoren übernehmen kann.
Die einzelnen Komponenten und ihr Zusammenspiel untereinander sind in Abbildung 5.3 dargestellt.

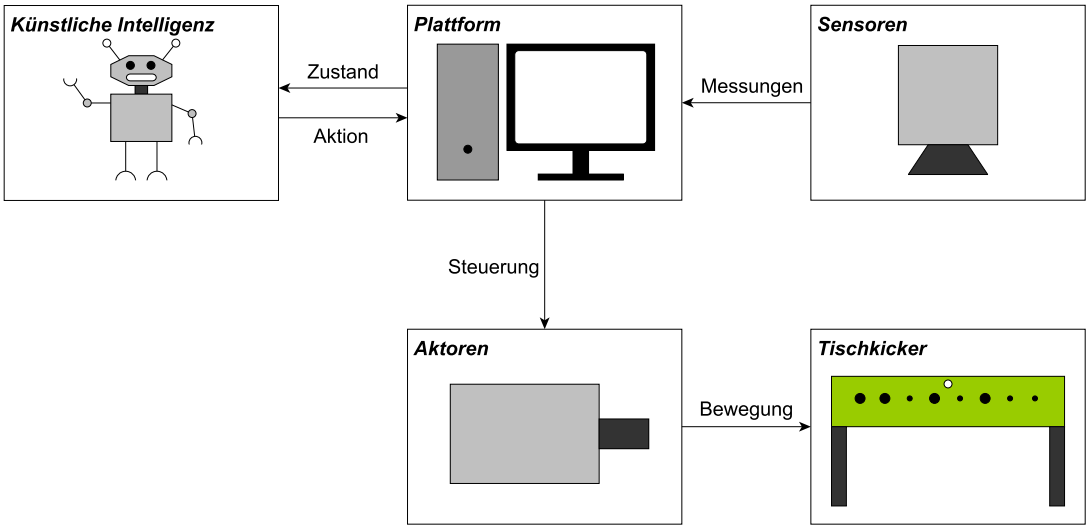


Abbildung 5.3: Zusammenspiel der Komponenten des automatisierten Tischkickers

5.3.2 Tischkicker

Die Auswahl der Hardware beginnt mit dem Tischkicker, denn die restlichen Komponenten müssen sich den Eigenschaften des Tischkickers anpassen. Es gibt viele verschiedene Arten von Kickern. Diese unterscheiden sich zum Beispiel vom Material, ihrer Form oder auch dem generellen Aufbau. Aus den Anforderungen geht hervor, dass der Tischkicker hochklappbar und vor allem relativ leicht, aber stabil sein muss. Ansonsten bestehen keine Vorgaben an die genaue Form. Die Auswahl kann also auf leichte, hochklappbare Modelle ausgedünnt werden und dann nach persönlichen Präferenzen bzw. dem Preis erfolgen.
In einem ersten Schritt wird hier jedoch ein kleineres Modell ausgewählt, welches zudem weniger Reihen und Spielfiguren besitzt als ein normales Modell. Dadurch ergibt sich ein weniger komplexer Zustands- und Aktionsraum, sodass das Training schneller und einfacher ist. Auf diese Weise kann zunächst mit weniger Ressourcen die generelle Funktionalität der KI in der Simulation und ihre Übertragbarkeit auf ein reales System geprüft werden. Der Minikicker ist somit ein Zwischenschritt, welcher die Komplexität der Aufgabe etwas erleichtert und zudem nicht die gleichen Anforderungen an den Aufbau besitzt wie die endgültige Version. Dadurch können auch verschiedene Konzepte im kleinen Maßstab ausprobiert werden. Bei der Auswahl des richtigen Minikickers ist vor allem der Preis entscheidend, während der Formfaktor vernächlässigbar ist, solange die generelle Funktionsweise ähnlich wie bei einem großen Tischkicker ist und mindestens zwei Reihen pro Seite verfügbar sind.

Abbildung 5.4: Ausgewählter Mini-Tischkicker

Die Auswahl ist auf den Tischkicker in Abbildung 5.4 gefallen, welcher die benötigten Anforderungen erfüllt und bei einem Gebrauchtmarkt zum Verschenken angeboten wur-

de. Der Kicker ist 40 cm lang, 30 cm breit und hat pro Spieler zwei Reihen mit je drei Figuren.

5.3.3 Aktoren

Um die von der KI errechneten Aktionen und Spielzüge auf dem Tischkicker umzusetzen, müssen die Stangen einer Mannschaft entsprechend vom Computer bewegt werden. Die Bewegung unterteilt sich dabei in die Verschiebung und die Rotation der Stangen. Die Verschiebung dient der genauen Positionierung der Spielfiguren, um Bälle abzuwehren oder eine Figur in Schussposition zu bringen. Die Rotation hingegen wird vor allem zum Schießen des Balls verwendet.
Beide Bewegungsrichtungen müssen unabhängig voneinander und gleichzeitig ansteuerbar sein. Dabei ist eine hohe Präzision erforderlich um Techniken wie z. B. den Pull Shot zu erlauben. Bei dieser Schusstechnik ist auch die Geschwindigkeit der Bewegung entscheidend, da sie darauf abzielt an der Verteidigung des Gegners vorbei zu schießen bevor dieser reagieren kann. Um diese und auch andere Schussarten bestmöglich umsetzen zu können und gleichzeitig selbst nicht hierfür anfällig zu sein, ist also auch eine möglichst schnelle und kraftvolle Bewegung der Stangen erforderlich.

Antriebsart

Für die verschiedenen Bewegungen kommen mehrere Antriebsarten in Frage. Zum einen der elektrischer Antrieb über Elektromotoren oder zum anderen ein fluidtechnischer Antrieb durch Pneumatik oder Hydraulik. Ein Vergleich der Techniken ist in Tabelle 5.3 aufgeführt.

Tabelle 5.3: Vergleich der verschiedenen Antriebstechniken und ihrer Eigenschaften

	Elektromotoren	Pneumatik	Hydraulik
Kosten	mittel	mittel	hoch
Zuverlässigkeit	hoch	mittel	hoch
Kraft	niedrig	mittel	hoch
Geschwindigkeit	mittel	hoch	mittel
Genauigkeit	hoch	niedrig	mittel

Hydraulik ist als Antriebsart für den Tischkicker nicht die optimale Lösung. Zwar können hohe Kräfte erzeugt werden, die mit einer guten Präzision ausgeführt werden, allerdings

ist ein hydraulisches System recht komplex und dementsprechend teuer. Hydraulik wird aufgrund seiner Eigenschaften in der Regel für schwere Geräte wie z. B. Bagger oder Pressen eingesetzt [41]. Aus diesem Grund ist der Markt für anpassbare kleine Hydrauliksysteme überschaubar, sodass es keine passenden Lösungen für den Tischkicker zu kaufen gibt. Der große Vorteil der hohen Kraftentwicklung bei kleiner Bauform ist durch die gegebenen Anforderungen an den Tischkicker nicht zwingend notwendig, da die Stangen und der Spielball verhältnismäßig leicht sind.

Ein weiterer Ansatz wäre die Lösung über Pneumatik. Statt mit einer Flüssigkeit wie bei der Hydraulik wird der Druck hier durch Luft erzeugt. Dadurch ist das System etwas einfacher und günstiger, während auch die Wartung wesentlich leichter ist. Gleichzeitig können mittels Pneumatik sehr hohe Geschwindigkeiten mit einer hohen Beschleunigung erreicht werden. Deshalb eignet sich das System mit seiner wesentlich geringeren aber ausreichenden Kraftentwicklung für viele kleinere Anwendungen. Dementsprechend sind auch verschiedene anpassbare pneumatische Lösungen erhältlich, die für den Tischkicker infrage kommen.
Ähnlich wie bei der Hydraulik wird bei der Pneumatik allerdings ein Kompressor mit Druckluftspeicher benötigt, der den benötigten Druck erzeugt. Hierdurch steigt die Komplexität des Systems und vor allem die Größe sowie das Gewicht.
Die Pneumatik hat jedoch den entscheidenden Nachteil, dass sie sich nur sehr unzuverlässig regeln lässt. Die Luft verteilt sich nicht gleichmäßig in den Zylindern, wie die bei der Hydraulik verwendeten Flüssigkeiten, wodurch eine ungleichmäßige Bewegung entstehen kann. Die Geschwindigkeit wird über Ventile eingestellt, wobei es jedoch keine günstigen, elektrisch einstellbaren Ventile gibt, sodass stets nur mit einer festen Geschwindigkeit gearbeitet werden kann. Deshalb sind pneumatische Systeme am besten für Anwendungen geeignet, bei denen immer ein fester Weg vom Anfang bis Ende zurückgelegt wird, sodass keine genaue Positionierung erforderlich ist. Da dies beim Tischkicker nicht der Fall ist, scheint der Einsatz von Pneumatik ungeeignet.

Der elektrische Antrieb ist die einfachste der aufgezeigten Methoden und benötigt auch deshalb die geringste Wartung. Je nach Elektromotor wird möglicherweise noch ein Steuergerät benötigt, welches die Kosten und die Komplexität leicht erhöht, hierfür jedoch auch eine exakte Positionierung zulässt. Die Geschwindigkeit und Kraft von Elektromotoren ist im Vergleich zu den anderen Techniken etwas geringer. Für den Einsatz am Tischkicker sollte dies jedoch durch die Wahl eines geeigneten Motors reichen.

Aus den genannten Gründen scheint der Einsatz eines Elektromotors ein geeigneter Kompromiss für den Tischkicker.

Art der Elektromotoren

Bei der Auswahl des richtigen Elektromotors kommen vier verschiedene Arten von Antrieben in Frage: Bürstenbehaftete Gleichstrommotoren (BDC), Bürstenlose Gleichstrommotoren (BLDC), Servomotoren und Schrittmotoren. Für eine bessere Übersicht werden die genannten Elektromotoren in Tabelle 5.4 miteinander verglichen.

Tabelle 5.4: Vergleich der verschiedenen Elektromotoren und ihrer Eigenschaften

	BDC	BLDC	Schrittmotor	Servomotor
Kosten	niedrig	mittel	mittel	hoch
Kraft	niedrig	mittel	hoch	hoch
Geschwindigkeit	hoch	hoch	niedrig	mittel
Genauigkeit	niedrig	niedrig	hoch	hoch

Bürstenbehaftete Gleichstrommotoren haben den einfachsten Aufbau und benötigen als einzige Antriebsart keine weitere Steuereinheit, wodurch sie die günstigste Variante sind. Ihre maximale Geschwindigkeit ist hoch, allerdings sind sowohl die Kraft als auch die Genauigkeit sehr gering, sodass sie nicht optimal für den Einsatz am Tischkicker sind.
Eine Verbesserung hierzu sind die bürstenlosen Gleichstrommotoren, welche schneller, effizienter und kraftvoller sind als die bürstenbehaftete Motoren [39]. Es steigen hierfür allerdings die Komplexität und Kosten. Ausgelegt sind BLDC-Motoren in der Regel für sehr hohe Drehzahlen, sodass ohne ein Getriebe die Verwendung für die Rotation der Reihen nicht möglich ist. Gleichzeitig ist bei einer niedrigen Drehzahl das Drehmoment nur sehr gering.

Eine weitere Möglichkeit bilden die Schrittmotoren, welche in sehr feinen Schritten gesteuert werden können und dabei eine recht hohe Kraft aufbringen. Von Vorteil ist auch, dass der Motor im Stillstand ein hohes Haltemoment besitzt, welches überhaupt erst das Verteidigen von Schüssen möglich macht, denn ohne dieses Haltemoment kann ein harter Schuss eine Reihe leicht rotieren, sodass die Figuren kein Hindernis mehr darstellen. Nachteil von Schrittmotoren ist bei der Verschiebung allerdings die relativ geringe Rotationsgeschwindigkeit und die mit der Drehzahl abnehmende Kraft. Durch die Auswahl eines geeigneten Zahnrads für den Riemen und eines starken Motors kann dem jedoch

auch entgegengewirkt werden. Ein letztes Risiko besteht durch einen möglichen Schrittverlust, welcher beispielsweise bei zu starker Beschleunigung entstehen kann.

Eine Verbesserung zu Schrittmotoren sind Servomotoren. Diese sind was die maximale Drehzahl und Beschleunigung angeht eine bessere Lösung als die Schrittmotoren. Zudem beseht kein Risiko eines Schrittverlustes. Stattdessen misst der Motor seinen eigenen Zustand und versucht bei einer Abweichung zwischen gewünschter und tatsächlicher Position diesen Unterschied auszugleichen. Servomotoren kommen jedoch auch mit zwei Nachteilen. Die Kraftdichte ist geringer als beim Schrittmotor, d. h. ein Servomotor mit gleichem Drehmoment ist wesentlich schwerer und größer. Außerdem ist der Preis meist höher als für Schrittmotoren. Selbst für den Minikicker ist die Verwendung von ausschließlich Servomotoren nicht sinnvoll, da auch hier insgesamt vier Motoren für die zwei Computergesteuerten Reihen benötigt werden. Zusammen mit den Schienen und anderen Kleinteilen könnte so das Budget schnell knapp werden.

Es zeigt sich also, dass es keine perfekte Universallösung für das dargestellte Problem gibt. Stattdessen sind mehrere Ansätze möglich, die alle mit gewissen Vor- und Nachteilen einhergehen. Ein geeigneter Kompromiss für die Auswahl ist ein Closed-Loop Schrittmotor. Dieser vereint die Vorteile des Schrittmotors und des Servomotors. So kann ein Schrittverlust wie beim Servo detektiert und ausgeglichen werden. Gleichzeitig ist der Preis deutlich geringer als für einen gleichwertigen Servomotor.
Für die genaue Auswahl des Schrittmotors muss zunächst eine Abschätzung der benötigten Kraft gemacht werden. Hierfür muss jedoch erst die Anbringung der Motoren geklärt sein, da je Konzept unterschiedliche große Kräfte für die Bewegung der Motoren benötigt werden.

Anbringung der Elektromotoren

Die Verschiebung und die Rotation der Stangen müssen für ein möglichst erfolgreiches Spiel unabhängig voneinander in einer hohen Geschwindigkeit mit ausreichend Kraft erfolgen. Für eine gleichzeitige Bewegung muss entweder einer der zwei Motoren für die unterschiedlichen Bewegungsrichtungen stets vom anderen mitbewegt werden oder es bedarf einer speziellen Vorrichtung. Diese muss regeln, dass die Befestigung für die Rotation während der Verschiebung an der gleichen Stelle bleibt und trotzdem jederzeit verwendet werden kann. Denkbar wäre hier z. B. eine Stange mit einer Einkerbung entlang der Ver-

schiebungsrichtung. Die Befestigung für die Rotation wäre dann ein Ring, welcher um die Stange gelegt wird und eine Nut passend zu der Einkerbung der Stange hat (siehe Abbildung 5.5). Im Optimalfall besteht die Nut aus einer Kugelkette, sodass Reibung und somit Verluste vermieden werden. So kann die Stange entlang der Verschiebungsrichtung bewegt werden, während die Befestigung für die Rotation an der gleichen Stelle bleibt und die Rotation zeitgleich ausgeführt werden kann. Der klare Vorteil dieses Systems ist, dass keine schweren Motoren und Kabel bewegt werden müssen. Dadurch können schnellere und kraftvollere Bewegungen durchgeführt werden, bzw. kleine Motoren verwendet werden.

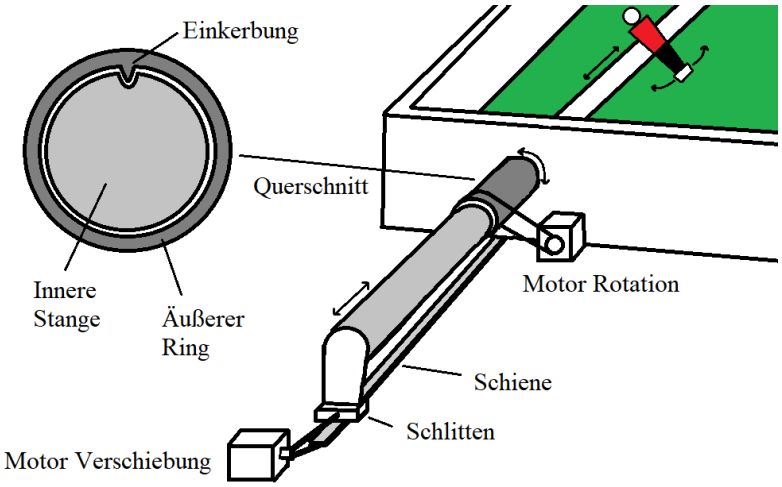


Abbildung 5.5: Konzept der angepassten Stange mit zwei stationären Motoren

Die Herstellung einer solchen speziellen Stange ist relativ komplex und daher schwierig umzusetzen. Dies auch nicht zuletzt, weil die Stabilität der Stange verringert wird, wenn eine Einkerbung eingearbeitet wird. Aus diesem Grund wird dieser Ansatz zunächst verworfen.
Stattdessen kann ganz klassisch gearbeitet werden, indem der Motor für die Rotation auf dem Schlitten für die Verschiebung steht. Dieses Verfahren hat allerdings den Nachteil, dass das Gewicht des Motors für die Rotation stets vom Motor für die Verschiebung mitbewegt werden muss. Dies sorgt für dafür, dass die Verschiebung wesentlich langsamer sein wird bzw. einen stärkeren Motor benötigt.

Umsetzung der Verschiebung

Ein Problem beim Elektromotor ist, dass dieser in der Regel nur eine radiale Bewegung zulässt. Für die Verschiebung der Stangen muss daher die Rotation der Elektromotoren in eine lineare Bewegung umgewandelt werden. Hierfür kommen zwei verschiedene und häufig eingesetzte Verfahren in Frage. Zum einen der Spindelantrieb und zum anderen der Antrieb mittels Zahnriemen. Die wichtigsten Eigenschaften werden in Tabelle 5.5 verglichen.

Tabelle 5.5: Vergleich der verschiedenen Antriebstechniken zum Umwandeln der Rotation eines Elektromotors in eine lineare Bewegung

	Spindelantrieb	Zahnriemenantrieb
Kosten	mittel	mittel
Kraft	hoch	mittel
Geschwindigkeit	niedrig	mittel
Genauigkeit	hoch	mittel

Insgesamt sind die beiden Arten der Übertragung relativ ähnlich vom Kostenaufwand und Formfaktor. Allerdings ist der Spindelantrieb je nach Art des Gewindes etwas langsamer, sprich es werden mehr Umdrehungen des Elektromotors benötigt, um eine bestimmte Strecke zurückzulegen. Damit geht jedoch auch eine höhere Kraftentwicklung und eine bessere Genauigkeit als beim Antrieb über Zahnriemen einher. Zwar sind auch die Genauigkeit und Kraft für die Bewegung entscheidend, allerdings muss die Priorität auf der Geschwindigkeit liegen, da der Ball recht hohe Geschwindigkeiten erreichen kann, während die Spielfiguren und der Ball relativ leicht sind und somit nur verhältnismäßig wenig Energie benötigt wird, um diese zu bewegen. Zudem kann durch die Wahl eines größeren und stärkeren Motors die Kraft angepasst werden, während es nur ein sehr begrenztes Angebot an unterschiedlichen Spindeln gibt. Ein weiterer Vorteil der Zahnriemen ist die hohe Flexibilität. Zum einen können die Zahnräder nach Bedarf dimensioniert und verhältnismäßig leicht ausgetauscht werden, zum anderen ist auch die Länge der Führung beliebig anpassbar und nach oben offen, während bei Spindelantrieben beide Faktoren von der Spindel abhängig sind und diese teuer und nicht einfach austauschbar ist. Um dem Verlust der Genauigkeit entgegenzuwirken, können Motoren verwendet werden, die eine feine Steuerung über Microstepping ermöglichen.
Für den ersten Aufbau wird ein GT2 Zahnriemen verwendet, da er hohen Belastungen standhält, günstig ist und ein großes Angebot an passenden Zahnrädern besitzt.

Auswahl der Linearführung

Für den Antrieb über einen Zahnriemen muss auch eine Linearführung eingeplant werden, auf welcher der Rotationsmotor angebracht wird. Sie sollte einen möglichst geringen Verschiebewiderstand haben und hohe Geschwindigkeiten und Beschleunigungen zulassen. Tabelle 5.6 zeigt einen Vergleich verschiedener Linearführungsarten.

Tabelle 5.6: Vergleich der verschiedenen Linearführungsarten

	Kugelführung	Rollenführung	Gleitführung	Kontaktlos
Kosten	mittel	mittel	niedrig	hoch
Geschwindigkeit	hoch	hoch	niedrig	hoch
Beschleunigung	hoch	mittel	niedrig	hoch
Widerstand	niedrig	niedrig	hoch	niedrig
Wartung	mittel	niedrig	niedrig	niedrig

Im Vergleich der verschiedenen Führungsarten schneiden die kontaktlosen Führungen am besten ab. Sie haben kleine Verschiebewiderstände und lassen hohe Geschwindigkeiten zu, allerdings ist ihr Preis auch wesentlich höher und ihre Installation bedarf weiterer Komponenten, weshalb sie hier nicht weiter berücksichtigt werden.

Die nächstbesten Führungen sind die Rollen- und Kugelführungen. Sie lassen ebenfalls hohe Geschwindigkeiten in der Verschiebung zu und haben dabei einen geringen Widerstand. Ihre Kosten sind moderat, sodass sie für die Anwendung am Tischkicker optimal geeignet sind.

Da die Gleitführungen durch ihren hohen Verschiebewiderstand und die geringe maximale Geschwindigkeit nicht in Frage kommen, muss eine Auswahl zwischen einer passenden Kugel- oder Rollenführung gemacht werden. Tabelle 5.7 zeigt drei verschiedene passende Modelle, aus denen eine Auswahl getroffen werden muss.

Tabelle 5.7: Vergleich der verschiedenen passenden Linearführungen

	ARC 25 FN	MGW15H	LWK 6-24
Art	Kugelführung	Kugelführung	Rollenführung
Kosten	87 €	66 €	82 €
Geschwindigkeit	bis 10 $\frac{\mathrm{m}}{\mathrm{s}}$	k. A.	k. A.
Beschleunigung	bis 500 $\frac{\mathrm{m}}{\mathrm{s}^2}$	k. A.	k. A.
Widerstand	k. A.	k. A.	k. A.

Preislich liegen die drei Optionen im ähnlichen Segment. Die Preise sind dabei auf die Angaben im DOLD-Mechatronik Onlineshop [3] angepasst und beinhalten je einen Führwagen und 30 cm der entsprechenden Linearführung.
Ein Problem bei der Auswahl sind die fehlenden Angaben bei den meisten Angeboten. Aus diesem Grund wird hier der etwas teurere und größere Linearführwagen ARC 25 FN ausgewählt, da er als einziges ein gut dokumentiertes Datenblatt besitzt. Hierbei wird auch der etwas höhere Preis in Kauf genommen sowie der Fakt, dass die Kugelführung gelegentlich geschmiert werden muss.

Umsetzung der Rotation

Die Rotation ist wesentlich einfacher umsetzbar als die Verschiebung, da keine Umwandlung der Bewegung der Motoren nötig ist. Diese können stattdessen direkt auf dem Schlitten der Linearführung stehen und mit den Stangen verbunden werden. Gegebenenfalls kann hier noch über ein Riemengetriebe ein passendes Übersetzungsverhältnis eingebaut werden, welches z. B. die Geschwindigkeit der Bewegung erhöht. So lässt sich sehr einfach die Umsetzung der Schussbewegung implementieren.

Dimensionierung der Elektromotoren

Es gilt jetzt die richtige Dimensionierung der Motoren anhand des benötigten Drehmoments zu finden. Für die Verschiebung kann die benötigte Kraft etwa mit

$$F_V = \frac{m_{Stange,ges} \cdot \Delta v_{Stange,max}}{\Delta t} \approx \frac{1,5 \text{ kg} \cdot 2\frac{\text{m}}{\text{s}}}{0,20 \text{ s}} = 15 \text{ N} \tag{5.1}$$

abgeschätzt werden. Hierfür wird die ungefähre Masse der Reihe samt Schlitten und Motor mit $m_{Stange,ges} \approx 1.5$ kg angenähert. Diese Masse muss nach den aufgestellten Anforderungen N6 und N9 innerhalb von $\Delta t \approx 0.2$ s auf die maximale Verschiebegeschwindigkeit $\Delta v_{Stange,max} \approx 2\frac{\text{m}}{\text{s}}$ gebracht werden. Aus diesem Wert kann mithilfe des Radius des Zahnrads $r_V \approx 0,04$ m das nötige Drehmoment

$$M_V = F_V \cdot r_V \approx 15 \text{ N} \cdot 0,04 \text{ m} = 0,6 \text{ Nm} = 60 \text{ Ncm} \tag{5.2}$$

berechnet werden. Der ausgewählte Motor muss also mindestens ein Drehmoment von $M_V \geq 60$ Ncm haben. Ein gewisser Spielraum ist dabei wichtig, da Schrittmotoren bei höheren Drehzahlen Drehmoment einbüßen.

Die Rotation lässt sich nicht so gut auf die gleiche Weise annähern, da z. B. das Gewicht der Spielfiguren sehr ungleichmäßig verteilt ist. Grob kann aber eine ähnliche Abschätzung getroffen werden, wenn die Anforderung N7 beachtet wird, die das Schießen des Balls mit bis zu $\Delta v_{Ball} = v_{Ball,test} = 5\ \frac{\mathrm{m}}{\mathrm{s}}$ fordert. Das Gewicht des Balls beträgt rund $m_{Ball} \approx 10$ g, um aber auch die Stange zu berücksichtigen, wird hier mit $m_{mix} \approx 150$ g gearbeitet. Mit einem Radius von rund $r_R \approx 0,1$ m ergibt sich ein Drehmoment von

$$F_R = \frac{m_{mix} \cdot \Delta v_{Ball}}{\Delta t} \approx \frac{0,15\ \mathrm{kg} \cdot 3\frac{\mathrm{m}}{\mathrm{s}}}{0,10\ \mathrm{s}} = 4,5\ \mathrm{N} \tag{5.3}$$

$$M_R = F_R \cdot r_R \approx 4,5\ \mathrm{N} \cdot 0,10\ \mathrm{m} = 0,45\ \mathrm{Nm} = 45\ \mathrm{Ncm}. \tag{5.4}$$

In beiden Fällen kann also ein Motor mit einem Drehmoment von etwa $M \approx 1$ Nm verwendet werden, um genügend Spielraum für verschiedene Verluste zu haben, ohne dabei überdimensioniert zu sein.

Auswahl des Elektromotors

Nach der Konzeption der Bewegung der Stangen mit den Spielfiguren kann die Auswahl der Schrittmotoren getroffen werden. In Tabelle 5.8 werden mögliche Optionen dargestellt. Die Preise und Informationen der Tabelle sind aus [3] entnommen.

Tabelle 5.8: Vergleich verschiedener Schitt- und Servomotoren

	103-H7126-1740	iHSS57	iHSV57
Art	Schrittmotor	Closed-Loop Schrittmotor	Servomotor
Kosten	39 €	110 €	133 €
Kraft in Nm	1,65	1,2	0,57
RPM	600	k. A.	3000
Gewicht in kg	1	0,7	1,6

Bei der Auswahl der verschiedenen Optionen entfällt der Servomotor, da er recht teuer ist und nicht die benötigte Kraft besitzt. Zwar ist er schnell, aber auch sein Gewicht ist vergleichsweise hoch.
Der einfache Schrittmotor scheint dem Closed-Loop Schrittmotor überlegen, da er wesentlich weniger kostet und ein höheres Haltemoment besitzt. Allerdings ist sein Gewicht leicht erhöht und er besitzt auch keine integrierte Steuereinheit. Diese müsste nachgerüstet werden. Wie aber zuvor schon erwähnt, kann es bei einfachen Schrittmotoren zu

Schrittverlust kommen. Damit die Ausführung jedoch immer zuverlässig durchgeführt wird, soll hier ein Closed-Loop Schrittmotor eingesetzt werden, der dies verhindert.

5.3.4 Sensoren

Nachdem der Tischkicker und die Aktoren ausgewählt sind, können die Sensoren bestimmt werden. Wie in Abschnitt 5.1 beschrieben soll die KI nicht über ein Kamerabild arbeiten, sondern die Informationen zu den Spielfiguren und den Ball in Form von Metadaten bekommen. Dadurch kann vor allem das Training und auch die Berechnung der KI vereinfacht werden. Bei der Sensordatenverarbeitung hingegen verlangt diese Herangehensweise einen zusätzlichen Schritt, da die Informationen zunächst aus dem Kamerabild ausgelesen werden müssen, bevor die Daten an die KI weitergegeben werden können. Andererseits lassen sich so auch andere Methoden als eine Kamera zur Bestimmung des Zustandes des Tischkickers verwenden. Die verschiedenen Optionen werden in den nächsten beiden Abschnitten genauer beleuchtet.

Erkennung des Spielballs

Die Erkennung des Balls ist prinzipiell am schwierigsten, da sich dieser frei im Raum bewegen kann und nicht fest mit dem Tisch verbunden ist. Mögliche Optionen werden im Folgenden aufgezählt und miteinander verglichen.

Die Erkennung der Position des Spielballs ist möglich über:

- Infrarot Touchscreen
- RFID/Bluetooth-Lokalisierung
- 2D/3D-Kameradaten

Eine Detektion mithilfe eines Infrarot Touchscreens bzw. 1D-Lasern lässt sich relativ einfach und genau durchführen. Hierfür müsste z. B. ein enges Raster aus Lasern am Tisch befestigt werden, wobei die Spielfiguren hier ein gewisses Hindernis darstellen und so das Auslesen erschweren. Das größte Problem bei dieser Art der Detektion ist die mangelnde Auswahl an passenden Produkten. Zwar kann auch ein Raster aus vielen einzelnen verfügbaren Sensoren aufgebaut werden, allerdings ist der Anschluss und die Verarbeitung der Daten so vieler Sensoren aufwändig.

Der Ball könnte auch über ein RFID/Bluetooth-System lokalisiert werden, wobei mindestens drei Empfänger am Tisch montiert und der Ball speziell als Sender präpariert sein müssten. Die Lokalisierung durch RFID bzw. Bluetooth ist eine mögliche Option, allerdings ist hier das Hauptproblem, dass es keine passenden Produkte für das gegebene Problem gibt und erst eine eigene Lösung designt werden müsste. Für das Projekt eines automatisierten Tischkickers ist dieser Schritt jedoch zu aufwändig.

Der vermutlich beste Lösungsansatz ist die Detektion über eine Kamera. So ist die Erkennung des Balls ganz ähnlich wie beim Menschen möglich. Um „blinde Bereiche“ durch Spielfiguren, welche die Sicht versperren, zu verhindern, sollten im besten Fall mehrere Kameras eingesetzt werden, die in verschiedenen Positionen angebracht sind, sodass unterschiedliche Blickwinkel möglich sind. Die Verarbeitung von Kameradaten ist jedoch nicht gerade einfach, da die Unterscheidung zwischen Spielfiguren und Ball durch ähnliche Farben und Formen schwerer wird. Mithilfe von einem neuronalen Netz, das auf die Erkennung der Bälle trainiert ist, können diese vermutlich trotzdem erkannt werden, allerdings muss ein solches Netz erst aufwändig trainiert werden, was besonders durch die Notwendigkeit von ausreichend Trainingsdaten sehr zeitaufwändig ist.
Eine weitere Lösung der Balldetektion ist die Erkennung mithilfe einer 3D-Kamera, die womöglich in Kombination mit Farbbildverarbeitung eingesetzt werden kann. Hierbei kann sich die Form des Balls zu Nutze gemacht werden, welche immer im gleichen Abstand vom Spielfeld zu finden sein muss. So sollte auch eine Unterscheidung zwischen Spielfigur und Ball stets möglich sein. Problematisch ist bei den 3D-Bilddaten, dass diese nicht ganz störungsfrei sind, da z. B. Streueffekte an verschiedenen Kanten auftreten, die eine Weiterverarbeitung der Daten deutlich erschweren.

Da in dieser Arbeit der Fokus auf dem Training der KI über die Simulation liegt, ist für den ersten Aufbau eine einfache Kameradatenauswertung am sinnvollsten. So kann Zeit bei der Entwicklung gespart und für andere Verarbeitungsschritte eingesetzt werden. Hierbei kann vernachlässigt werden, dass der Ball teilweise nicht erkennbar ist, da durch geeignete Tests die generelle Funktionalität auf der Hardware trotzdem untersucht werden kann.

Erkennung der Spielfiguren

Die Erkennung der Spielfiguren bietet einen etwas größeren Spielraum, da durch die feste Montage der Stangen direkt Sensoren angebracht werden können. Die Möglichkeiten werden ebenfalls aufgelistet und miteinander verglichen.

Die Erkennung der Verschiebung und Rotation der Spielfiguren ist möglich über:

- 2D/3D-Kameradaten
- 1D-Abstandsmesser
- Position der Schrittmotoren

Um die Position und Lage der verschiedenen Spielfiguren zu ermitteln, kann wie beim Ball mit Kameradaten vorgegangen werden. Dies gestaltet sich hier wiederum etwas schwieriger, da je nach Rotation die Form der Figuren unterschiedlich ist, was eine geometrische Zuordnung erschwert. Zudem sind besonders kleine Unterschiede in der Rotation durch das Bild einer Kamera schwer zu erkennen, wodurch die aufgestellten Anforderungen möglicherweise nicht erfüllt werden. Zudem befinden sich die Spielfiguren leicht oberhalb des Spielfelds, wodurch perspektivische Verzerrungen entstehen können, die eine Auswertung erschweren.
Mithilfe von bestimmten Markierungen auf den Stangen kann wie in [13] die Messung der Verschiebung und Rotation übernommen werden. Dieser Ansatz ist vielversprechend, aber erfordert eine gute Bildverarbeitung und entspricht nicht den gestellten Anforderungen an die Genauigkeit. Dies gilt vor allem bei der Rotation, da hier in der Regel von oben eine Hälfte des Markers zu sehen ist und so keine genaue Rotation bestimmt werden kann.

Anders als beim Ball kann hier jedoch mit klassischen Sensoren wie einem 1-dimensionalen Abstandsmesser gearbeitet werden, da die Stangen in ihrer festen Position stets hin und her gleiten und so die zwei verschiedenen Bewegungsrichtungen ermöglichen. Mit einem Abstandsmessgerät kann die Verschiebung leicht gemessen werden. Für die Rotation kann durch den Anbau einer Spirale wie in Abbildung 5.6 auch ein passender Wert zugewiesen werden. Hierfür muss dann nur die gemessene Entfernung der Verschiebung abgezogen werden, sodass eine Differenz entsteht, die abhängig von der Rotation ist. Diese Methode ist allerdings nur möglich, wenn die Technik hierfür auf der vom Spieler abgewandten Seite untergebracht wird, damit das Spielerlebnis nicht durch störende Messinstrumente verschlechtert wird. Um die Position der vom Menschen gesteuerten Mannschaft zu

ermitteln, darf bei dieser Art der Detektion der Tischkicker dementsprechend nicht mit sogenannten Teleskopstangen ausgestattet sein.

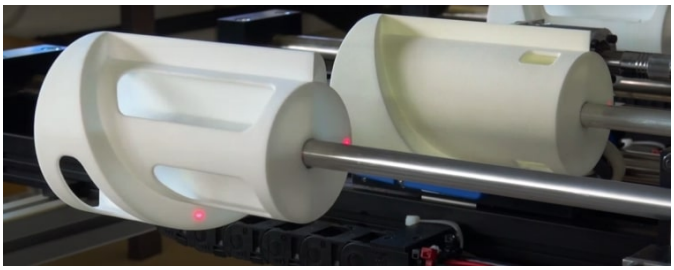


Abbildung 5.6: Messung der Verschiebung und Rotation mittels 1D-Laser über eine Spirale nach [87]

Für die vom Computer gesteuerten Stangen kann jedoch noch ein weiteres Verfahren verwendet werden, da der Antrieb über Schrittmotoren realisiert wird. In diesem Fall kann einfach die Anzahl der ausgeführten Schritte in eine Richtung gezählt werden und mit der Schrittweite in die entsprechende Position auf dem Spielfeld umgerechnet werden. Eine Herausforderung ist hierbei jedoch die Ermittlung des Startzustandes, da die Verschiebung und Rotation der Stangen zum Start des Programms nicht bekannt sind. Bei der Verschiebung lässt sich die Stange relativ leicht in eine feste Position bringen, indem ein Taster am Ende einer Stange angebracht wird, welcher genau dann auslöst, wenn die Stange bis zum Anschlag auf eine Seite geschoben wird. Zum Start des Programms werden dann die Stangen so lange in Richtung des Tasters bewegt, bis dieser auslöst. Dann befinden sich die Stangen in der gewünschten Position der Verschiebung. Gleichzeitig kann der Taster auch zwischendurch verwendet werden, um den Schrittmotor neu zu kalibrieren, falls ein oder mehrere Schritte verloren gehen.
Bei der Rotation kann hingegen zunächst kein Taster verwendet werden, da es hier keinen natürlichen Anschlag gibt. Wenn jedoch beachtet wird, dass sogenanntes Thrillern, also das Rotieren um mehr als 360°, verboten oder zumindest unerwünscht ist, kann ein künstlicher Anschlag hinzugefügt werden, wenn die Figuren über Kopf stehen. So lassen sich dann sowohl die Verschiebung als auch die Rotation über das Zählen der Schritte der Schrittmotoren auswerten.

Die Verschiebung und Rotation der Stangen soll im besten Fall durch die Position der Schrittmotoren erfolgen, da so eine extrem hohe Genauigkeit erreicht wird, die den Anforderungen mehr als genügt. Der Zustand der benutzergesteuerten Seite kann so natürlich nicht ermittelt werden. Hier scheint ein 1D-Laser die beste Wahl. Alternativ kann auch

eine grobe Auswertung über die Kamera erfolgen, da für die gegnerischen Reihen eine etwas ungenauere Auswertung ausreichen kann.

Kamera

Die Auswahl der Kamera ist gebunden an die Anforderung N1 für die Bestimmung der Position des Balls und an N8 für die zu erwartende Abtastrate.
Eine möglichst gute Auflösung sorgt für die Einhaltung des Genauigkeit. Dabei ist jedoch darauf zu achten, dass die Kamera eine höhere Brennweite und somit keinen Weitwinkel hat, damit das Bild möglichst wenig verzerrt und gleichzeitig die Auflösung an den Rändern nicht verringert ist. Um die Anforderung N1 einzuhalten muss etwa ein Pixel pro Millimeter vorhanden sein. Für den hier ausgewählten Tischkicker mit 400 mm x 300 mm, sind keine hohen Anforderungen an die Qualität gestellt. Da die Auswertung der Kameradaten in dieser Arbeit nicht im Vordergrund steht, wird eine günstige Lösung gesucht.
Aus einem vorigen Projekt steht eine Intel D415 Stereokamera [42] kostenfrei zur Verfügung. Sie bietet die Möglichkeit mit 60 FPS Farbbilder aufzunehmen und kann zudem auch mit 90 FPS 3D-Tiefenbilder liefern. Dies ermöglicht verschiedene Arten der Auswertung, wodurch sie gut für dieses Projekt geeignet ist. In [32] wird die Kamera zudem mit anderen Modellen verglichen und als optimale Lösung dargestellt. Da hier ähnliche Anforderungen in den Parametern Genauigkeit, Arbeitsbereich und Sichtfeld gestellt sind, kann die D415 Kamera für dieses Projekt verwendet werden.

Abstandssensor

Für die Erfassung der Verschiebung und Rotation der vom Menschen gesteuerten Reihen soll nach dem Vorbild in [87] mithilfe einer speziellen Spirale und zweier Abstandssensoren gearbeitet werden. Benötigt werden hierfür pro Reihe zwei Sensoren, welche einen stark gebündelten Laser zur Messung verwenden, damit die Entfernung zu einem genau ausgerichteten Punkt auf der Spirale erfolgen kann. Ein sehr wichtiges Kriterium ist also das sogenannte Field of View (FOV), welches bestimmt, wie weit sich der Kegel des Sichtfeldes öffnet. Sollte das FOV nicht klein genug sein, könnten auch Abstrahlungen von anderen Objekten wie z. B. der Stange gemessen werden, wodurch sich das Ergebnis der Messung verfälscht. Neben dem FOV ist auch die Reichweite des Sensors und die zu erreichende Genauigkeit der Messung von Interesse. Die Reichweite muss mindestens den

maximalen Verfahrweg der Reihen betragen, was selbst für einen großen Tischkicker bei unter $s_{max} = 1$ m liegt. Interessanter ist jedoch der Mindestabstand, welcher nur sehr gering sein darf, da die Stangen nicht viel Spielraum bieten und so schon kaum Platz zwischen Spirale und Tischkicker für den Sensor bleibt. Durch die Verlängerung der Stange durch die Spirale kann dem leicht entgegengewirkt werden, allerdings muss der Messbereich dennoch bei mindestens $s_{min} = 5$ cm anfangen, um ein möglichst platzsparendes Format zu erhalten. Bei der Genauigkeit geben die Anforderungen N2 und N3 für die Bestimmung der Verschiebung und Rotation die Vorgaben an den Sensor. Diese sind für die Verschiebung direkt übertragbar, sodass der Sensor diese Werte einhalten muss. Die Rotation ist hingegen abhängig von der verwendeten Spirale, welche so angepasst werden kann, dass die Anforderung für die Rotation mit der Genauigkeit aus der Verschiebung eingehalten wird. Zuletzt ist auch noch die Abtastrate zu beachten, welche ebenfalls aus den Anforderungen abgeleitet werden kann. Aus N8 geht hervor, dass mindestens eine Abtastrate von $f_{s,min} = 50$ Hz erreicht werden soll.

Ein solcher Sensor kann nicht in einer geeigneten Preisspanne gefunden werden. Die meisten Abstandssensoren haben ein zu großes Sichtfeld, wodurch keine zuverlässige Messung auf einen Punkt der Spirale möglich ist. Zwar gibt es günstige digitale Maßbänder [2], die den hier aufgestellten Anforderungen genügen, jedoch sind die benötigten Sensoren nicht einzeln verfügbar.

Aus diesem Gründen wird hier vom 1D-Laser abgesehen und die Auswertung über die Kamera empfohlen. In dieser Arbeit kann jedoch aus zeitlichen Gründen von der Auswertung über eine Kamera abgesehen werden, da die Position der computergesteuerten Stangen durch die Schrittmotoren bekannt ist. Die Verschiebung und Rotation der gegnerischen Stangen sind beim ersten Aufbau nicht weiter relevant.

5.3.5 Plattform

Die gestellten Anforderungen an den automatisierten Tischkicker sehen vor, dass dieser ein eigenständiges Gesamtsystem darstellt, welches transportiert und durch den Anschluss eines einzelnen Netzsteckers betrieben werden kann. Hierfür muss eine Plattform bereitgestellt werden, welche die Verarbeitung der Sensor- bzw. Kameradaten, die Ansteuerung der Motoren und die Berechnung der KI ausführen kann. Diese Plattform benötigt dementsprechend passende Anschlüsse für Kamera, verschiedene Sensoren und

Motorsteuerung. Sowohl für die Motorsteuerung, als auch für die meisten Sensoren werden GPIO Pins benötigt, während die ausgewählte Kamera über USB angeschlossen wird. Die ausgewählte Plattform muss dementsprechend über ein breites Angebot an GPIO-Pins verfügen und gleichzeitig mindestens einen USB3-Port besitzen.
Da sich der Ball beim Tischkicker sehr schnell bewegen kann und dementsprechend schnelle Reaktionszeiten benötigt werden ist auch die schnelle Verarbeitung der Daten essentiell. Besonders die manuelle Verarbeitung der Kameradaten ist ein entscheidender Faktor, da hierfür sehr viel Rechenleistung benötigt wird. Somit fallen viele Mikrocontroller als Option weg, da sie zwar die benötigten Anschlüsse besitzen, allerdings in der Regel nicht leistungsstark genug sind, um die in den Anforderungen abgeschätzten 50 FPS in der Bildverarbeitung zu erreichen.
Der Einsatz eines Computers würde das Problem der Rechenkapazität lösen, jedoch besitzt dieser keine Pins für den Anschluss von Peripheriegeräten. Gleichzeitig ist der Formfaktor eine entscheidende Frage, da ein großes und schweres Gehäuse den Transport des Tischkickers erschweren kann. Die optimale Lösung ist die Verwendung eines rechenstarken Mikrocontrollers, der speziell für die Bildverarbeitung und die Verwendung von neuronalen Netzen ausgelegt ist, wie der NVIDIA Jetson Nano [68].
Für einen ersten Aufbau bietet sich jedoch der Einsatz eines gewöhnlichen Computers an, welcher die Verarbeitung der Kameradaten und die Berechnung der nächsten Aktion durch die KI übernimmt. Um gleichzeitig auch die Steuerung der Motoren zu gewährleisten, muss der Computer per USB an einen einfachen Mikrocontroller angeschlossen werden. Zwischen den beiden Geräten müssen dann die relevanten Informationen ausgetauscht werden. Als Mikrocontroller wird hier der ESP32 ausgewählt, da er günstig ist und eine verhältnismäßig hohe Rechengeschwindigkeit hat.

6 Entwicklung

In diesem Kapitel wird die Entwicklung des KI-gesteuerten Tischkickers dargestellt. Zunächst werden der Aufbau und die Steuerung der Hardware erklärt. Zudem muss die Auswertung der Sensordaten erfolgen. Anschießend wird die Entwicklung der Simulation behandelt. Mit dieser lässt sich zuletzt das Training der KI durchführen.

6.1 Aufbau der Hardware

Der erste Schritt der Entwicklung besteht aus dem Aufbau der Hardware mit den ausgewählten Komponenten. Das Ziel ist dabei die Realisierung eines funktionalen und robusten Systems, welches auch für spätere Erweiterungen geeignet ist.

Im ersten Schritt wird der Minikicker wie in Abbildung 6.1 zu sehen auf einer größeren Holzplatte befestigt. Dabei endet die Platte auf der Seite des Spielers, sodass die Griffe frei erreichbar sind und ein Benutzer ungestört spielen kann. Auf der KI gesteuerten Seite wird mehr Platz eingeplant, damit Befestigungen für die Motoren und Sensoren leicht angebracht werden können. Vorteil dieses günstigen und sehr flexiblen Aufbaus ist die hohe Stabilität bei einem trotzdem relativ geringen Gewicht. So kann der Aufbau verhältnismäßig leicht transportiert und z. B. auf einem Tisch aufgestellt werden.
Da für den ersten Aufbau lediglich die Bewegung einer Stange über zwei Motoren geplant ist, werden lediglich die Stangen mit Torwart verwendet, wobei eine mit Motoren ausgestattet wird und die Andere von einem Menschen gesteuert werden kann. Die restlichen Reihen werden entfernt, sodass ein faires Spiel möglich ist. Mit den übrigen Reihen kann sowohl das Abwehren als auch das Angreifen getestet werden.

Damit die KI die Steuerung des Tischkickers übernehmen kann, werden die im Konzept ausgewählten Schrittmotoren benötigt. Damit sie fest montierbar sind, werden zwei verschiedene Halterungen entworfen und mittels 3D-Druck gefertigt (siehe Abbildung 6.2).

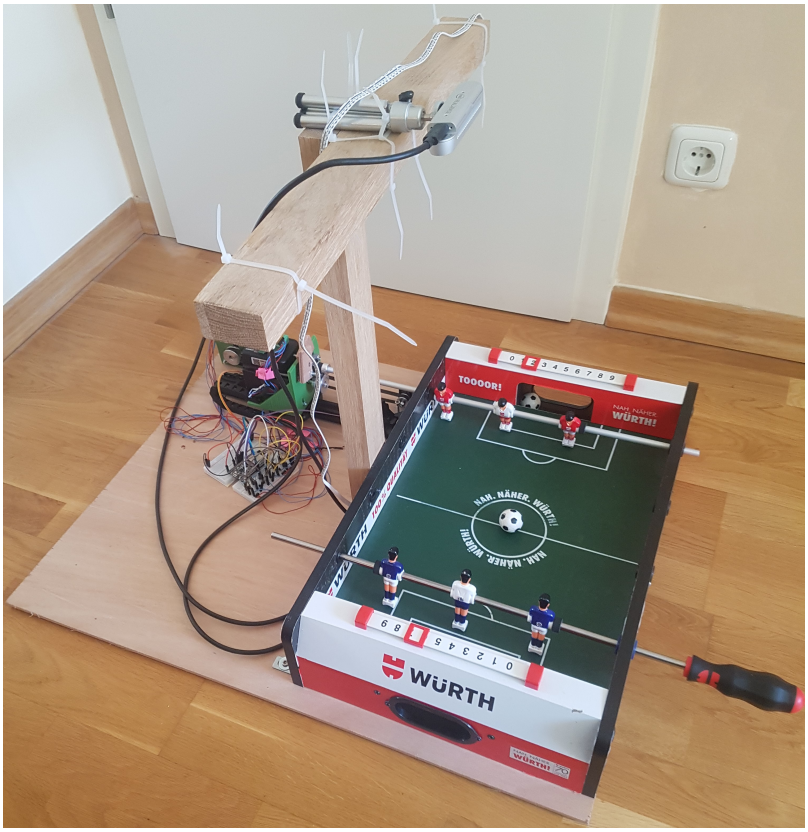
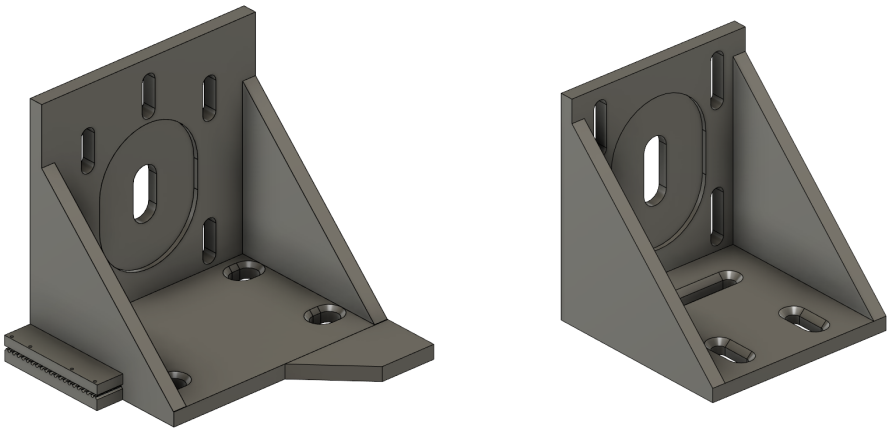


Abbildung 6.1: Aufbau der Hardware

Eine Halterung wird direkt auf der Holzplatte befestigt und ist für die Verschiebung verantwortlich. An der zweiten Halterung wird der Motor für die Rotation befestigt. Sie steht auf dem Liniarführwagen und hat zusätzlich Befestigungsmöglichkeiten für den Zahnriemen, der durch den ersten Motor bewegt wird. Gleichzeitig gibt es eine kleine Ecke an der die Energiekette, die alle Kabel für den Rotationsmotor führt, festgeschraubt wird.

Um die Bewegungen der Motoren auf die Stange übertragen zu können, wird eine Stange ohne Griff angefertigt. Diese ist etwas länger als die originalen Reihen, wodurch eine etwas höhere Flexibilität bei der Befestigung ermöglicht wird. Ansonsten ist sie den Originalen nachempfunden, sodass drei Spieler im gleichen Abstand auf ihr angebracht sind. Die Stange selbst wird nicht direkt am Rotationsmotor befestigt, sondern in ein Flanschlager, welches an der Motorhalterung angebracht ist, eingespannt. Dies hat den Vorteil, dass der Motor in einer geringeren Höhe angebracht werden kann und die Rotation des Motors über zwei Zahnräder einstellbar ist. Die Befestigung ist in Abbildung 6.3 zu sehen.



(a) Halterung Rotation (b) Halterung Verschiebung

Abbildung 6.2: Modelle der Motorhalterungen für den 3D-Druck

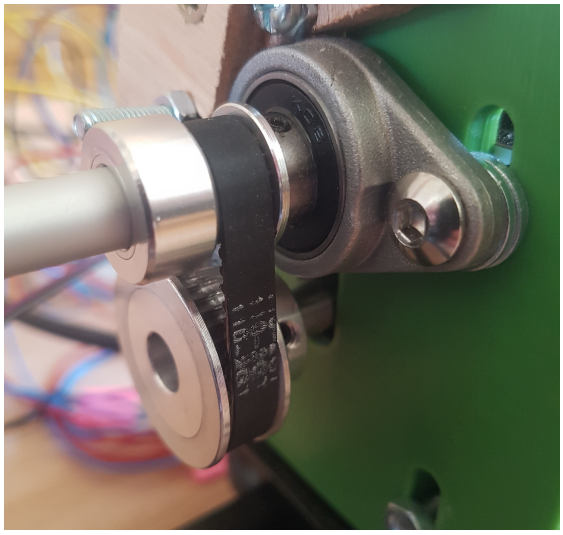
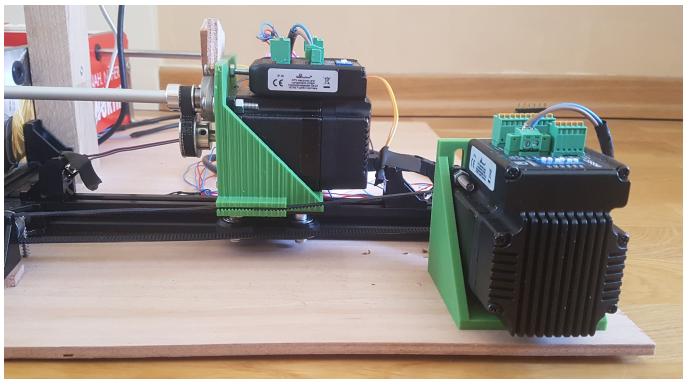


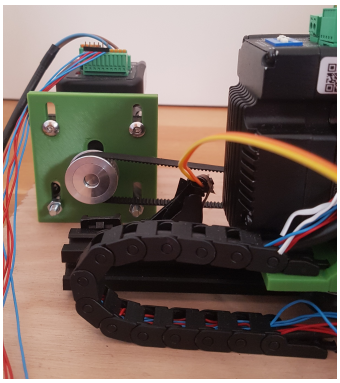
Abbildung 6.3: Befestigung der Stange mit den Motoren

Um den Rotationsmotor und so auch die Stange verschieben zu können, wurde im Konzept eine Linearführung mit einem passenden Wagen ausgewählt, auf dem der Motor befestigt werden soll. Ziel dabei ist eine möglichst reibungslose Bewegung des Motors. Die Auswahl der Linearführung ist im Konzept auf die AR/HR 25 Reihe gefallen, da sie hohe Geschwindigkeiten und Beschleunigungen zulässt und dies in einem ausführlichen Datenblatt dokumentiert ist [27]. Zu dem Verschiebewiderstand werden allerdings keine Angaben gemacht. Anders als angenommen ist dieser Widerstand sehr hoch, sodass mit

dem ausgewählten Motor keine Verschiebung mit bis zu $v_{Stange,max} = 2\ \frac{\mathrm{m}}{\mathrm{s}}$ erfolgen kann. Aus diesem Grund muss eine Alternative gefunden werden. Im Konzept wurden die Vorteile der Linearführungen mit Kugellager dargestellt, da aber kein passendes Produkt mit ausreichend Angaben im Datenblatt ermittelt werden kann, wird eine Rollenführung aus dem Bereich des 3D-Drucks verwendet. Diese ist für die Bewegung durch kleine Schrittmotoren entworfen und bietet die Möglichkeit zur Einstellung des Verschiebewiderstands. Der Rollwagen ist zudem für den Betrieb auf Aluminiumprofilen gedacht, welche z. B. in einem späteren endgültigen Aufbau für die Verbindung aller Komponenten genutzt werden können. Im Gegensatz zu der Kugelumlaufführung wird beim Rollenwagen kein Schmiermittel benötigt, was ebenfalls eine kleine Verbesserung darstellt. Die Kosten für die Linearführung mit passenden Profilführungen und Schrauben liegt bei 62 €. Diese Angabe ist aus [1] entnommen.
Die Linearführung wird mit den Motoren und der Energiekette auf der Holzplatte an dem Tischkicker befestigt. Der Aufbau ist in Abbildung 6.4 zu sehen. Er bietet eine gute Robustheit, während die Bewegung der Stange ohne größere Verluste gewährleistet ist.



(a) Gesamtansicht



(b) Energiekette und Befestigung des Zahnrads

Abbildung 6.4: Linearführung samt Motoren und Energiekette

Die Steuerung der Motoren wird von einem ESP32 Mikrocontroller übernommen. Dieser wird mit der integrierten Motorsteuereinheit der beiden Schrittmotoren über seine GPIO-Pins verbunden. Ein vereinfachter Schaltplan ist in Abbildung 6.5 zu sehen. Die Berechnung der KI wird in diesem ersten Prototypen von einem Computer übernommen. Dieser wird mit dem Mikrocontroller per USB verbunden, sodass die Befehle übermittelt werden können.

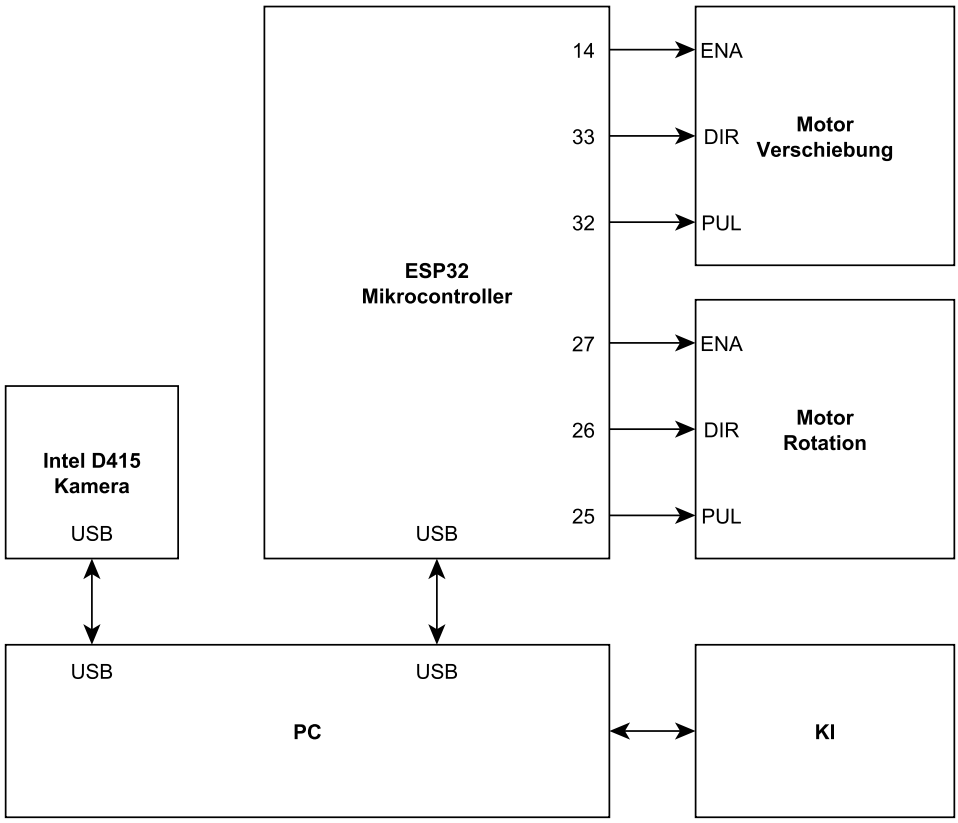


Abbildung 6.5: Grober Schaltplan Hardware

Ein Hindernis beim Anschluss der Motoren an den ESP32 ist die benötigte Spannung zum Schalten. Die Motorsteuerungen registrieren bei einer Eingangsspannung von 0 V $\leq$ $U_{in} \leq 0,5$ V ein LOW-Signal (0) und erst bei $3,5$ V $\leq U_{in} \leq 24$ V ein HIGH-Signal (1) [28]. Der Mikrocontroller hingegen verwendet lediglich eine Spannung von $U_{ESP} = 3,3$ V. Bei einem LOW-Signal sind $U_{out} = 0$ V am Ausgang, während bei einem HIGH-Signal lediglich $U_{out} = 3,3$ V geschaltet werden [8]. Dies reicht nicht aus, um die Hysteresespannung der Motorsteuerung zu überwinden. Aus diesem Grund bedarf es einer Transformation der Ausgangspegels des Mikrocontrollers. Der einfachste und günstigste Weg ist hierbei über eine Transistorschaltung mittels (N-Kanal) MOSFET [38].
Die Ausgänge des Mikrocontrollers werden an das jeweilige Gate des Transistors angeschlossen und können so den Widerstand zwischen Drain und Source bestimmen und als Schalter fungieren. Die Eingänge der Motorsteuerungen werden wie in Abbildung 6.6 mit einer vom ESP32 bereitgestellten Spannung von 5 V versorgt und am Drain angeschlos-

sen, während Source mit Ground verbunden wird. Zusätzlich wird ein hoher Widerstand von $R_1 = 100\ \text{k}\Omega$ zwischen Gate und Source eingebaut, welcher die parasitäre Kapazität des MOSFET ausgleicht [21].
Bei einer niedrigen Spannung am Gate ist der Transistor gesperrt, sodass kein Strom fließt und am Eingang der Motorsteuerung eine Spannung von $U_{in} = 0$ V anliegt. Wenn der Mikrocontroller $U_{out} = 3,3$ V schaltet ist der MOSFET zwischen Drain und Source niederohmig. Dadurch kann ein Strom fließen und die Motorsteuerung erhält eine Spannung von $U_{in} = 5$ V.

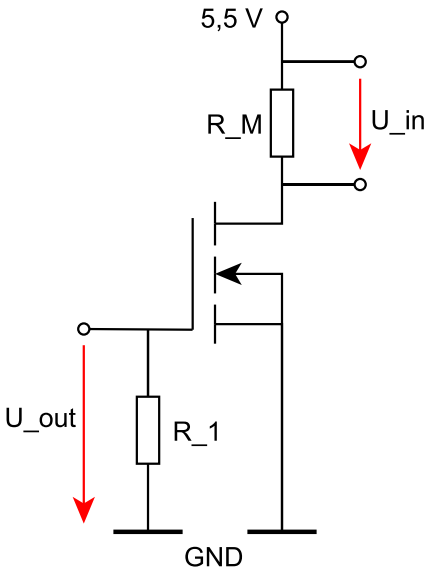


Abbildung 6.6: MOSFET 3,3 V auf 5,5 V Transistorschaltung

Damit die KI Informationen über den aktuellen Zustand des Tischkickers erhält, wird ein einfaches Stativ aus Holz auf der Seite des Computers aufgestellt, an welchem die verwendete Kamera festgemacht werden kann. Es bietet der Kamera eine Draufsicht von etwa 50 cm Höhe über dem Boden, was im Arbeitsbereich liegt. Das Holzstativ bietet zusätzlich Platz für weitere potentielle Kameras und LED-Streifen.
Die Befestigung der Intel D415 erfolgt mittels Kabelbinder und einem kameraeigenen Stativ. Dies ermöglicht die flexible Einstellung der Kamera auch nach der Montage. Die Kamera selbst wird ebenfalls über ein USB-Kabel an den Computer angeschlossen, welcher dann die Bildverarbeitung übernimmt.

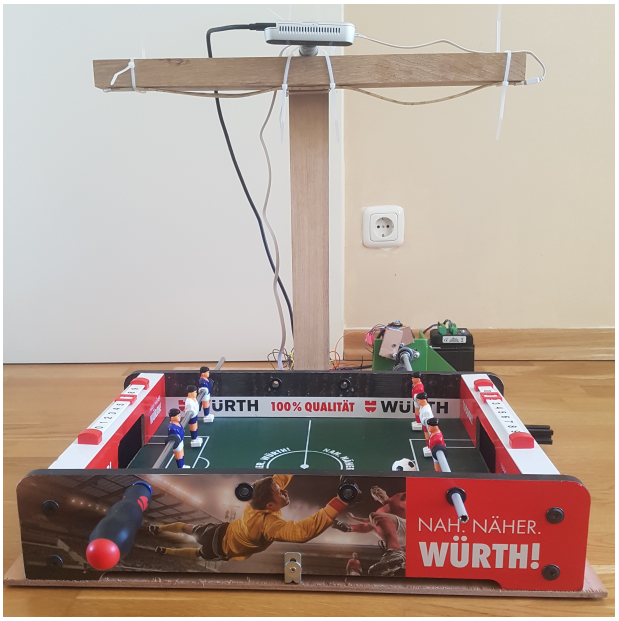


Abbildung 6.7: Befestigung der Kamera über dem Tischkicker am Stativ

Die Stromversorgung der Motoren wird in diesem ersten einfachen Aufbau durch ein geeignetes Labornetzteil geliefert. Dieses ist in Abbildung 6.8 zu sehen. Auf dem Netzteil kann eine Ausgangsspannung bis $U = 30$ V eingestellt werden, während Ströme bis $I = 10$ A möglich sind. Dies ist für die zwei Schrittmotoren ausreichend.

Abbildung 6.8: Labornetzteil für den Betrieb der Motoren

6.2 Motorsteuerung

Nach dem Aufbau kann die Motorsteuerung entwickelt werden, welche die Verbindung zwischen der KI und der Mechanik darstellt. Zunächst müssen die Steuereingänge der Motoren analysiert werden.
Die integrierte Steuereinheit der Schrittmotoren hat drei verschiedene Eingangssignale: ENA, DIR und PUL. Mittels ENA kann die Steuerung aktiviert oder deaktiviert werden. So ist es möglich den Motor in einen Ruhemodus zu versetzen, wenn er länger nicht benötigt wird. Das DIR-Signal ist zum Einstellen der Drehrichtung des Motors gedacht. Es sorgt zusammen mit dem PUL-Signal für die Umsetzung der Bewegung des Motors. Je nach Einstellung des Schalters SW5 wird mit einer steigenden oder einer fallenden Flanke an PUL der Motor um einen Schritt weiter gestellt. Die Schrittweite ist dabei über die Schalter SW1-4 festlegbar. Die Motoren werden hier jeweils auf 6400 Schritte pro Umdrehung eingestellt, da dies einen guten Kompromiss aus Geschwindigkeit und Genauigkeit darstellt und so ein ruhiger und leiser Betrieb möglich ist.

Die Ausgabe der Steuersignale wird wie in Kapitel 6.1 beschrieben durch den ESP32 übernommen. Er erhält dabei zunächst über seine serielle Schnittstelle die Befehle der KI. Diese bestehen aus einem Buchstaben zur Auswahl des jeweiligen Motors und aus einer Aktion im Intervall [-100, 100]. Der Wert der Aktion entspricht einer einheitslosen Geschwindigkeit v_{ziel}, mit welcher die Verzögerungszeit zwischen zwei Schritten eines Motors errechnet werden kann. Das Vorzeichen über die Drehrichtung entscheidet.
Das Programm zur Steuerung der KI auf dem Computer gibt nach jeder Berechnung der nächsten Aktion durch die KI die gewünschte Geschwindigkeit für die verschiedenen Motoren aus. Wenn sich der Befehl für einen Motor nicht ändern sollte, wird dies jedoch unterdrückt, um möglichst wenige Daten über den Bus an den Mikrocontroller zu versenden. Dies wird gemacht, um die benötigte Zeit für die Verarbeitung der Befehle auf dem ESP zu reduzieren. Dies ist wichtig, da bei einem großen Tischkicker mit acht Motoren und einer KI, die nach Anforderung N11 über 50 Berechnungen in einer Sekunde ausführt, über 400 Befehle verarbeitet werden müssen. Da das Auslesen der Befehle parallel zu der eigentlichen Steuerung der Motoren durchgeführt werden muss, sind Verzögerungen nicht vorteilhaft.
Das Auslesen der über den Bus gesendeten Daten wird daher auch über ein Interrupt der seriellen Schnittstelle ausgelöst. Dadurch wird kein regelmäßiges Polling benötigt, sodass die Steuerung der Signale für die Motoren möglichst nicht (lange) unterbrochen wird. Beim Aufruf des Interrupts wird das jeweilige Byte nur an einen String angehängt, der in

einen Integer-Wert umgewandelt wird, sobald ein Zeilenumbruch eingelesen wird. Beim Einlesen eines Buchstabens wird im Anschluss die entsprechende Aktion des jeweiligen Motors angepasst.

Für die Steuerung der Motoren muss zunächst die Aktion bzw. Geschwindigkeit v_m in eine Verzögerung umgerechnet werden. Die Verzögerung bestimmt den zeitlichen Abstand zwischen dem Pegelwechsel am PUL-Signal. Eine hohe Verzögerung entspricht also einer kleinen Geschwindigkeit und andersherum. Ein komplette Stillstand ist so jedoch nicht realisierbar, da dies einer unendlich hohe Verzögerung entspricht. Deshalb wird kein Schritt ausgeführt, wenn $v_m \leq 0.1$ ist. Anderenfalls kann eine Berechnung der Verzögerung über

$$T_m \;=\; \frac{100}{(w_{100} - w_0) \cdot |v_m|} \cdot 10^{-12} \tag{6.1}$$

erfolgen. Dabei entsprechen die Faktoren $w_{100} = \frac{1}{T_{100}}$ und $w_0 = \frac{1}{T_0}$ einer Art Winkelgeschwindigkeit, welche aus der gewünschten Verzögerung für die Geschwindigkeiten $v_m = 0$ und $v_m = 100$ berechnet werden. Die gewählten Verzögerungen liegen bei $T_{100} = 4\ \mu$s und $T_0 = 1000\ \mu$s. So lassen sich sehr hohe und niedrige Geschwindigkeiten realisieren. Ohne die Umrechnung der Verzögerungen auf die Winkelgeschwindigkeiten wäre die erreichte Bewegung durch die Motoren, also die tatsächliche Verschiebung oder Rotation der Stangen, nicht linear proportional zu der Geschwindigkeit v_m.
Eine Herausforderung der Motoren ist ihre Beschleunigung, denn sie können nicht aus dem Ruhezustand sofort zu ihrer maximalen Geschwindigkeit wechseln [85]. Ohne eine angemessene Beschleunigung kann es zu Schrittverlusten kommen. Um dies zu verhindern wird die Geschwindigkeit v_m nur sukzessiv erhöht. Dabei wird ihr neuer Wert mit einer zuvor für die verschiedenen Motoren abgestimmten maximalen Beschleunigung a_{max} berechnet. Mit

$$\Delta_{max} v_m \;=\; T_m \cdot a_{max} \tag{6.2}$$

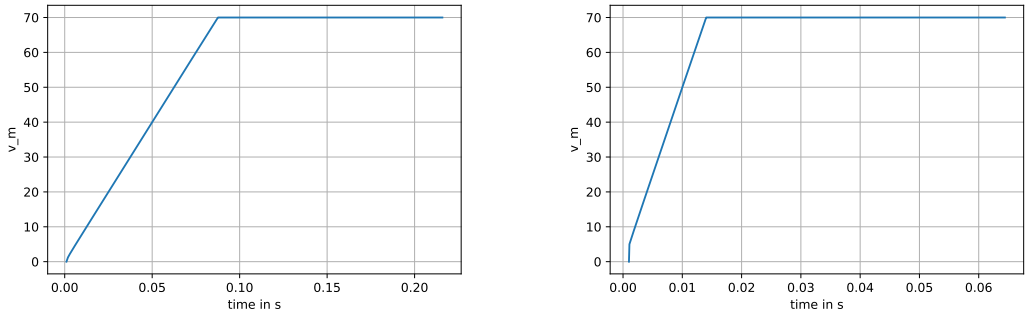
kann die maximale Differenz zwischen den Geschwindigkeiten berechnet werden, wobei natürlich auch die letzte Verzögerung T_m beachtet wird. Die Beschleunigung wird für Rotationsmotoren auf $a_{max} = 5000\frac{1}{\mathrm{s}^2}$ und für die Verschiebung auf $a_{max} = 800\frac{1}{\mathrm{s}^2}$ festgelegt. Die Motoren für die Verschiebung haben eine höhere Last zu ziehen, was die niedrigere Beschleunigung begründet.

Die neue Winkelgeschwindigkeit v'_m wird über

$$v'_m = v_m + \mathrm{sgn}\left(\mathrm{v_{ziel}} - \mathrm{v_m}\right) \cdot \Delta_{\mathrm{max}}\mathrm{v_m} \tag{6.3}$$

berechnet, wenn $|v_{ziel} - v_m| > \Delta_{max} v_m$ ist. Anderenfalls ist $v'_m = v_{ziel}$. Das Vorzeichen der Differenzgeschwindigkeit wird an das Vorzeichen der Berechnung $v_{ziel} - v_m$ angepasst. So kann die Berechnung auch für ein Abbremsen eingesetzt werden.

Die Beschleunigung der Motoren kann in den Abbildungen 6.9 und 6.10 gesehen werden. Der Rotationsmotor kann innerhalb von $t_r < 0.015$ s aus dem Ruhezustand in seine volle Geschwindigkeit beschleunigen, während dieser Prozess bei der Verschiebung mit $t_v < 0.1$ s etwas länger braucht. Die maximale Geschwindigkeit ohne Schrittverlust wird bei beiden Motoren als $v_{m,max} = 70$ ermittelt.

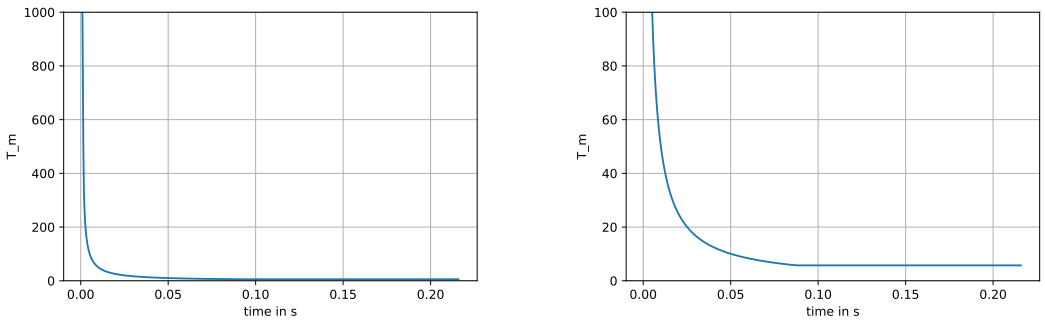


(a) Motor für die Verschiebung mit $a_{max} = 800\frac{1}{s^2}$

(b) Motor für die Rotation mit $a_{max} = 5000\frac{1}{s^2}$

Abbildung 6.9: Beschleunigung der Motorgeschwindigkeit v_m von 0 auf 70

Die Umsetzung der Steuerung erfordert neben der Berechnung einer Geschwindigkeit und Verzögerung auch die Ausgabe der gewünschten Signale. Hierfür muss lediglich das DIR-Signal je nach Vorzeichen der aktuellen Geschwindigkeit v_m angepasst und nach der berechneten Verzögerung T_m das entsprechende PUL-Signal des Motors invertiert werden. Das heißt, dass der Ausgang von HIGH auf LOW bzw. von LOW auf HIGH gesetzt wird. Die Herausforderung dabei ist jedoch, dass das Steuerprogramm nicht einfach die Verzögerung aktiv wartend absitzen kann, da mehrere Motoren verwendet werden und diese ganz unterschiedliche Verzögerungen besitzen. Statt in diesem Fall wie beim Einlesen der Daten des Busses wieder mit einem Interrupt zu arbeiten, wird hier eine andere Lösung implementiert. Der Nachteil bei der Verwendung von Timer-Interrupts ist, dass auf dem ESP32 nur vier Stück zur Verfügung stehen [46]. Bei einem späteren Aufbau



(a) Gesamte Grafik

(b) Ausschnitt der kleineren Verzögerungszeiten

Abbildung 6.10: Verzögerungszeit T_m bei der Beschleunigung der Motorgeschwindigkeit v_m des Motors für die Verschiebung von 0 auf 70

mit bis zu acht Motoren wäre eine Steuerung über einen einzelnen Mikrocontroller nicht mehr möglich. Da die Verzögerungen als ganzzahlige Integer gespeichert werden, die einen Wert zwischen $T_{100} = 4\ \mu s$ und $T_0 = 1000\ \mu s$ annehmen können, bietet sich eine Quantisierung der Verzögerung in $T_Q = 1\ \mu s$ an. So lässt sich nach einem Zeitschritt von T_Q eine Überprüfung durchführen, ob die verschiedenen Verzögerungen der Motoren abgelaufen sind. Wenn dies der Fall ist, wird dementsprechend das jeweilige Ausgangssignal invertiert und die nächste Geschwindigkeit v_m und Verzögerung T_m für den jeweiligen Motor berechnet. So können beliebig viele unterschiedliche Verzögerungen gleichzeitig berücksichtigt werden. Die benötigten Berechnungen zwischendurch haben eine gewisse Laufzeit, wodurch die Verzögerungszeiten leicht erhöht sind. Solange die Rechenzeit jedoch geringer als die kleinste Verzögerung T_{100} ist, kann dieser Effekt vernachlässigt und durch eventuell höhere Geschwindigkeiten ausgeglichen werden.

Sowohl die Motoren für die Rotation als auch für die Verschiebung, haben eine begrenzte Verfahrstrecke. Gerade bei der Verschiebung ist dies wichtig zu beachten, da die Begrenzung aus den Wänden besteht. Wenn die Motoren das Ende ihrer Verfahrstrecke erreichen, müssen sie zum Stehen kommen, um nicht in die Wand zu fahren. Dementsprechend muss beim Erreichen der Endposition die Bewegung in die entsprechende Richtung verhindert werden. Hierfür muss jedoch schon früher ein Abbremsen der Motoren erfolgen, da durch die Trägheit der bewegten Masse kein sofortiger Stopp möglich ist. Hierfür werden am Rand der Verfahrstrecken kleine Bereiche definiert, in denen die Ziel-

geschwindigkeit v_{ziel} manuell auf Null gesetzt wird, solange sie Richtung Wand verläuft. So kommt der Motor immer vor der Kollision mit der Wand zum Stehen und erreicht bei guter Abstimmung der Bereiche dennoch das Ende der Verfahrstrecke.
Eine zusätzliche Maßnahme, die auch bei möglichen Schrittverlust helfen kann, ist die Verwendung von zwei Tastern an den Enden der Verfahrstrecke. Wenn der Motor auf der Linearführung das Ende der Strecke erreicht, wird der entsprechende Taster eingedrückt. In diesem Fall müssen softwareseitig die Position und Geschwindigkeit angepasst werden, sodass keine weitere Verschiebung in Richtung Bande erfolgt.

Um bei längeren Tests keinen unnötigen Stromverbrauch durch die Schrittmotoren zu haben, welche bei Stillstand einen hohen Strom benötigen, werden diese automatisch über das ENA-Signal abgestellt, sofern für 5 s kein Pegelwechsel stattfindet. Eine Aktivierung der Motoren findet automatisch beim nächsten Pegelwechsel statt.

6.3 Kameradatenauswertung

Für die Umsetzung der Steuerung der Hardware durch die KI, wird eine Ermittlung des Zustands benötigt, da die KI mittels Metadaten arbeitet, statt die Informationen direkt aus einem Kamerabild zu ziehen. Aus zeitlichen Gründen kann in dieser Arbeit keine genaue und robuste Sensordatenverarbeitung zur Ermittlung der Metadaten entwickelt werden. Für den ersten Aufbau mit dem Minikicker wird dennoch eine Auswertung der Position und Bewegung des Balls über die Kameradaten benötigt. Anderenfalls ließe sich die Entwicklung der KI in der Simulation nicht auf der Hardware überprüfen. Für den Testaufbau sind die Anforderungen etwas verringert, da weniger Stangen und Figuren auf dem Feld sind und so weniger Hindernisse die Sicht der Kamera verdecken. Es reicht durch die große freie Fläche zunächst eine Auswertung der Position des Balls, wenn dieser unverdeckt erkennbar ist.
Neben dem Ball können über die Kamera auch Hände, Tore und die Verschiebung und Rotation der Spielfiguren erkannt werden.

6.3.1 Balldetektion

Zunächst wird die Balldetektion behandelt. Sie ist der wohl wichtigste Teil der Datenauswertung der Kamera. Zur Detektion gehört auch die Filterung bzw. Weiterverarbeitung

der Daten, sodass auch bei einer nicht erfolgreichen Detektion eine plausible Position an die KI weitergeleitet werden kann.

Anpassung des Bilds

Im ersten Schritt lohnt sich eine Anpassung des Bilds an den Tischkicker, denn die Kamera zeichnet bei einer nicht optimalen Ausrichtung einen größeren Ausschnitt als das Spielfeld auf. Zudem kann das Bild der Größe des Tischkickers angepasst werden, sodass ein Pixel einem Millimeter entspricht. So kann die Position des Ball entsprechend der Anforderung N1 mit einer hohen Genauigkeit im mm-Bereich über die Position im Bild bestimmt werden. Gleichzeitig kann so die Zeit für die Bildverarbeitung möglichst gering gehalten werden, da weniger Pixel verarbeitet werden müssen als bei der Verwendung eines größeren Bilds. Das Spielfeld muss hierfür jedoch zunächst ver- bzw. entzerrt werden, sodass die Ecken des Spielfelds mit den Ecken des zu verarbeitenden Bilds übereinstimmen.
Das Finden der Spielfeldecken könnte über verschiedene Algorithmen erfolgen, jedoch wird hier eine vom Benutzer manuelle Eingabe verwendet. Dies ist verhältnismäßig leicht umzusetzen und muss nur selten durchgeführt werden. Ein Benutzer kann einfach auf einer grafischen Ausgabe die Ecken in einer bestimmten Reihenfolge anklicken. Sobald alle vier Punkte gesetzt sind, werden zukünftige Bilder entzerrt. Der Vorteil gegenüber einer festen Einprogrammierung ist die gegebene Flexibilität, wodurch eine leichte Veränderung der Kameraposition keine Umprogrammierung erfordert.
Die Entzerrung des Bilds kann über die OpenCV-eigene Funktion *warpPerspective()* erfolgen. Dabei wird das neue Bild img_{dst} aus dem originalen Ausschnitt img_{src} über

$$img_{dst}(x,y) \;=\; img_{src}\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31} + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31} + M_{32}y + M_{33}}\right) \tag{6.4}$$

berechnet [72]. Die 3x3 Tansformationsmatrix $\boldsymbol{M}$ muss zuvor über die Funktion *getPerspectiveTransform()* mithilfe der vier ausgewählten Punkten und den Ecken des Bilds ermittelt werden. Sie ist dabei so definiert, dass

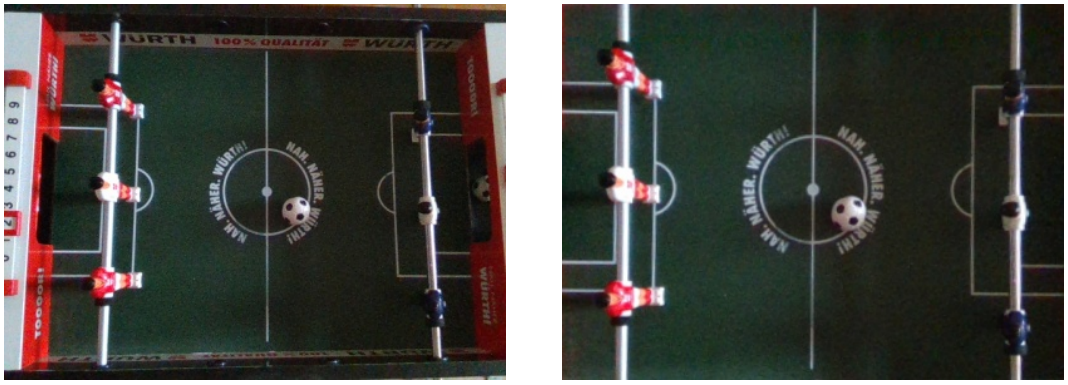
$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} \;=\; \boldsymbol{M} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{6.5}$$

ergibt, wobei t_i einen Skalierungsfaktor darstellt. Mit $i = 0, 1, 2, 3$ können die vier Eingangspunkte P_{src} und Ausgangspunkte P_{dst} ausgewählt werden. Ihre Koordinaten sind über $P_{dst}(i) = (x'_i, y'_i)$ und $P_{src}(i) = (x_i, y_i)$ definiert. Das sich hieraus ergebende Gleichungssystem muss einmal gelöst werden um die acht Werte der Transformationsmatrix

$$\boldsymbol{M} = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{bmatrix} \tag{6.6}$$

zu erhalten [48].

Das Ergebnis der Transformation des Bilds zur Anpassung an das Spielfeld ist in Abbildung 6.11 zu sehen.



(a) Originales Bild (b) Angepasster Ausschnitt

Abbildung 6.11: Anpassung des Kamerabilds an das Spielfeld

Neben der Entzerrung wird häufig noch eine weitere Verformung des Bilds durchgeführt, um die Effekte durch die Eigenschaften des Objektivs der Kamera auszugleichen. Besonders bei Kameras mit Weitwinkel, also einem breiten Sichtfeld, ist dieser Effekt stark ausgeprägt [57]. Abbildung 6.12 zeigt wie sich die Verformung auf ein Bild auswirken kann. Bei Kameras handelt es sich in der Regel um eine tonnenförmige Verzeichnung, welche mit dem Winkel des Sichtfelds zunimmt. Problematisch ist dies, da die Position im Bild nicht mehr zwingend der Position auf dem Kicker entspricht. Die ausgewählte Kamera hat jedoch den Vorteil, dass ihr Sichtfeld mit 65° × 40° recht moderat ist und so keine stärkere Verzerrung zu erkennen ist [42]. Aus diesem Grund wird in diesem ersten Ansatz auf weitere Maßnahmen verzichtet.

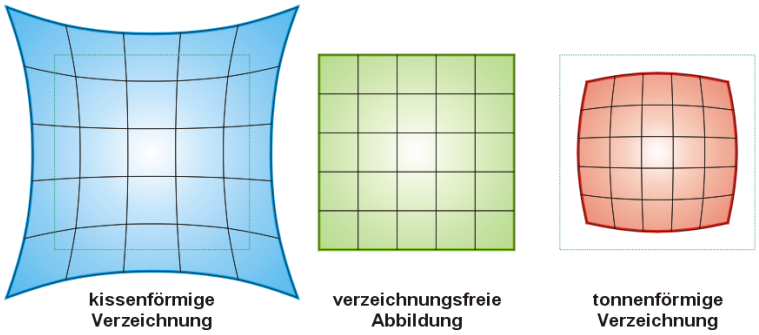


Abbildung 6.12: Abbildung eines Rechtecks bei der Verzerrung durch das Objektiv einer Kamera [104]

Schon vor der Entzerrung kann das Bild auf die Länge und Breite des Tischkickers über die OpenCV-Funktion *resize()* angepasst werden. Im Fall des Minikickers wird das Bild auf eine Größe von 400×300 Pixel reduziert, da dies der Größe des Felds in mm entspricht. Es gehen möglicherweise Informationen durch die Reduzierung der Bildgröße verloren, jedoch ist eine Genauigkeit entsprechend der Anforderung N1 gegeben. Gleichzeitig erhöht sich die Rechengeschwindigkeit in den folgenden Schritten, da weniger Daten verarbeitet werden müssen.

Farbbild

Ein erster Ansatz zur Erkennung des Balls ist klassisch über ein Farbbild. Hierbei kann sich zunutze gemacht werden, dass der Spielball größtenteils weiß bzw. hell ist, während das Spielfeld einen dunkelgrünen Boden besitzt. Die Daten der Kamera liegen in Form von RGB-Werten vor, die einzeln keine vollständige Aussage über die Farbe oder Helligkeit des Objekts zulassen. Da hier die weißen Bereiche erkannt werden sollen und an diesen Stellen alle RGB-Werte hoch sind, ist die Umwandlung des Farbbilds in ein Graustufenbild sinnvoll. Die Umwandlung des Farbbilds in Graustufen im Intervall von [0, 255] kann in Abbildung 6.13 gesehen werden.

Für die Erkennung von Objekten in dem entstandenen Bild muss zunächst noch eine Segmentierung stattfinden. Dabei soll der dunkle Hintergrund klar vom hellen Ball getrennt werden. Dies kann beispielsweise über eine Binarisierung des Bilds erfolgen. Dabei werden alle Pixel innerhalb des Bilds auf 255 oder 0 gesetzt. Die Entscheidung, welchen

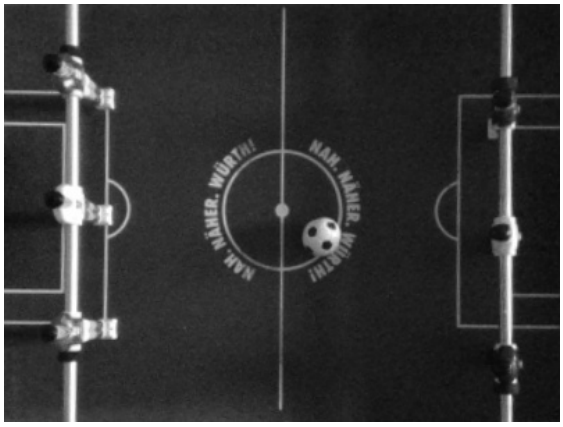


Abbildung 6.13: Umwandlung des entzerrten Bilds in Graustufen

Wert ein Pixel im Ausgangsbild annimmt, ist abhängig von der Intensität der Graustufe im Eingangsbild und wird über einen Schwellwert getroffen. Durch unterschiedliche Lichtverhältnisse und Kameraeinstellungen kann nur selten ein allgemeingültiger Schwellwert gefunden werden. Deshalb kann dies beispielsweise über das Otsu-Verfahren gelöst werden (siehe Abbildung 6.14). Häufig wird das Verfahren in Kombination mit einem Tiefpassfilter angewendet, um z. B. Effekte die durch Bildrauschen entstehen zu verhindern [17].

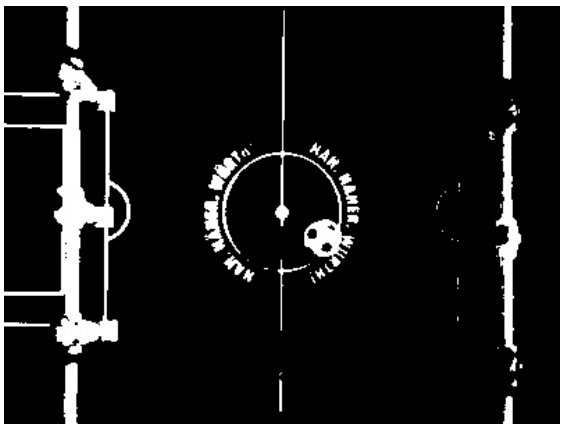
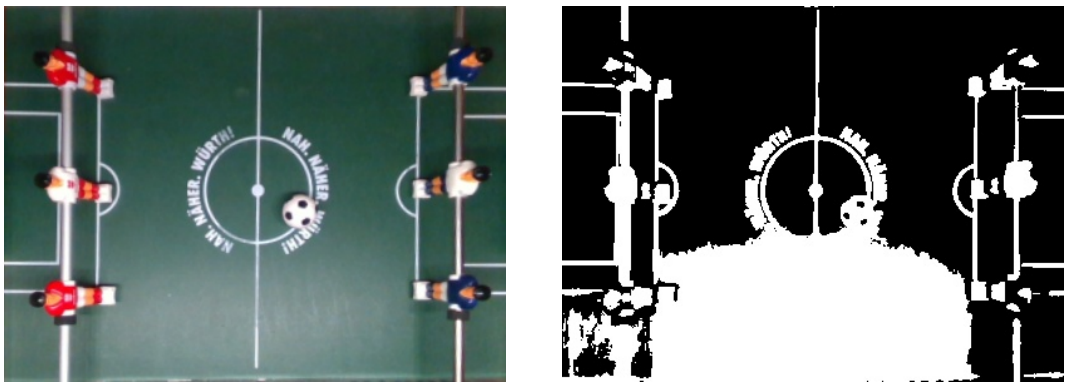


Abbildung 6.14: Binarisierung des Graustufenbilds über das Otsu-Verfahren

Nach der Binarisierung müssen die verschiedenen Konturen untersucht und mögliche Bälle ermittelt werden. Es fällt jedoch auf, dass die Entwicklung eines geeigneten Algorithmus nicht trivial ist, da neben dem Ball auch viele weitere Objekte, wie z. B. die Spielfiguren, als hell markiert werden. So verschwimmen auch die Linien auf dem Spiel-

feld mit dem Ball, sodass seine Kontur nicht direkt ausgelesen werden kann.
Dies sind grundsätzlich lösbare Aufgaben, die mithilfe weiterer Verarbeitungsschritte, wie dem Einsatz von Kantenfiltern oder durch morphologische Operationen, gelöst werden können. Weiter lassen sich wie in [44] statische Objekte, wie die Stangen und Linien, über Masken herausfiltern, sodass nach Möglichkeit nur der Ball übrig bleibt. Diese Maßnahmen sind bis sie ein gutes Ergebnis liefern jedoch recht aufwändig und im Rahmen dieser Arbeit nicht durchführbar. Hinzu kommt der Nachteil, dass die Bilder stark vom verfügbaren Licht abhängig sind. Für einen stets einsatzbereiten Tischkicker müsste Sonnenlicht möglichst vermieden werden und zusätzliche LEDs eingebaut werden, damit möglichst gleichbleibende Bedingungen erzeugt werden. LEDs haben jedoch auch den Nachteil, dass sie trotz der matten Oberfläche des Kickers stärkere Reflexionen erzeugen, welche die Verarbeitung erschweren (siehe Abbildung 6.15). Aus den genannten Gründen wird zunächst von einen Implementierung der Balldetektion über das Farb- bzw. Graustufenbild abgesehen.



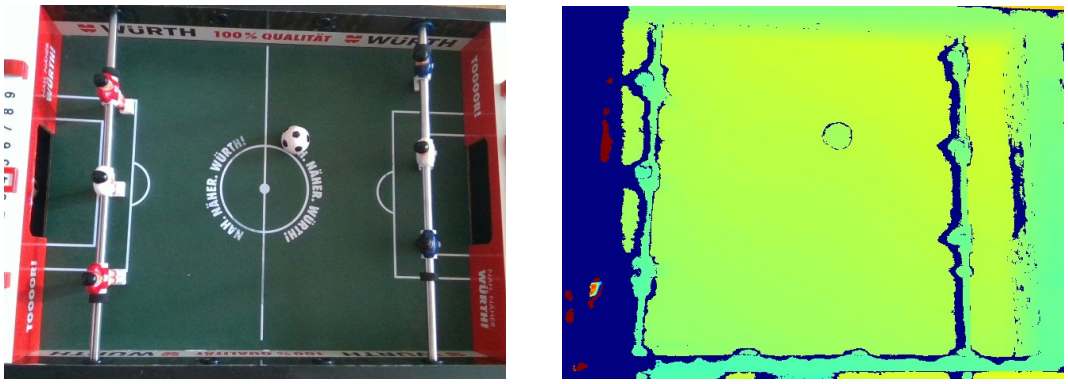
(a) RGB-Bild (b) Binarisiertes Bild

Abbildung 6.15: Bildverarbeitung bei künstlich beleuchtetem Spielfeld

3D-Bild

Die Intel D415 Kamera bietet neben dem klassischen RGB-Farbbild auch ein 3D-Tiefenbild. Dieses wird mittels zweier Bildsensoren aufgenommen, die bei schlechten Bedingungen von einem Infrarot Projektor unterstützt werden. Die Tiefendaten werden beim Zusammenführen der Bilder über ihre Merkmale berechnet [42]. Dies funktioniert aufgrund des Infrarot Projektors auch bei sehr schlechten Lichtverhältnissen bzw. Dunkelheit. Zudem lässt sich die Bildrate bei der Verwendung der 3D-Daten auf bis zu 90 FPS setzen, was

beim Farbbild mit maximal 60 FPS nicht möglich ist. Hierfür muss jedoch ein Anschluss der Kamera über USB 3.1 erfolgen.



(a) RGB-Bild (b) 3D-Tiefendaten

Abbildung 6.16: Vergleich verschiedenen Bilddaten Intel D415

In Abbildung 6.16 kann ein solches Tiefenbild gesehen werden. Je weiter weg ein Objekt ist, desto höher ist der Graustufenwert des Bilds (in Abbildung 6.16 werden die Graustufen zur besseren Übersicht in verschiedenen Farben angezeigt). Es fällt einerseits auf, dass der Ausschnitt dieses Bilds nicht ganz mit dem des Farbbilds übereinstimmt, andererseits sind links neben den einzelnen Objekten Schatten zu sehen. Der leicht veränderte Bildausschnitt ist für die weitere Verarbeitung nicht weiter relevant. Die Schatten hingegen blockieren vor allem links von der linken Stange die Sicht. Dadurch ist an dieser Stelle keine Erkennung mehr möglich. Gerade für die erste Implementierung ist dieser Umstand jedoch hinnehmbar, da idealerweise der Ball von der KI nicht in diesen Bereich gelassen wird. Die Schatten entstehen dadurch, dass das Tiefenbild auf den linken Stereo Sensor projiziert wird und die Bereiche links von den Objekten nicht vom rechten Sensor aufgenommen werden können [42]. Durch einen höheren Abstand zum Kicker kann dieser Effekt deutlich verringert werden.

Die eigentliche Detektion des Balls muss über die Tiefendaten, also den Abstand der Objekte im Bild zur Kamera, erfolgen. Hierbei kann sich zur Nutze gemacht werden, dass der Ball auf dem flachen Spielfeld liegt. Da die Kamera über dem Tisch angebracht ist, befindet sich der Ball immer näher zur Kamera als der Boden an der jeweiligen Stelle. Um den Boden von dem Ball unterscheiden zu können, muss nur eine Subtraktion der Höhe des Bodens vom Tiefenbild erfolgen, sodass dieser im Bild 0 und somit komplett

schwarz wird. Dies kann mithilfe einer Konstanten c_{off} erfolgen, sodass sich

$$\boldsymbol{D}(x,y)' \;=\; \boldsymbol{D}(x,y) - c_{off} \tag{6.7}$$

ergibt. Da der Ball sich näher am Sensor befindet, hat er einen kleineren Wert. Bei der Subtraktion der Entfernung des Bodens erfolgt ein arithmetischen Unterlauf wie in Abbildung 6.17 und die Entfernung des Balls wird mit einem hohen (weißen) Wert versehen. So sind Ball und Boden leicht unterscheidbar.

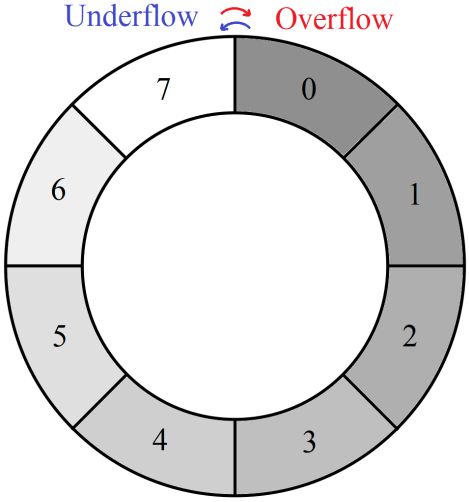


Abbildung 6.17: Beispiel eines arithmetischen Über- und Unterlaufs

Das vorgeschlagene Verfahren alleine reicht jedoch noch nicht aus, da wie in Abbildung 6.18 der Boden des Spielfelds nicht an allen Stellen die gleiche Entfernung zur Kamera besitzt. So kann keine Konstante gefunden werden, die den Boden an allen Stellen auf (fast) 0 setzt, während der Ball an jeder Stelle auf dem Feld im Unterlauf ist.

Eine Lösung für dieses Problem kann durch die Verwendung eines flexiblen Parameters erfolgen. Statt immer einen konstanten Wert c_{off} zu verwenden, wird in diesem Fall die Distanz zum Boden für jeden Punkt im Bild errechnet. Dies kann z. B. über ein Referenzbild erfolgen, wobei hier der Nachteil ist, dass auch die beweglichen Figuren als Referenz verwendet werden und so kein einheitliches Bild entsteht. Wenn die Figuren später bewegt werden, können z. B. durch die Köpfe runde Konturen entstehen, die eine Auswertung erschweren.

Abbildung 6.18: Tiefenbild der Kamera nach Abzug der Konstanten a

Statt ein Referenzbild zu verwenden, kann der Abstand der Ebene an jedem Punkt im Bild errechnet werden. Hierfür müssen vom Benutzer mindestens 3 Punkte (P_{E1}, P_{E2} und P_{E3}) auf dem Boden des Kickers ausgewählt werden, welche zusammen eine Ebene aufspannen. So hat der Ball immer den gleichen Grauwert und die Figuren können durch den einheitlichen Unterlauf besser gefiltert werden.
Die Ebene kann über die Berechnung des Normalenvektors über die zwei Vektoren $\overrightarrow{P_{E1}P_{E2}}$ und $\overrightarrow{P_{E1}P_{E3}}$ bestimmt werden [6]. Da der Normalenvektor orthogonal zu beiden Vektoren steht, lässt er sich über das Kreuzprodukt

$$\boldsymbol{n} = \overrightarrow{P_{E1}P_{E2}} \times \overrightarrow{P_{E1}P_{E3}} = \boldsymbol{a} \times \boldsymbol{b} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \times \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 b_2 - a_2 b_1 \\ a_2 b_0 - a_0 b_2 \\ a_0 b_1 - a_1 b_0 \end{pmatrix} \quad (6.8)$$

berechnen [25]. Ein grober Überblick zum Normalenvektor kann aus Abbildung 6.19 gewonnen werden.

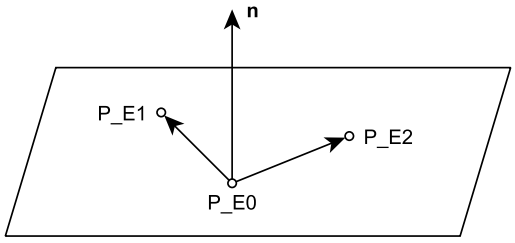


Abbildung 6.19: Normalenvektor einer Ebene

Der Vektor $\overrightarrow{P_{E1}P_E}$ eines beliebigen Punkts $P_E(x, y, z)$ auf der Ebene muss immer orthogonal zu $\boldsymbol{n}$ sein, sodass

$$\boldsymbol{n} \cdot \overrightarrow{P_{E1}P_E} \;=\; n_0 \cdot (x_{E1} - x) + n_1 \cdot (y_{E1} - y) + n_2 \cdot (z_{E1} - z) \;=\; 0 \tag{6.9}$$

ergibt. Diese Formel kann nach z, also der Tiefe des Bilds, umgestellt werden.

Für die Implementierung wird jedoch ein anderer Ansatz gewählt, bei dem eine Ebene möglichst genau den Referenzpunkten angenähert wird. So können mehr als drei Punkte ausgewählt werden, sodass mögliche Ungenauigkeiten durch Rauschen oder Unebenheiten des Bodens ausgleichbar sind. Hierfür wird die Funktion *scipy.linalg.lstsq*() eingesetzt, welche die Gleichung $\boldsymbol{Ac} = \boldsymbol{b}$ über den minimalen quadratischen Fehler annähert [89]. Als Eingang dienen hier $\boldsymbol{A}$ und $\boldsymbol{b}$, womit $\boldsymbol{c}$ bestimmt wird. $\boldsymbol{A}$ setzt sich aus den xy-Positionen der Punkte und einer Konstanten 1 zusammen, sodass sich

$$\boldsymbol{A} \;=\; \begin{pmatrix} x_{E0} & y_{E0} & 1 \\ x_{E1} & y_{E1} & 1 \\ x_{E2} & y_{E2} & 1 \\ \vdots & \vdots & \vdots \end{pmatrix} \tag{6.10}$$

ergibt. Die Tiefenwerte z_E der jeweiligen Punkte sind in $\boldsymbol{b}$ gespeichert. Mit der errechneten Lösung von $\boldsymbol{c}$ kann über

$$\boldsymbol{D}_E(x, y) \;=\; c_0 \cdot x + c_1 \cdot y + c_2 \tag{6.11}$$

der entsprechende angepasste Tiefenwert der Ebene an dieser Stelle errechnet werden. Abbildung 6.20 zeigt das Tiefenbild der Kamera nach Abzug der berechneten Referenzebene. Da kleine lokale Unterschiede auftreten, wird ein zusätzlicher Offset von $c_{off} \approx 10$ mm addiert. So treten keine Störungen mehr auf und der Ball bleibt stets mit vollem Umfang im Unterlauf, da sein Radius $r_{Ball} = 15$ mm beträgt.

Nach der Vorverarbeitung muss das Tiefenbild wieder binarisiert werden, um die Objekterkennung durchzuführen. Hierbei kann erneut die Binarisierung über Otsu durchgeführt werden. Das Ergebnis der Binarisierung ist in Abbildung 6.21 zu sehen. Anders als bei der Verwendung des Farbbilds, ist hier eine klare Trennung des Balls vom Hintergrund möglich.

Abbildung 6.20: Tiefenbild der Kamera nach Abzug der Referenzebene

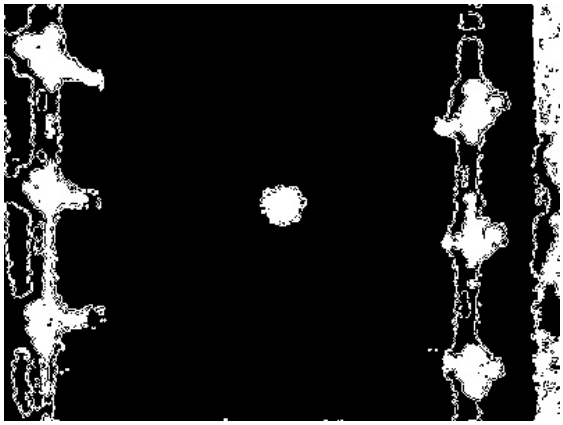


Abbildung 6.21: Binarisiertes korrigiertes Tiefenbild

Auffällig im binarisierten Bild sind die Randeffekte an den Objekten, die durch die zuvor erwähnten Schatten entstehen. Um diese zu eliminieren, kann eine Erosion durchgeführt werden, wobei die weißen Pixel invertiert werden, wenn nicht alle benachbarten Pixel in Vierer-Nachbarschaft ebenfalls weiß sind. Um möglichst viele Störeffekte herauszufiltern, werden 2 Iterationen durchgeführt. Dies ermöglicht auch eine bessere Trennung zwischen den einzelnen Objekten. Das Ergebnis hiervon ist in Abbildung 6.22 zu sehen.

Bei den verbleibenden Konturen muss der Ball von den Spielfiguren unterschieden werden. Dies kann einerseits über die Größe erfolgen, da der Ball kleiner als die Spielfiguren ist. Andererseits ist auch die Form ein Indikator. Deshalb wird zunächst der kleinstmögliche Kreis gesucht, der alle Pixel der Kontur einschließt. Wenn sein Radius im Bereich von 10 bis 14 Pixeln bzw. mm liegt, kann die Kontur von der Größe dem Ball entspre-

Abbildung 6.22: Binarisiertes Tiefenbild nach der Erosion

chen. Die Form der Kontur kann anhand des Verhältnis der eigenen Fläche (Anzahl der Pixel) zu der Fläche des kleinsten Kreises um sich selbst berechnet werden. Der Wert aus $f = \frac{A_{cnt}}{A_{circ}}$ wird dann maximal 1, wenn die Kontur einem perfekten Kreis entspricht. Je kleiner der Wert, desto weniger ähnelt die Form einem Kreis. Als Grenzwert wird hier $f = 0.5$ ermittelt.
Der Mittelpunkt des umschließenden Kreises wird als Ballmittelpunkt festgelegt. Für den Benutzer wird der Kreis in die grafische Ausgabe eingezeichnet, sodass überprüft werden kann, ob eine erfolgreiche Detektion stattfindet.
Mit diesen einfachen Mitteln lässt sich bei ungestörtem Blickfeld auf den Ball eine erfolgreiche Detektion durchführen. Andere Objekte lassen sich wie in Abbildung 6.23 gut unterscheiden.

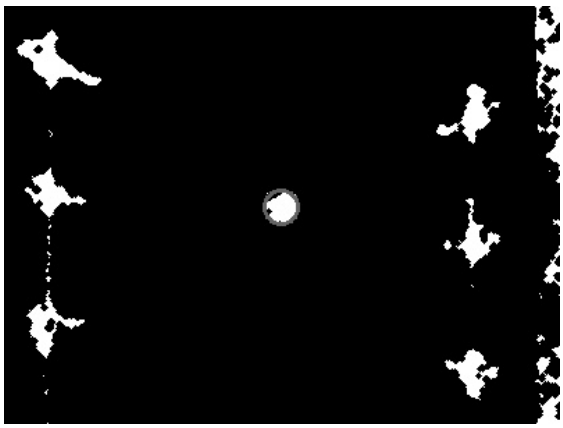


Abbildung 6.23: Vollständige 3D-Balldetektion

Ballverfolgung

Neben der Position sind auch die Bewegungsrichtung und die Geschwindigkeit interessant, da diese Daten an die KI weitergegeben werden. Die Geschwindigkeit kann einfach aus der Distanz

$$s = \sqrt{\Delta x_B^2 + \Delta y_B^2} \tag{6.12}$$

zwischen der aktuellen und letzten Position über

$$v_B = \frac{s}{T_s} \tag{6.13}$$

berechnet werden, wenn die Zeit T_s zwischen den beiden Abtastungen bekannt ist. Der Winkel der Bewegungsrichtung kann dann durch

$$\varphi = \arccos\left(\frac{\Delta x_B}{s}\right) \tag{6.14}$$

berechnet werden. Für eine robustere, jedoch etwas verzögerte Berechnung kann eine Art digitales Schieberegister eingeführt werden, welches die Berechnung von Δx_B und Δy_B über mehrere Bilder zulässt.

Eine Herausforderung bei der Balldetektion stellen die Spielfiguren da, welche den Ball (teilweise) verdecken können. Mit dem derzeitigen Ansatz ist in diesen Fällen eine Detektion nicht mehr möglich, die KI benötigt jedoch trotzdem eine Position des Balls. Zudem muss auch die Bewegungsrichtung und Geschwindigkeit berechnet werden, was ohne eine passende Position schwer möglich ist.

Kalman-Filter

Ein Ansatz mit erfolglosen Detektion umzugehen, ist über die Verwendung eines Kalman-Filters. Dieser ermöglicht Vorhersagen für die nächste Position des Balls. Der Kalman-Filter hat drei wichtige Vorteile, die im Folgenden erläutert werden.

Einerseits werden die eingehenden Daten gefiltert, sodass Rauschen geglättet wird. Dadurch kann wie in Abbildung 6.24 eine leichte Verbesserung der Messung erreicht werden.

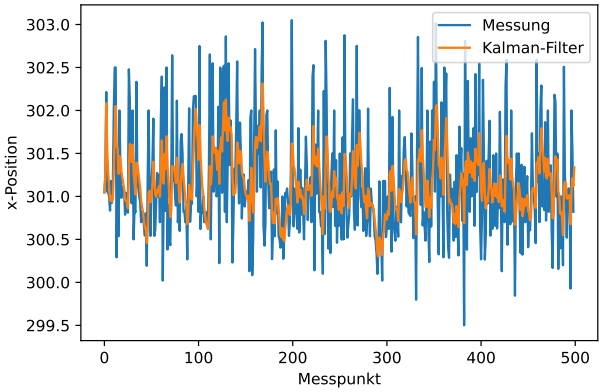


Abbildung 6.24: Glätten von Messrauschen durch den Kalman-Filter

Andererseits kann bei nicht erfolgreichen Balldetektionen über die gespeicherte Geschwindigkeit eine gute Schätzung der nächsten Position erfolgen. Dies ermöglicht der KI die Reaktion auf Bälle, die unter den Stangen durchrollen.
Zuletzt ist zu beachten, dass durch die Schätzungen auch zusätzliche Abtastungen erfolgen können, ohne ein Bild zu verarbeiten. So kann zwischen zwei Auswertungen der Bilder eine Schätzung über den Kalman-Filter vorgenommen und die Abtastrate sozusagen verdoppelt werden.
Die Einstellung des Kalman-Filters für den hier vorgestellten Fall werden im Folgenden erklärt. Als Messeingang dient lediglich die xy-Position des Balls. Zur besseren Übersicht wird nur die Filterung einer Komponente dargestellt. So ist der Systemzustand über

$$\boldsymbol{x} = \begin{bmatrix} x \\ v_x \end{bmatrix} \tag{6.15}$$

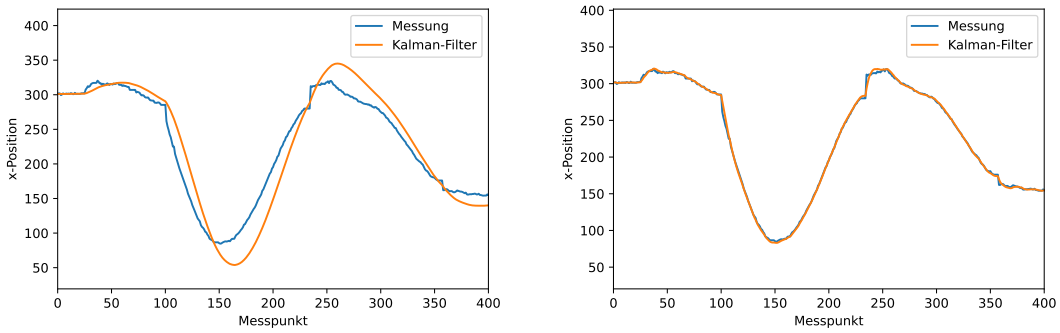
definiert. Hierbei entspricht x der gemessenen Position und v_x der Geschwindigkeit, also der Ableitung. Neben der Position wird hier auch die Geschwindigkeit abgeschätzt, wodurch eine manuelle Berechnung zuvor nicht nötig ist. Da nur die Position als Messeingang verwendet wird, ist die Beobachtungsmatrix

$$\boldsymbol{H}^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{6.16}$$

Passend zum Systemzustand wird die Beobachtungsmatrix als

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{6.17}$$

angelegt, wobei durch den Zeitschritt Δt zwischen den Messungen bei der Vorhersage auch die eigene Geschwindigkeit beachtet wird.
Ein sehr wichtiger Faktor ist die Messrauschkovarianzmatrix, welche die Messgenauigkeit angibt. In diesem Fall besteht die Matrix aus einem einzigen sehr niedrigen Wert, denn wird der Ball erkannt, ist die ermittelte Position sehr genau. Eine größere Abweichung vom Messwert ist nicht erwünscht, kann sich aber bei der Wahl eines hohen Werts der Messungenauigkeit ergeben (siehe Abbildung 6.25).



(a) Hohe Messgenauigkeit

(b) Geringe Messgenauigkeit

Abbildung 6.25: Effekt der Messrauschkovarianzmatrix auf die Positionsbestimmung

Bei der Initialisierung des Kalman-Filters ist die Position noch nicht bekannt und muss geschätzt werden. Dementsprechend wird die Kovarianzmatrix $\boldsymbol{P}$ mit hohen Werten belegt. Nach einer Messung oder Vorhersage werden diese Werte natürlich angepasst.

Verwendung letzter gültigen Messung

Ein zweiter Ansatz ist das Beibehalten der letzten gültigen Position. Dies erscheint zunächst ein wesentlich schlechterer Ansatz als der Kalman-Filter, da die Geschwindigkeit des Balls nicht weiter berücksichtigt wird. Es ist jedoch zu beachten, dass der Ball unter den Stangen häufig gar nicht durchrollt, sondern von der Spielfiguren abgelenkt oder gestoppt wird. In diesem Fall entfernt sich bei der Berücksichtigung der Geschwindigkeit des Balls dieser von der eigentlichen Position auf dem Tisch.

Der optimale Ansatz ist vermutlich eine Mischung aus dem Kalman-Filter und dem zweiten Ansatz. Dabei können die Daten stets über den Kalman-Filter verarbeitet werden und fehlende Positionsmessungen ausgeglichen werden. Wenn jedoch ein Ball länger nicht erkannt wird, sollte auf den zweiten Ansatz gesetzt werden, da der Spielball vermutlich von einer Figur angehalten und verdeckt wird.
Im besten Fall erreicht die Balldetektion einen Stand, bei dem der Ball auf jedem Punkt des Felds erkannt wird, wodurch weitere Verarbeitungsschritte weniger wichtig sind. Eine optimale Implementierung der vorgeschlagenen Lösung ist nicht nötig, da beim Aufbau lediglich die Torwart-Stangen verwendet werden und der Ball dementsprechend selten verdeckt ist. Im diesem Aufbau wird zunächst nur die Verwendung der letzter gültigen Messung implementiert.

Tor Erkennung

Wichtig für eine Spielstandsanzeige oder das Training der KI ist die Erkennung von Toren. Aufgrund der eingeschränkten Funktionalität der Balldetektion kann insbesondere in den Randbereichen nur schwer ermittelt werden, ob ein Tor gefallen ist. Da die Spielstandsanzeige keiner hohen Priorität unterliegt und das Training der KI zunächst nur in der Simulation stattfinden braucht, ist eine Erkennung von Toren nicht zwingend nötig für den Erfolg dieser Arbeit. Aus zeitlichen Gründen wird diese Funktion nicht umgesetzt.

6.3.2 Handerkennung

Die Handdetektion ist erforderlich, um die Motoren rechtzeitig abzustellen, wenn der Benutzer mit seiner Hand in den Spielbereich greift. Diese Sicherheitsmaßnahme soll Verletzungen vermeiden und dem Spieler die Möglichkeit geben, einen liegen gebliebenen Ball anzustoßen.
Hierbei kann ähnlich wie bei der Balldetektion vorgegangen werden. Statt jedoch eine Kontur zu finden, die dem Ball entspricht, wird hier versucht eine Hand zu ermitteln. Dies kann vor allem über die Größe der gefundenen Kontur bestimmt werden. Während der Ball und die Spielfiguren nur relativ kleine Konturen haben, ist eine menschliche Hand zumeist eine große zusammenhängende Struktur im Bild. Sie kann also allein über die Fläche der Kontur ermittelt werden. Wenn ein gewisser Grenzwert überschritten

wird, handelt es sich um einen Fremdkörper und der Betrieb muss gestoppt werden. Als geeigneter Grenzwert wird hier $A_{cnt} > 5000$ Pixel bzw. mm^2 festgelegt.
Auch hier wird wie in Abbildung 6.26 gezeigt für den Benutzer ein Rechteck um den Fremdkörper gezeichnet.

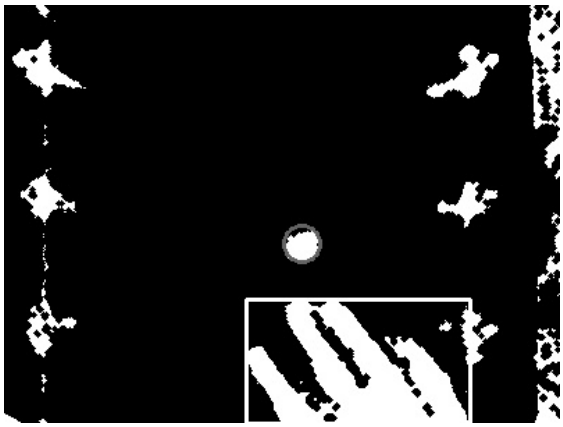


Abbildung 6.26: Erkennung einer Hand im Spielbereich

6.4 Simulation

Vor dem Training der KI muss die Simulation des Tischkickers programmiert werden. Die Entwicklung dieser wird im Folgenden erläutert. Dabei wird zunächst auf den grundsätzlichen Aufbau und die Datenstruktur sowie die Visualisierung dieser eingegangen. Anschließend werden die benötigten Funktionen für die Kollisionsdetektion und das Bewegen der Spielfiguren erarbeitet. Zuletzt wird die Ausführung der Simulation beschrieben, wobei genauer auf die Kollisionsdetektion und die Berechnung der Bewegung des Balls eingegangen wird.

6.4.1 Datenstruktur

Der Grundpfeiler der Simulation ist die Datenstruktur, denn eine gut durchdachte Struktur kann dabei helfen die Komplexität der Programmierung möglichst gering zu halten und gleichzeitig eine effiziente Berechnung zu erlauben. Die Struktur ist hier aufgrund der Anforderung SN1 in die einzelnen Objekte unterteilt. Dies hat zum Vorteil, dass die Eigenschaften leicht und modular anpassbar sind. Gleichzeitig können Funktionen wie

die Kollisionsdetektion auf die Parameter der jeweiligen Klasse, zu der sie gehören, zugreifen, sodass weniger Parameter übergeben werden müssen.
Die verschiedenen Objekte und ihre Beziehungen zueinander sind im vereinfachten Klassendiagramm in Abbildung 6.27 abgebildet. Im Folgenden werden die Klassen aufgelistet und mit ihren Parametern und Funktionen beschrieben. Die meisten Parameter sind bei der Initialisierung einstellbar. Die genauen Funktionen werden dabei jedoch vereinfacht dargestellt, um die Auflistung übersichtlich zu halten.

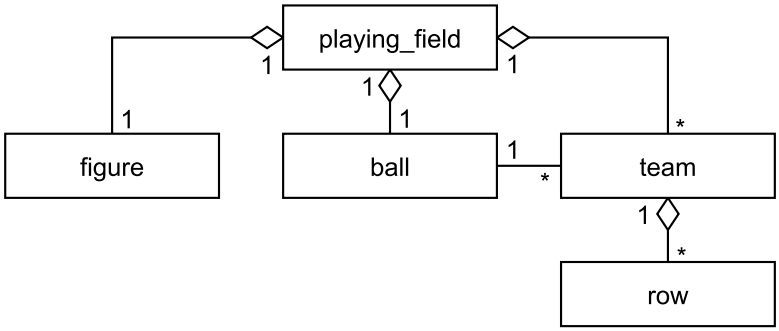


Abbildung 6.27: Struktur der verschiedenen Klassen

playing_field

Der gesamte Tischkicker wird in einem Objekt beschrieben, sodass die Art des Kickers und sein Zustand als ein Objekt übergeben werden können. Der Tischkicker umfasst deshalb auch die weiteren Unterobjekte, wie z. B. den Ball oder die Spielfiguren. Tabelle 6.1 zeigt einen Überblick der Klasse des Tischkickers (**playing_field**) und der dazugehörigen Funktionen. Die Klasse der Teams besteht aus einer Liste mit einer flexiblen Größe. So können je nach Bedarf ein oder zwei Mannschaften dem Spiel hinzugefügt werden.
Die Funktion *get_state()* dient dem Auslesen des Zustands des Tischkickers, um diese Informationen an die KI weiterzugeben. Hierbei werden die eigenen Parameter, sowie die Parameter der Unterobjekte abgefragt und in einer Liste ausgegeben. Die Informationen der anderen Unterobjekte werden dabei über ähnliche Funktionen abgefragt.
Mit *plot_game()* können die Informationen zum Zustand des Tischkickers ebenfalls ausgelesen werden, allerdings erfolgt die Ausgabe nicht in Form einer Liste, sondern über eine visuelle Darstellung. Dies dient der Auswertung der Daten durch einen Benutzer.

Tabelle 6.1: Beschreibung der Klasse des Spielfelds

playing_field	Name	Beschreibung
Eigenschaften	width	Länge des Spielfelds
	height	Breite des Spielfelds
	goal	Breite der Tore
	figure	Klasse der Spielfiguren
	teams	Klasse einer oder mehrerer Mannschaften
	ball	Klasse des Spielballs
Methoden	get_state()	Gibt den aktuellen Zustand des Spielfelds zurück
	plot_game()	Gibt den aktuellen Zustand des Tischkickers grafisch aus

Die einzigen Parameter, die direkt in dieser Klasse beschrieben werden, sind die Dimensionen des Tischkickers und die Breite der Tore. Mit diesen kann ein Koordinatensystem aufgespannt werden, in welches die anderen Objekte ebenfalls eingefügt werden. In diesem Koordinatensystem wird die Länge des Spielfelds auf der Abszisse und die Breite entlang der Ordinate abgebildet. Die Höhe der Objekte über dem Boden wird im Koordinatensystem auf der Applikate gespeichert. Dabei wird eine Ecke des Spielfelds als Ursprung festgelegt. In der gesamten Simulation werden [mm] als Standardeinheit verwendet. Abbildung 6.28 zeigt ein solches Spielfeld, wobei mit der Breite des Tors 2 gegenüberliegende Tore in der Mitte der jeweiligen Seite eingefügt werden (hier durch je zwei schwarze Punkte gekennzeichnet).

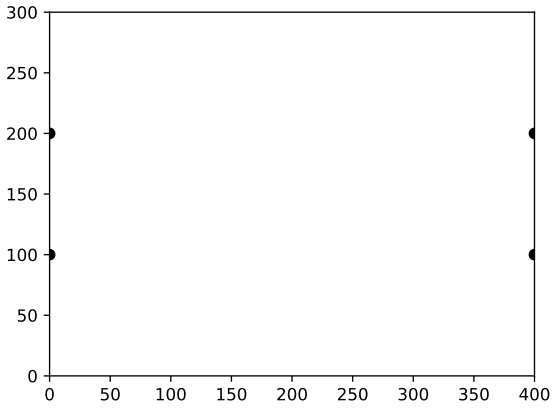


Abbildung 6.28: Aufgespanntes Spielfeld mit einer Länge von 400 mm, einer Breite von 300 mm und einer Breite der Tore von 100 mm

figure

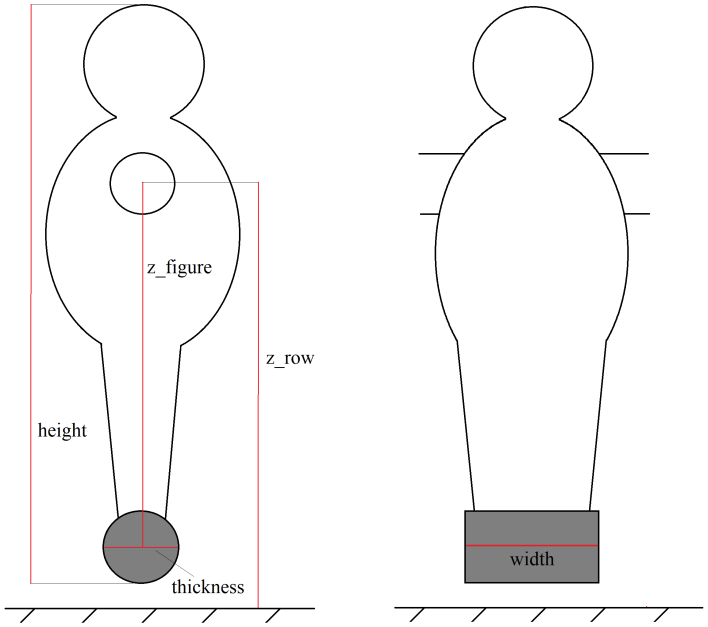
Das erste Objekt, welches dem Tischkicker hinzugefügt wird, beschreibt die Spielfiguren. Diese sind gegebenenfalls in hoher Anzahl auf dem Tisch vertreten, dabei ist ihre Form jedoch immer gleich. Ziel dieser Klasse ist die möglichst genaue Beschreibung der Form der Spielfiguren, sodass dies nicht bei jeder weiteren Reihe und Figur erneut erfolgen muss. Die Form der Spielfiguren müssen demnach nur einmal an die Eigenschaften des realen Tischkickers angepasst werden. Die hierfür benötigten Parameter werden in der Tabelle 6.2 aufgelistet.
Auch für die Figuren gibt es die *get_ state()* Funktion, welche die eigenen Parameter auflistet und für die KI bereitstellt. Diese Funktion wird wie bei den anderen Unterobjekten auch durch die Funktion der **playing_field** Klasse aufgerufen.

Tabelle 6.2: Beschreibung der Klasse der Spielfiguren

figure	Name	Beschreibung
Eigenschaften	height	Höhe der Figuren
	width	Breite w der Figuren
	thickness	Tiefe th der Figuren
	z_row	Höhe der Aufhängung/Stangen
	z_figure	Abstand des unteren Endes der Figuren zu der Aufhängung
Methoden	get_state()	Gibt den aktuellen Zustand des Spielfelds zurück

Die angegeben Parameter der **figure** Klasse sind nicht ganz selbsterklärend, weshalb sie in Abbildung 6.29 definiert werden. In der Regel ist die Form der Spielfiguren eines Tischkickers recht komplex. Eine detailgetreue Darstellung der Figuren ist aus verschiedenen Gründen nicht sinnvoll. Zum einen ist das Speichern dieser Informationen recht aufwändig, denn hierfür müsste zunächst ein geeignetes 3D-Modell der Figuren erstellt werden. Dieser Schritt würde z. B. sehr komplexe Software benötigen, die aus Kamerabildern ein genaues 3D-Modell erstellen kann. Zum anderen wird die spätere Kollisionsdetektion wesentlich komplexer und zeitaufwändiger, je genauer das Modell der Spielfigur ist. Zuletzt darf nicht vergessen werden, dass der Großteil der Spielfiguren völlig irrelevant für den Spielverlauf ist, da sie (fast) nie mit dem Spielball in Berührung kommen, da dieser in der Regel auf dem Boden rollt.
Es ist dementsprechend sinnvoll nur den unteren Teil der Figuren anzunähern und für die Kollisionsdetektion zu nutzen. Dieser Bereich ist in Abbildung 6.29 grau gekennzeichnet. Hierbei ist die Entscheidung auf einen Zylinder gefallen, da bei vielen Tischkickern die

Figuren an den Füßen abgerundet sind und so dem Zylinder nahe kommen. Gleichzeitig ist ein Zylinder ein einfacher Körper für die 3D-Kollisionsdetektion.



(a) Seitenansicht der Spielfiguren

(b) Frontalansicht der Spielfiguren

Abbildung 6.29: Definition der Parameter der Spielfiguren

team

Einer der Parameter des Tischkickers beschreibt die verschiedenen Mannschaften. Dabei wird eine Mannschaft immer von einem Spieler gesteuert. Das Team selbst hat keine entscheidenden Funktionen, da es in die einzelnen Reihen unterteilt wird. Die Anzahl der Reihen ist flexibel, da so ein einzelner Torwart oder wie bei einem normalen Tischkicker gleich mehrere Reihen simuliert werden können. Die Tabelle 6.3 zeigt die Eigenschaften und Funktionen der Klasse.

Tabelle 6.3: Beschreibung der Klasse der Mannschaften

team	Name	Beschreibung
Eigenschaften	rows	Klasse einer oder mehrerer Reihen im Team
Methoden	get_state()	Gibt den aktuellen Zustand der Mannschaft zurück

row

Eine logische Ebene unter den Mannschaften sind einzelnen Reihen, die der jeweiligen Mannschaft angehören und die von dem Spieler bzw. der KI gesteuert werden können. Die Tabelle 6.4 zeigt die Parameter und Funktionen der Klasse.

Tabelle 6.4: Beschreibung der Klasse der einzelnen Stangen samt Spielfiguren

row	Name	Beschreibung
Eigenschaften	x	Position der Reihe entlang der Abszisse
	step_width_shift	Schrittweite des Motors für die Verschiebung
	y_offset_max	Maximale Verschiebung der Reihe auf der Ordinate
	y_offset	Verschiebung der Reihe entlang der Ordinate
	v_shift	Geschwindigkeit der Verschiebung
	max_a_shift	Maximale Beschleunigung des Motors für die Verschiebung
	step_width_rot	Schrittweite des Motors für die Rotation gemessen am unteren Ende der Spielfiguren
	alpha	Winkel α_r der Rotation der Reihe
	v_rot	Geschwindigkeit der Rotation gemessen am unteren Ende der Spielfiguren
	max_a_rot	Maximale Beschleunigung des Motors für die Rotation
	y_figures	Position einer oder mehrerer Spielfiguren auf der Reihe entlang der Ordinate
Methoden	get_state()	Gibt den aktuellen Zustand der Reihe zurück
	row_update()	Berechnet die neue Position und Lage der Reihe und die aktuelle Geschwindigkeit dieser

Die Reihen werden entlang des Tischkickers platziert und beinhalten eine oder mehrere Figuren, jedoch in diesem Fall nicht das Objekt, sondern bloß die Position der Figuren auf der Stange entlang der Ordinate. Für die Kollisionsdetektion muss diese Information mit den in der **figure**-Klasse gespeicherten Daten fusioniert werden. Dabei müssen auch die Verschiebung der Stange auf der Ordinate, sowie die Rotation berücksichtigt werden.

Anders als die letzten Klassen haben die Reihen mit *row_ update()* eine aufrufbare Funktion. Diese dient der Steuerung der Stangen über zwei Eingangsbefehle. Hierbei werden sowohl die Verschiebung als auch die Rotation gleichzeitig durchgeführt. Zudem wird die aktuelle Geschwindigkeit der Verschiebung und Rotation berechnet. Bei der Rotation ergibt sich die Geschwindigkeit aus der Winkelgeschwindigkeit multipliziert mit dem

Radius $v_{rot} = \omega \cdot z_{figure}$.

Die restlichen Parameter werden für die Berechnung der Geschwindigkeit und Position benötigt. So gibt es eine maximale Beschleunigungsgeschwindigkeit für beide Motoren, welche sich aus dem realen Aufbau ableiten. Weiter wird auch die maximale Verschiebung mit [-$y_{offset,max}$, $y_{offset,max}$] bestimmt. Für die Rotation ist dies nicht nötig. Zwar ist das Durchdrehen der Stangen untersagt, jedoch kann die Begrenzung immer durch [-π, π] bestimmt werden.

ball

Das letzte und in gewisser Weise wichtigste Objekte ist der Spielball, denn neben seinen eigenen Eigenschaften hat der Ball auch viele für das Spiel wichtige Methoden.

Eine dieser Methoden ist *update_position()*, welche mithilfe der eigenen Geschwindigkeit und Bewegungsrichtung die nächste Position des Balls berechnet und diese im kartesischen Koordinatensystem des Spielfelds speichert. Dabei wird zudem der aktuelle Zeitschritt *dt* beibehalten, welcher für spätere Korrekturen benötigt wird. Zu der Funktion gehört auch der Aufruf der Kollisionsdetektion, welche sich in zwei Schritte unterteilt. Einerseits die Kollisionsdetektion mit den Spielfiguren der Mannschaften und andererseits mit den Wänden des Tischkickers. Hierzu gehört auch die Tordetektion, welche im Grunde eine spezielle Kollisionsdetektion mit den Wänden ist.
Sollte eine tatsächliche Kollision auftreten, wird die *change_movement()* Funktion aufgerufen, welche die Bewegungsrichtung des Balls ändert. Bei dieser Berechnung werden Parameter wie der Restitutionskoeffizient benötigt. Er entscheiden zusammen mit dem Gewicht des Balls über den Verlust des Momentums bei einer Kollision. Dabei ist k für die Wände und Spielfiguren unterschiedlich. Weiter kann mit dem Parameter *gravity* verhindert werden, dass sich die Bewegungsrichtung auf der Applikate ändert, da diese Dimension für den Tischkicker im Grunde vernachlässigbar ist.
Zuletzt gibt es eine Funktion zum Anpassen des Zeitschritts an die Geschwindigkeit des Balls, damit die Anforderung SN4 eingehalten werden kann und eine Kollisionsdetektion spätestens nach der angegebenen Strecke Δs erfolgt.

Die beschriebene Klasse samt ihrer Eigenschaften und Methoden wird in Tabelle 6.5 zusammengefasst und beschrieben.

Tabelle 6.5: Beschreibung der Klasse des Balls

ball	Name	Beschreibung
Eigenschaften	mass	Masse des Balls
	k_wall	Restitutionskoeffizient für die Kollision mit Wänden
	k_figure	Restitutionskoeffizient für die Kollision mit Spielfiguren
	radius	Radius r des Balls
	x	Position auf der Abszisse
	y	Position auf der Ordinate
	z	Position auf der Applikate
	phi	Winkel der Bewegungsrichtung des Balls in der Äquatorebene
	theta	Winkel der Bewegungsrichtung des Balls in der Polachse
	velocity	Geschwindigkeit des Balls
	dt	Zeitschritt für die letzte Positionsänderung
	gravity	Parameter für die Verwendung von Gravitation
Methoden	get_state()	Gibt den aktuellen Zustand des Teams zurück
	update_position()	Berechnung der nächsten Ballposition anhand der gegebenen Parameter unter Berücksichtigung möglicher Kollisionen
	collision_team()	Überprüfung einer Kollision mit den Reihen und Figuren der Mannschaft
	collision_wall()	Überprüfung einer Kollision mit den Wänden
	goal_check()	Überprüfung eines möglichen Tors
	change_movement()	Berechnung der neuen Bewegungsrichtung und Geschwindigkeit nach einer Kollision
	calc_dt()	Berechnung des maximalen Zeitschritts anhand der Geschwindigkeit des Balls

6.4.2 Visualisierung

Damit der Benutzer die Funktionalität der KI und der Simulation selber überprüfen kann, muss nach Anforderung SF14 eine einfache grafische Ausgabe aufrufbar sein. Sie muss dabei keine hohen Anforderungen erfüllen, aber einen groben Überblick erlauben. Es sollen dementsprechend der Ball, die Reihen mit ihren Spielfiguren und das Spielfeld selber abgebildet werden. Zwar erfolgt die Simulation nach Anforderung SN5 im 3-dimensionalen Raum, allerdings ist für die Visualisierung eine 2-dimensionale Darstel-

lung, also eine Draufsicht von oben, ausreichend. Es muss jedoch gewährleistet sein, dass auch die Rotation der Spielfiguren einsehbar ist.
Die Funktion zur Ausgabe des aktuellen Zustands ist, wie im vorigen Abschnitt beschrieben, Teil der Tischkicker-Klasse und kann jederzeit abgerufen werden, um den aktuellen Zustand auszugeben. Da die grafische Darstellung verhältnismäßig langsam ist, aber sehr viele Berechnungen für die Kollisionsdetektion benötigt werden und gleichzeitig die Änderung der Position nach Anforderung SN4 nur sehr klein sein soll, ist ein Aufruf dieser Funktion nicht nach jeder Aktualisierung der Position des Balls nötig. Die Steuerung der Ausgabe wird allerdings nicht durch die Funktion selber gesteuert, sondern durch die Funktion zur Ausführung der Simulation. Diese beinhaltet auch einen Parameter, mit welchem die grafische Ausgabe gänzlich abgestellt werden kann, sodass beim Training Rechenzeit gespart werden kann.

Die visuelle Darstellung selber wird über die Python Bibliothek Matplotlib realisiert. Diese ist eigentlich für die Darstellung von verschiedensten Graphen gedacht und nicht für Echtzeitanwendungen optimiert. Andere Möglichkeiten wie PyQtGraph sind gegebenenfalls schneller [84] und können so eine flüssige Ausgabe in Echtzeit ermöglichen, allerdings können mit Matplotlib ähnlich komplexe Ausgaben wie z. B. Pong realisiert werden [61]. Der Vorteil der Bibliothek ist die Unterstützung in JupyterLab, sodass die erzeugten Grafiken direkt im Editor angezeigt werden können, ohne ein eigenes Fenster öffnen zu müssen [60]. Dies rechtfertigt die Verwendung von Matplotlib, da für den Anwender die Benutzung erleichtert wird. Mögliche niedrigere Bildraten können hierfür in Kauf genommen werden.

Bei der Darstellung wird als erstes das Spielfeld mit den dazugehörigen Toren wie in Abbildung 6.28 markiert. Das Spielfeldbegrenzung entspricht dabei dem Rahmen der Zeichnung. Durch die Verwendung der gleichen Skalierung der Achsen kann das tatsächliche Seitenverhältnis beibehalten werden.
Die Stangen werden durch eine einfache Linie in der Farbe des Teams, welche automatisch zugewiesen wird, markiert. Da die Kollisionsdetektion mit einem kleinen Zylinder an den Füßen der Spielfigur durchgeführt wird und sich die Position je nach Verschiebung und Rotation ändern kann, werden die Zylinder in grau ebenfalls eingezeichnet. Dabei entscheidet die Position des Zylinders auf der Applikate über die Graustufe, damit unterschieden werden kann, ob die Spielfiguren z. B. auf dem Kopf oder auf ihren Füßen stehen. Je dunkler der Ton, desto niedriger ist der Körper für die Kollisionsdetektion positioniert. Für die bessere Übersicht werden die Zylinder durch zwei farbige Striche

zusätzlich mit der dazugehörigen Stange verbunden.
Der Ball kann durch einen schwarzen Kreis in der dem Radius entsprechenden Größe eingezeichnet werden. Verschiedene Farben für unterschiedliche Höhen werden dabei nicht verwendet, da davon ausgegangen wird, dass sich der Ball immer am Boden des Kickers befindet.
Ein Beispiel der grafischen Auswertung des Zustands ist Abbildung 6.30 zu entnehmen. Dabei gehören neben dem Ball zwei Mannschaften mit je zwei Reihen zum Tischkicker. In jeder Reihe sind drei Spielfiguren befestigt. Die einzelnen Reihen sind unterschiedlich weit verschoben und rotiert.

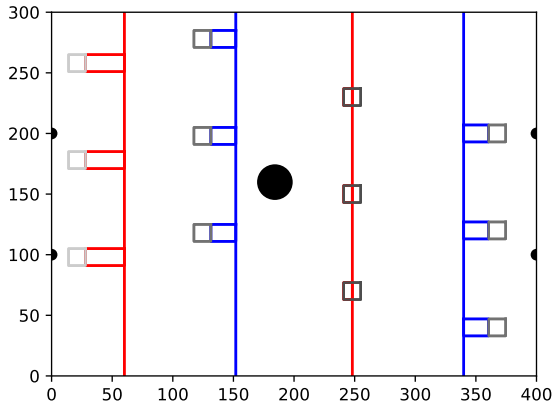


Abbildung 6.30: Darstellung eines Tischkickers mit mehreren Reihen und Ball in der Simulation

6.4.3 Gravitation

Die Simulation soll im 3-dimensionalen Raum stattfinden, damit eine realistische Interaktion zwischen den Spielfiguren und dem Ball erfolgt. Dabei besteht auch die Möglichkeit, dass der Ball einen Impuls in Richtung der Applikate erhält. Entweder ist der Impuls in den Boden oder umgekehrt gerichtet. Wenn er in die Richtung des Bodens verläuft, ist dies kein Problem, da dieser den Ball auffängt. Spätestens im anderen Fall wird die Notwendigkeit von Gravitation in der Simulation deutlich. Ohne sie würde der Ball kontinuierlich dem Impuls folgen und so aus dem Spielbereich fliegen und unerreichbar für die Spielfiguren werden. Allerdings ist dieser Fall bei den meisten Tischkickern sehr selten, da der Ball immer am Boden bleiben soll, damit niemand durch einen solchen Quer-

schläger verletzt wird. Auch aus diesem Grund hängen die Figuren bei einem Tischkicker etwas erhöht, sodass wie in Abbildung 6.31 der Berührungspunkt mit dem Ball nicht unterhalb seines Mittelpunkts liegt. Es kann jedoch trotzdem nicht gänzlich verhindert

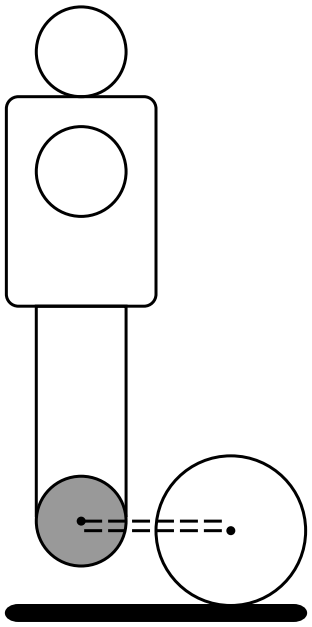


Abbildung 6.31: Aufhängung einer Tischkickerfigur im Verhältnis zur Ballposition

werden, dass der Ball keinen Aufwärtsimpuls erfährt. Da es sich dabei um relativ seltene Einzelfälle handelt, können diese für eine einfache Simulation, die kein perfektes Abbild der Realität sein soll, ignoriert werden. In diesem Fall wird die Implementierung der Gravitation unnötig, sofern verhindert wird, dass die Bewegungsrichtung des Balls auf der Applikante, also θ, verändert wird. Die Simulation beschränkt sich demnach, abgesehen von der Kollsisionsdetektion zwischen Spielfigur und Ball, auf eine 2-dimensionale Ebene. Dadurch kann zusätzlicher Arbeits- und Rechenaufwand gespart werden. Es wird z. B. keine Kollisionsdetektion mit dem Boden benötigt, die anderenfalls in fast jedem Schritt bedacht werden müsste, da der Ball bei fast jedem Zeitschritt in diesen hinein beschleunigt werden würde. Dabei entstehende Verluste durch Reibung können stattdessen durch simplere Berechnungen berücksichtigt werden.

6.4.4 Bewegen der Spielfiguren

Damit ein Spiel überhaupt erst möglich wird, müssen die verschiedenen Reihen von der KI bewegt werden können. Hierfür muss die Position und Geschwindigkeit der Bewegung für jede Reihe individuell anpassbar sein. Dabei wird die Implementierung dem realen Aufbau auf der Hardware nachempfunden.
Wie in Abschnitt 6.2 beschrieben, kann mithilfe der eingestellten maximalen Beschleunigungen der Motoren die angepasste Winkelgeschwindigkeit über Formel 6.3 berechnet werden. Zwar wird in der Simulation die Berechnung der Geschwindigkeit nicht im gleichen Abstand wie auf der Hardware aufgerufen, durch die Berücksichtigung der Zeitschritte kann jedoch die konstante Beschleunigung berechnet werden. Die jeweilige Geschwindigkeit wird für den gesamten Zeitschritt verwendet, wodurch bei größeren Zeitschritten gewisse Abweichungen entstehen können. Da die Zeitschritte in der Simulation relativ klein bleiben, ist der Fehler vernachlässigbar.
Auch hier erfolgt eine Umrechnung der Winkelgeschindigkeit in die Verzögerung über die Formel 6.1. Die Geschwindigkeit der Spielfiguren durch die Verschiebung v_{shift} und die Geschwindigkeit der Zylinder an den Beinen der Figuren v_{rot} können über

$$v_{shift} = \frac{step\_width_{shift}}{T_m} \tag{6.18}$$

$$v_{rot} = \frac{step\_width_{rot}}{T_m} \tag{6.19}$$

berechnet werden. $step\_width$ entspricht hierbei der Distanz, die in einem Schritt zurückgelegt wird. Die Position der Verschiebung kann einfach mit

$$y_{offset} = y_{offset} + v_{shift} \cdot T \tag{6.20}$$

bestimmt werden, während beim Winkel eine Umrechnung durch

$$\alpha_r = \alpha_r + T \cdot \frac{v_{rot}}{2 \cdot z_{figure}} \tag{6.21}$$

nötig ist. Der Zeitschritt zwischen zwei Berechnungen der Position der Spielfiguren entspricht hier T.
Mit diesen einfachen Mitteln lässt sich die tatsächliche Bewegung der realen Motoren gut annähern.

6.4.5 Bewegung des Balls

Der wichtigste Teil der Simulation ist die Bewegung des Balls, denn sie entscheidet letztlich über das Spielgeschehen. Durch die zuvor angesprochene Art der Speicherung des Zustands des Balls ist dieser Teil einfach realisierbar. Die benötigten Informationen sind hierbei im Objekt als Parameter gespeichert und können für die Berechnung der nächsten Position abgerufen werden. Die Parameter x, y und z beschreiben die aktuelle Position im Koordinatensystem, die Geschwindigkeit des Balls wird durch v angegeben und die Winkel φ und θ bestimmen die Bewegungsrichtung im Raum. Ein Überblick über die Parameter ist in Abbildung 6.32 gegeben. Dabei ist die aktuelle Position zur besseren Übersicht im Ursprung eingezeichnet. Die Winkel und Geschwindigkeit werden natürlich immer von der aktuellen Position des Balls aus berechnet, da sie sonst keine Aussagekraft hätten. Für den Polarwinkel θ ist zu beachten, dass dieser nur im Intervall $[0, \pi]$ gültig ist, um unnötige Dopplungen in der Definition zu vermeiden.

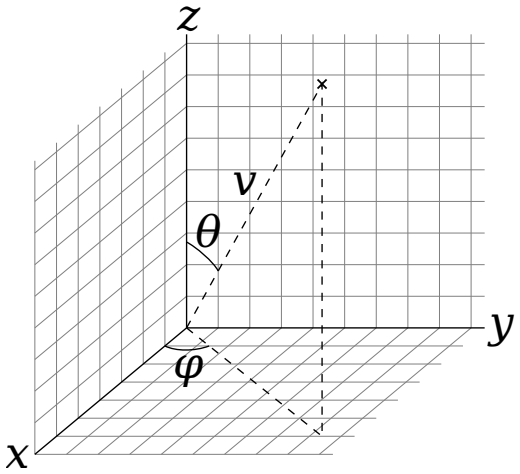


Abbildung 6.32: Räumliche Einordnung der Parameter der Bewegungsrichtung des Balls [105]

Die Berechnung der nächsten Position kann mithilfe der angegebenen Parameter durchgeführt werden. Zusätzlich wird jedoch noch der Zeitschitt T zwischen den zwei Zeitpunkten benötigt, welcher flexibel über die Geschwindigkeit des Balls berechnet wird, sodass die Anforderung SN4 erfüllt ist. So lassen sich die neuen Koordinaten (x', y', z')

vom Ausgangspunkt (x, y, z) durch die Formeln

$$x' = x + T \cdot v \cdot \sin(\theta) \cdot \cos(\varphi), \tag{6.22}$$

$$y' = y + T \cdot v \cdot \sin(\theta) \cdot \sin(\varphi), \tag{6.23}$$

$$z' = z + T \cdot v \cdot \cos(\theta) \tag{6.24}$$

berechnen. Dieser Schritt muss mehrfach wiederholt werden und beschreibt so die Bewegung des Balls.
Durch die Aktualisierung der Position allein ist die Simulation jedoch noch nicht vollständig, denn erst durch die Interaktion des Balls mit seiner Umgebung lassen sich die Spielmechaniken umsetzen. Hierfür muss nach jeder Positionsberechung eine Kollisionsdetektion erfolgen, welche bei Kontakt mit anderen Objekten einen Impuls auf den Ball auswirken kann.

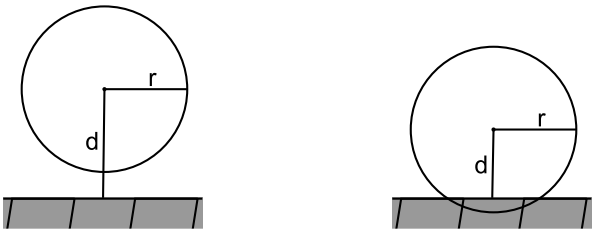
6.4.6 Kollisionsdetektion

Nach der Berechnung einer neuen Position des Balls, muss eine Kollisionsdetektion durchgeführt werden, um mögliche Kontakte mit anderen Objekten zu erkennen. Dabei unterteilt sich die Detektion in mehrere Bereiche, welche jeweils den Zusammenstoß von dem Ball mit einem spezifischen Objekt überprüfen. Durch das Vernachlässigen der Gravitation kann kein Kontakt mit dem Boden auftreten, weshalb hierfür keine Erkennung benötigt wird. Anders ist dies jedoch bei den Wänden, welche das Spielfeld begrenzen und gleichzeitig die Tore beinhalten. Zu der Kollisionsdetektion der Wände gehört deshalb auch die Erkennung von geschossenen Toren. Ein weiterer Bereich ist die Kollision mit den Spielfiguren, welche eine wesentlich komplexere Form als die Wände besitzen.
Bei der Programmierung einer komplexen 3-dimensionalen Kollisionsdetektion wird meist mit Verfahren wie dem SAT gearbeitet. Dieses Verfahren benötigt jedoch eine Unterteilung runder Körper in ein Netz aus Punkten. Je genauer dieser Körper dargestellt werden soll, desto mehr Punkte werden benötigt, wodurch ein verhältnismäßig hoher Rechenaufwand entsteht. Da in dieser Simulation nur die Kollision des Balls mit den Wänden und Figuren erfolgt, kann auch eine genaue manuelle Bemessung des Abstands erfolgen. Besonders die Eigenschaften des Balls machen dies erst möglich.
Die Kollisionsdetektion wird, wie in SN4 gefordert, nach spätestens $\Delta s = 2$ mm Bewegung des Balls durchgeführt, sodass dieser nicht unbemerkt durch andere Objekte rollen kann.

Im Folgenden werden die Vorgehensweisen für die zwei verschiedenen Arten der Kollision beschrieben. Für spätere Anpassungen der Bewegungsrichtungen ist auch die Ermittlung des Berührungspunkts sinnvoll.

Wände

Zunächst wird die Kollision mit den Wänden untersucht. Diese kann ermittelt werden, indem der Abstand des Ballmittelpunkts zu diesen gemessen wird. Hierfür ist lediglich eine Überprüfung in der 2-dimensionalen xy-Eben nötig, da die Wände entlang der Applikate verlaufen und der Mittelpunkt des Balls logischerweise zu jedem Punkt auf seiner Schale einen Abstand entsprechend seines Radius hat. Wenn sich der Ball einer Wand so annähert, dass der Mittelpunkt einen Abstand kleiner des Radius zu der Wand besitzt, liegt eine Kollision vor. In Abbildung 6.33 wird dies beispielhaft dargestellt.



(a) $d > r$ keine Kollision (b) $d \leq r$ Kollision liegt vor

Abbildung 6.33: 2-dimensionale Kollisionsdetektion zwischen Ball und Wand mittels Abstand d und Radius r

Die Kollisionsdetektion zwischen dem Ball und den Wänden hängt also nur vom Abstand d dieser ab. Die Berechnung des Abstands ist durch die Art der Positionierung des Spielfelds sehr einfach, da die Wände alle parallel zur Abszisse oder Ordinate verlaufen. Es muss dementsprechend nur für die Wände, die sich parallel zur Ordinate befinden, die Position der Wand auf der Abszisse mit der des Ballmittelpunkts verglichen werden. Gleiches gilt andersherum für die Wände parallel zur Abszisse. Dies ist dabei auch immer völlig unabhängig von der Bewegungsrichtung. Der Berührungspunkt setzt sich auch immer aus zwei Werten des Ballmittelpunkts (also z und entweder x oder y) und der jeweiligen Position der Wand auf der Achse orthogonal zu ihrem Verlauf zusammen.

Eine Besonderheit bilden die beiden Tore an den Stirnseiten des Tischkickers, welche den Ball natürlich nicht stoppen sollen. Wenn der Ball sich mit dem Mittelpunkt zwischen den zwei Pfosten des Tors befindet, muss eine genauere Prüfung durchgeführt werden, denn eine Berührung mit den Pfosten kann auch nicht gänzlich ausgeschlossen werden. Der Ball kann wie in Abbildung 6.34 bereits zum Teil die Torlinie überquert haben und erst später seitlich die Torpfosten berühren.

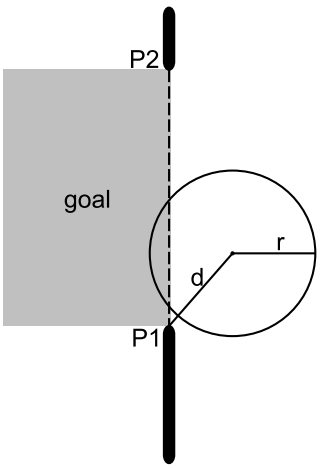


Abbildung 6.34: Übertretung der Torlinie ohne Kollision

Wenn der Ball sich im Torbereich befindet, muss der Abstand des Ballmittelpunkts zu den zwei Pfosten ermittelt werden. Erst wenn der Abstand zu einem dieser Punkte kleiner wird als der Radius des Balls, liegt eine Berührung vor. Die Distanz auf der xy-Ebene lässt sich einfach durch die Formel

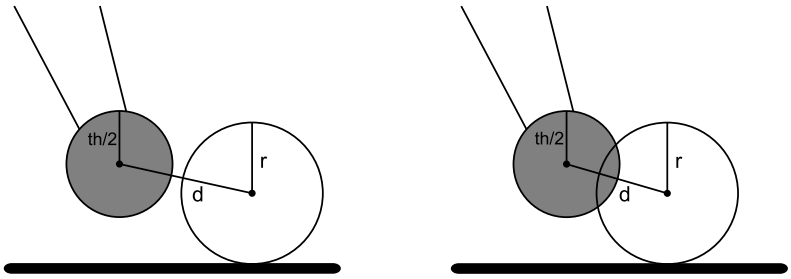
$$d = \sqrt{(x_{Ball} - x_P)^2 + (y_{Ball} - y_P)^2} \tag{6.25}$$

berechnen. Sollte ein Kontakt ermittelt werden, gilt der jeweilige Pfosten natürlich als Berührungspunkt.

Im Rahmen der Kollisionsdetektion mit den Wänden kann auch ein geschossenes Tor ermittelt werden. Dabei wird nur überprüft, ob der Ball die Torlinie mit vollem Umfang überschritten hat. Erst wenn dies der Fall ist, wird ein Tor gezählt.

Spielfiguren

Nach der Überprüfung eines Kontakts mit den Wänden, muss dieser Schritt auch für die Spielfiguren durchgeführt werden. Diese haben eine komplexere Struktur und können unter anderem auch in ihrer Position auf der Applikate werden. Aus diesem Grund muss bei den Spielfiguren eine Kollisionsdetektion im 3-dimensionalen Raum durchgeführt werden. Wie im Unterabschnitt 6.4.1 in Abbildung 6.29 beschrieben, wird die Form der Figur durch einen einfachen Zylinder an den Füßen der Spielfigur angenähert. Es gilt also eine mögliche Berührung der Zylinder mit dem Spielball zu ermitteln.
Da es relativ viele Figuren auf dem Tisch geben kann und die Detektion eine gewisse Zeit in Anspruch nimmt, sollte im ersten Schritt bereits eine grobe Auswahl getroffen werden. Dies kann über die einzelnen Reihen erfolgen, da alle Figuren sich auf der gleichen Position auf der Abszisse und Applikate befinden. Es muss für die gesamte Reihe nur der Abstand zum Ball in dieser Ebene überprüft werden, um möglicherweise eine Kollision auszuschließen bzw. genauere Überprüfungen durchzuführen, wenn der Abstand kleiner als der Radius des Balls und des Zylinders addiert ist. Die Berechnung der Distanz kann analog zu der in Gleichung 6.25 erfolgen. In Abbildung 6.35 ist ein Beispiel der Kollisionsdetektion in der xz-Ebene aufgezeigt.



(a) $d > r + \frac{th}{2}$ keine Kollision (b) $d \leq r + \frac{th}{2}$ Kollision liegt vor

Abbildung 6.35: Grobe 2-dimensionale Kollisionsdetektion zwischen Ball und Figuren in der xz-Ebene

Bevor jedoch diese Überprüfung stattfinden kann, muss erst der Mittelpunkt des Zylinders ermittelt werden, da sich dieser mit dem Winkel der Rotation der Reihe verändert. Ausgehend von der Stange lässt sich mit dem Rotationswinkel α_r der Mittelpunkt des Zylinders wie in Abbildung 6.36 berechnen.

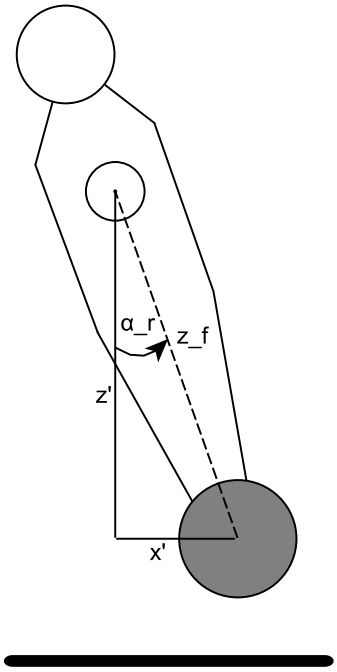


Abbildung 6.36: Position des Zylinders der rotierten Spielfigur

Dabei wird die Position auf der Abszisse über

$$x_{cyl} = x_{row} + z_{figure} \cdot \sin(\alpha_r) \tag{6.26}$$

und der Wert der Applikate durch

$$z_{cyl} = z_{row} + z_{figure} \cdot \cos(\alpha_r) \tag{6.27}$$

ermittelt.
Sofern der Abstand in der xz-Ebene klein genug ist, kann für die einzelnen Figuren auf der Stange geprüft werden, ob eine Berührung vorliegt. Zunächst erfolgt erneut eine grobe Einordnung. Der Mittelpunkt des Balls darf dabei auf der Ordinate maximal einen Abstand von $d_y \leq r$ zu der jeweiligen Spielfigur haben. Anderenfalls liegt kein Kontakt vor. Dies wird in Abbildung 6.37 dargestellt.

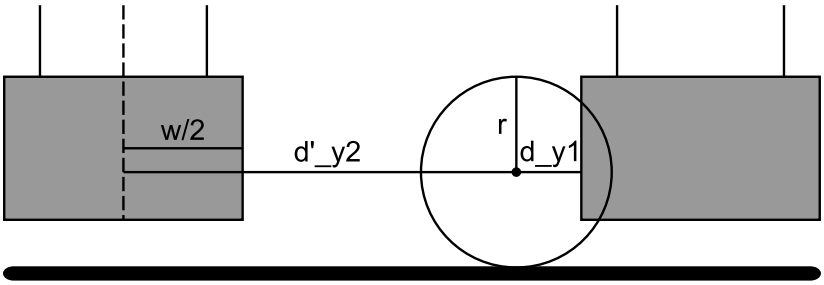
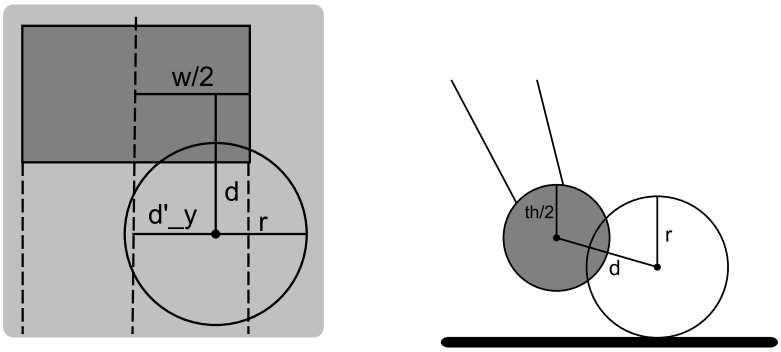


Abbildung 6.37: Grobe Kollisionsdetektion auf der Ordinate

Um die Überprüfung durchführen zu können, muss zunächst die Position der Figur auf der Ordinate über

$$y_{mid} = y_{figure} + y_{offset} \tag{6.28}$$

berechnet werden. Anschließend wird eine Ermittlung durchgeführt, auf welcher Seite sich der Ballmittelpunkt befindet und wie groß sein Abstand zum y_{mid} ist. Sollte der Abstand $d'_y \leq r + \frac{w}{2}$ sein, ist ein Kontakt mit der Figur möglich und eine genauere Kontrolle muss erfolgen.
Diese ist in drei Fälle gegliedert. Im ersten Fall ist, wie in Abbildung 6.38 gezeigt, der Abstand des Ballmittelpunkts zur Mitte der Figur auf der Ordinate $d'_y \leq \frac{w}{2}$.



(a) Draufsicht (b) Seitenansicht

Abbildung 6.38: Kollision bei Positionierung des Ballmittelpunkts innerhalb der Breite des Spielfigur auf der Ordinate

In diesem Fall liegt definitiv eine Berührung vor, da der volle Radius des Balls für die Abstandsbestimmung in der xz-Ebene genutzt werden kann und dieser Fall bereits geprüft wurde. Als Berührungspunkt kann dabei die Lage des Balls auf der Ordinate und die Position des Zylinders auf der xz-Ebene genommen werden. Dies ist zwar nicht der genaue Berührungspunkt, jedoch ist vor allem die Richtung für die spätere Veränderung der Bewegung wichtig.

Wenn der erste Fall nicht erfüllt wird, liegt im zweiten Fall eine Kollision vor, wenn der Abstand $d \leq \frac{th}{2}$ ist. Dabei berührt der Ball die Seite der Spielfigur. Als Berührungspunkt wird hier die Position des Ballmittelpunkts auf der xz-Ebene und die Lage der Kante der Spielfigur auf der Ordinate verwendet.

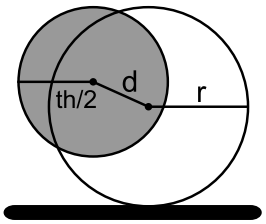
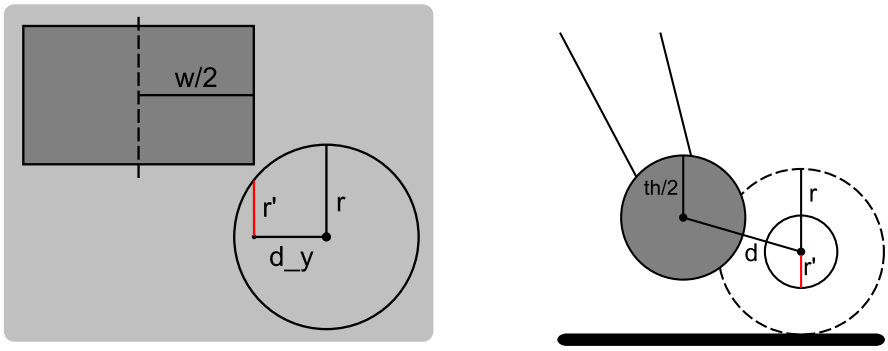


Abbildung 6.39: Seitliche Kollision von Ball und Zylinder bei $d \leq \frac{th}{2}$



(a) Draufsicht (b) Seitenansicht

Abbildung 6.40: Prinzip der Kollisionsdetektion an der Kante einer Spielfigur

Sofern keiner der ersten beiden Fälle eintritt, muss eine Überprüfung der Kollision mit der äußeren Kante der Spielfigur erfolgen. Das Problem hierbei ist, dass in der xz-Ebene nicht mehr der volle Radius des Balls verwendet werden kann. Abbildung 6.40 zeigt diesen

Fall, wo die Distanz ursprünglich klein genug war, aber durch die Lage auf der Ordinate nicht mehr ausreicht.

In diesem Fall muss zunächst der Radius des Balls am Punkt auf der Ordinate errechnet werden. Eine Übersicht hierfür ist in Abbildung 6.41 gegeben.
Der Öffnungswinkel wird nach Umstellen der Formel

$$d_y \;=\; r \cdot \cos(\alpha) \tag{6.29}$$

durch

$$\alpha \;=\; \arccos\left(\frac{d_y}{r}\right) \tag{6.30}$$

errechnet. Mit dem Winkel lässt sich anschließend der angepasste Radius

$$r' \;=\; r \cdot \sin(\alpha) \tag{6.31}$$

ermitteln. Mithilfe dieses neuen Radius kann überprüft werden, ob der Abstand des Zylinders vom Ball $d \leq r' + \frac{th}{2}$ ist. In diesem Fall liegt eine Kollision vor. Als Berührungspunkt wird dann wieder der Mittelpunkt auf der xz-Ebene und die Lage der Kante des Zylinders verwendet.
Sollte keiner der drei Fälle eintreten, ist kein Kontakt erfolgt.

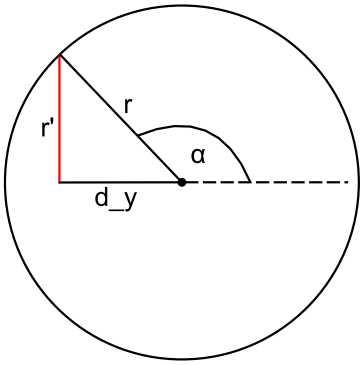


Abbildung 6.41: Ermittlung des neuen Radius r' an der Kante einer Spielfigur

6.4.7 Korrektur der Bewegung und Position des Balls

Sollte bei der Kollisionsdetektion ein Kontakt ermittelt werden, muss die neuen Bewegungsrichtung des Balls berechnet werden. Diese ist abhängig von der aktuellen Bewegungsrichtung und dem Berührungspunkt mit dem anderen Objekt.

Um zu verstehen wie sich die Richtung des Balls ändert, wenn er auf eine Wand trifft, lohnt sich zunächst der Blick auf die Reflexion von Licht. Wenn ein Lichtstrahl auf einen flachen Spiegel trifft, entspricht wie in Abbildung 6.42 dargestellt der Einfallswinkel dem Ausfallswinkel [30].

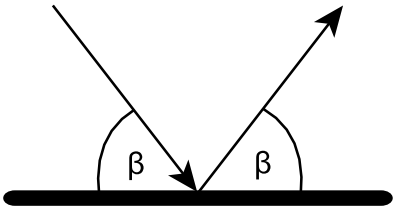


Abbildung 6.42: Reflexion eines Lichtstrahls durch einen Spiegel

Ähnlich gilt dies auch für physische Objekte. Beim Billard z. B. wird stets davon ausgegangen, dass der Einfallswinkel und der Ausfallswinkel bei einem Zusammenstoß mit der Wand gleich bleibt [18, 54]. Dies ist jedoch nicht ganz korrekt, wenn Verluste beachtet werden [5]. Die Verluste beim Billard sind aufgrund der Eigenschaften der Kugeln und des Tischs sehr gering, weshalb der Stoß einer Billardkugel als nahezu elastisch gilt und somit in der Realität kaum ein Unterschied zwischen den Winkeln zu erkennen ist.

Bei dem ausgewählten Tischkicker sind die Verluste bei Kontakt mit dem Tischkicker wesentlich höher, wodurch sich die Bewegungsrichtung des Balls stärker von der idealen Reflexion unterscheidet. Ein über dem Kicker fallen gelassener Ball springt beispielsweise nicht, sondern bleibt nach Kontakt sofort liegen. Hierbei handelt es sich um einen plastischen Stoß.
Die Berührung des Balls mit den Wänden und Spielfiguren ist hingegen weder elastisch, noch plastisch. Um den Verlust bei einer realistischen Kollision berechnen zu können, bedarf es dem Restitutionskoeffizient k des Spielballs. Dieser Koeffizient beschreibt, wie hoch der Verlust von kinetischer Energie bei einem Zusammenstoß ist. Wenn $k = 0$ ist,

dann handelt es sich um einen plastischen Stoß mit maximalem Energieverlust. Der elastische Stoß ist auf der anderen Seite des Spektrums und hat mit $k = 1$ keine Verluste [56]. Die Ermittelung dieses Koeffizienten kann durch einen einfachen Versuch erfolgen. Dieser wird im Folgenden erklärt und durchgeführt.

Die Geschwindigkeit von zwei Objekten nach einem realistischen Zusammenstoß können durch die Formeln

$$v_1' = \frac{m_1 \cdot v_1 + m_2 \cdot v_2 - m_2 \cdot (v_1 - v_2) \cdot k}{m_1 + m_2} \tag{6.32}$$

und

$$v_2' = \frac{m_1 \cdot v_1 + m_2 \cdot v_2 - m_1 \cdot (v_2 - v_1) \cdot k}{m_1 + m_2} \tag{6.33}$$

berechnet werden, wobei m_x der Masse und v_x der Geschwindigkeit vor dem Zusammenprall des jeweiligen Objektes entsprechen. Wenn eines der Objekte extrem schwer und unbeweglich wie z. B. eine Wand ist und vor dem Stoß eine Geschwindigkeit von $v_1 = 0$ besitzt, dann kann davon ausgegangen werden, dass auch $v_1' \approx 0$ ist. So kann die Formel auf

$$v_2' = \frac{m_2 \cdot v_2 - m_1 \cdot (v_2) \cdot k}{m_1 + m_2} \tag{6.34}$$

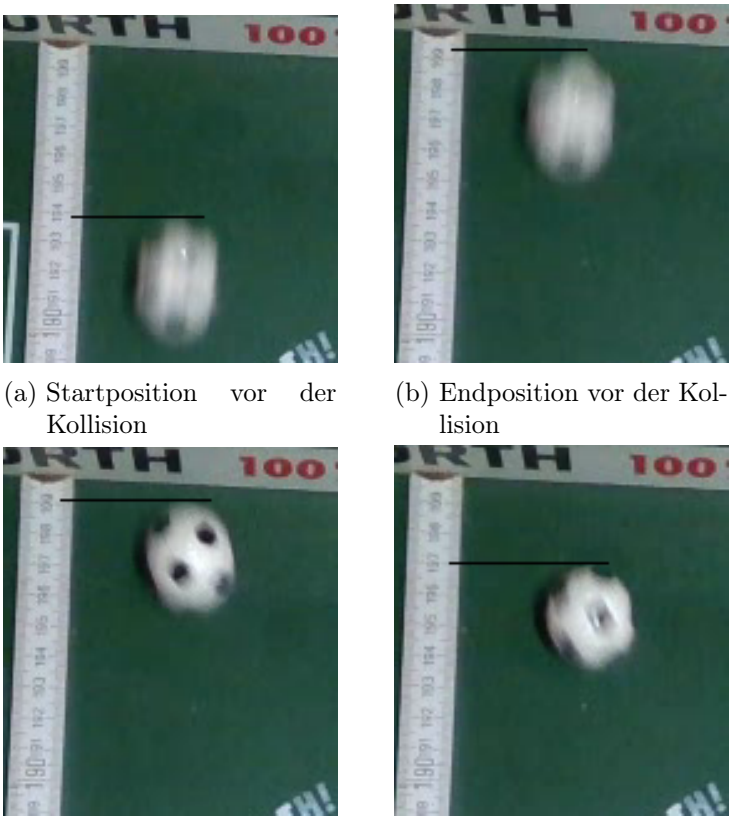
vereinfacht werden. Mit einem hohen Wert für $m_1 \gg m_2$ kann die Geschwindigkeit über

$$v_2' \approx -v_2 \cdot k \tag{6.35}$$

angenähert werden. Mithilfe eines Versuches, bei dem die Geschwindigkeit des Balls vor und nach dem Stoß mit einer Wand gemessen wird, kann über die Formel

$$k \approx -\frac{v_2'}{v_2} \tag{6.36}$$

der Restitutionskoeffizient bestimmt werden.
Zur Bestimmung des Koeffizienten, wird ein Versuch durchgeführt, bei dem der Tischkickerball gegen eine Wand gespielt wird und die Geschwindigkeit vor- und nachher gemessen wird. Durch eine Kamera lässt sich dies bewerkstelligen. Die Abbildung 6.43 zeigt die Messung des Versuchs.



(a) Startposition vor der Kollision

(b) Endposition vor der Kollision

(c) Startposition nach der Kollision

(d) Endposition nach der Kollision

Abbildung 6.43: Messung der Position des Balls zur Bestimmung seiner Geschwindigkeit vor und nach einer Kollision mit der Spielfeldwand

Die abgelesenen Werte auf dem Maßband neben dem Ball betragen $s_{00} = 193,8$ cm, $s_{01} = 199,4$ cm, $s_{10} = 199,2$ cm und $s_{11} = 197,0$ cm. Zwischen den Bildern ist ein Periode von $T = \frac{2}{30}$ s. Die Geschwindigkeiten werden dementsprechend mit

$$v_2 = \frac{s_{00} - s_{01}}{T} = 0,84 \frac{\text{m}}{\text{s}} \tag{6.37}$$

und

$$v_2' = \frac{s_{10} - s_{11}}{T} = 0,33 \frac{\text{m}}{\text{s}} \tag{6.38}$$

berechnet. Daraus ergibt sich ein Restitutionskoeffizient von

$$k_{wall} = -\frac{v_2'}{v_2} = 0.393. \tag{6.39}$$

Dieser Wert kann allerdings nicht universell eingesetzt werden. Durch das andere Material der Spielfiguren weicht der Koeffizient für den Kontakt mit ihnen ab. Ein Versuch zur Bestimmung von k_{figure} ist hierbei jedoch nicht gut umsetzbar, da die Figuren beweglich sind und z. B. durch einen etwas loseren Griff des Spielers ihre Stoßzahl verändern. Über Beobachtungen verschiedener Versuche, wird der Koeffizient auf $k_{figure} \approx 0.1$ geschätzt. Mithilfe dieser Werte kann über die Gleichung 6.33 eine realistische Kollision simuliert werden. Es muss jedoch beachtet werden, dass die Formel nur für eine 1-dimensionale Kollision gültig ist, während die Simulation im 2-dimensionalen Bereich erfolgt. Um dieses Problem zu lösen, muss der Bewegungsvektor des Balls in zwei Komponenten zerlegt werden. Ein Skalar beschreibt dann die Bewegungsgeschwindigkeit orthogonal und der andere parallel zur Aufprallfläche. Die Geschwindigkeit nach dem Aufprall wird für die senkrechte Komponente durch die Gleichung 6.33 berechnet, während die parallele Geschwindigkeit nicht beeinflusst wird. Im Beispiel der Kollision mit einer Wand wird der orthogonal zu ihr stehende Vektor negiert und verkürzt. Da die andere Komponente nicht verändert wird, ist der Ausfallswinkel etwas kleiner als der Einfallswinkel. Dieses Prinzip wird in Abbildung 6.44 dargestellt.

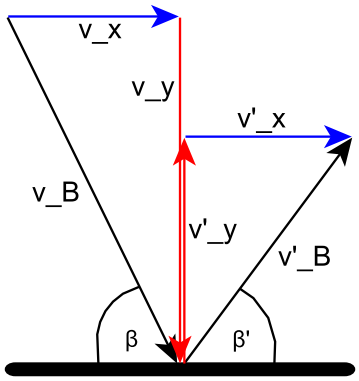


Abbildung 6.44: Realistische Reflexion eines Balls an einer festen Wand

Die eigentliche Berechnung der neuen Bewegungsrichtung kann an einem Beispiel wie in Abbildung 6.45 erklärt werden. Da die Bewegung auf der Applikate nicht berücksich-

tigt werden soll, ist hier nur die Berechnung des neuen Azimutwinkel φ' und der neuen Geschwindigkeit interessant.

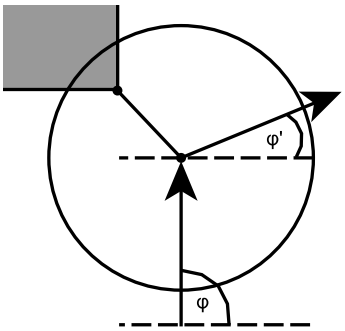
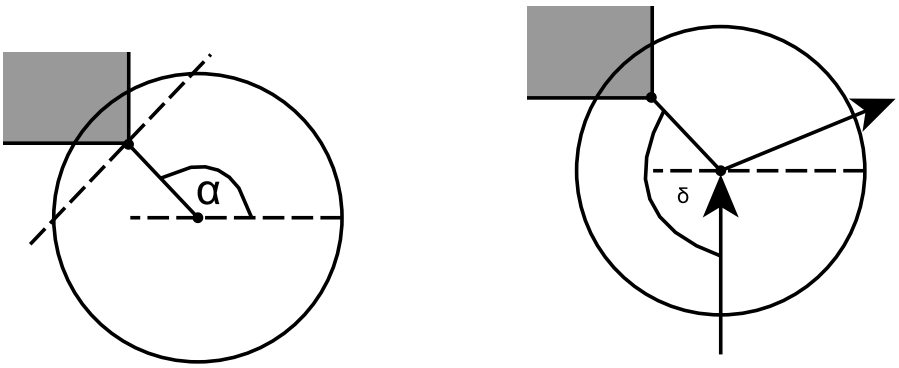


Abbildung 6.45: Bewegungsrichtung des Balls vor und nach einer Kollision

Aufgrund der runden und symmetrischen Geometrie des Balls kann bei einer Kollision jeder Berührungspunkt als eine Fläche betrachtet werden, die den Vektor zwischen sich selbst und Ballmitte als Normalenvektor besitzt (siehe Abbildung 6.46). So kann die Ermittlung der Reflexion genau wie in Abbildung 6.44 erfolgen.



(a) Winkel α zum Berührungspunkt (b) Winkel $\delta = \varphi_{Ball} + \pi - \alpha$

Abbildung 6.46: Winkel zum Berührungspunkt

Für die Berechnung ist dabei der Winkel α interessant, welcher die Lage der Spiegelachse beschreibt. Er kann nach der Subtraktion des Berührungspunkts P_1 vom Ballmittelpunkt

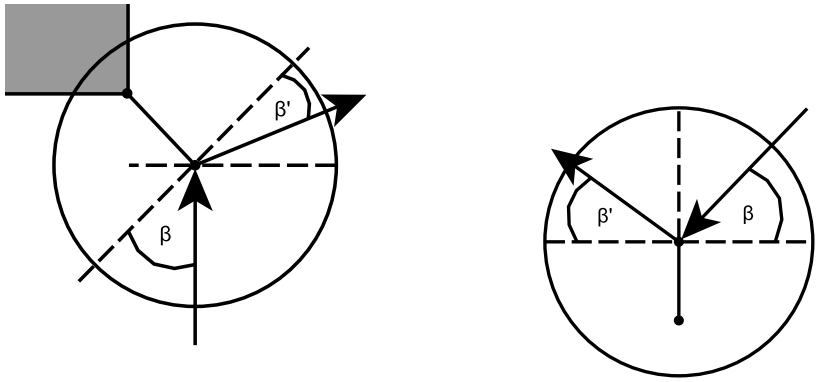
P_0 aus dem entstandenen Vektor $\vec{v}_c$ durch eine Umwandlung vom kartesischem in das Kugelkoordinatensystem ermittelt werden. Die Berechnung wird über die Formeln

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2}, \tag{6.40}$$

$$\varphi = \arccos\left(\frac{v_z}{v}\right), \tag{6.41}$$

$$\theta = \arctan\left(\frac{v_y}{v_x}\right) \tag{6.42}$$

durchgeführt [95]. α entspricht hierbei dem Ergebnis von φ. Der Polarwinkel soll zwar nicht verändert werden, aber der Winkel $\gamma = \theta$ wird dennoch später benötigt, um die Geschwindigkeit anzupassen.



(a) Reales Beispiel (b) Um $-\alpha$ rotierte Ansicht

Abbildung 6.47: Reflexionswinkel am Berührungspunkt

Für die Berechnung der Reflexion wird der Einfallswinkel β benötigt. Für seine Berechnung muss zunächst α vom umgekehrten Azimutwinkel $\varphi_{Ball,reverse} = \varphi_{Ball} + \pi$ der Bewegungsrichtung abgezogen werden sowie weitere $\frac{\pi}{2}$ vom absoluten Wert, da die Ebene orthogonal zum Winkel ist. Wenn $\frac{\pi}{2} > \varphi_{Ball,reverse} - \alpha$ ist, handelt es sich um eine theoretisch unmögliche Kollision, die nur durch die begrenzte Zahl an Abtastzeitpunkten entsteht. In diesem Fall muss keine Veränderung der Bewegung vorgenommen werden, da der Ball das Objekt nur leicht seitlich streift. Ansonsten wird wie in Abbildung 6.47

die Bewegungsrichtung so rotiert, dass durch die Berechnungen

$$v_{plane} = v \cdot \sin(\beta) \tag{6.43}$$

$$v_{rest} = v \cdot \cos(\beta) \tag{6.44}$$

die Bewegung in eine Komponente senkrecht v_{plane} und in eine parallel v_{rest} zur Ebene zerlegt wird. Nach dem Vorbild aus Formel 6.35 kann hier die Geschwindigkeit orthogonal zum Hindernis durch

$$v'_{plane} \;=\; -\, v_{plane} \cdot k \tag{6.45}$$

berechnet werden. Dies gilt auch für die Kollision mit den Spielfiguren, da für die Simulation davon ausgegangen wird, dass die Stangen auch bei Kontakt fest in ihrer Position gehalten werden. Anschließend wird die neue Geschwindigkeit

$$v' \;=\; \sqrt{v'^2_{plane} + v^2_{rest}} \tag{6.46}$$

bestimmt, mit der sich weiter der Ausfallswinkel

$$\beta' \;=\; \arcsin\left(\frac{v'_{plane}}{v'}\right) \tag{6.47}$$

berechnet.

Zuletzt muss noch der Winkel der neuen Bewegungsrichtung über

$$\varphi'_{Ball} \;=\; 2 \cdot \alpha - \varphi_{Ball} + \pi + \beta - \beta' \tag{6.48}$$

bestimmt werden.
Zwar soll die vertikale Bewegungsrichtung nicht verändert, ihr Einfluss allerdings dennoch beachtet werden. Die Geschwindigkeit in der xy-Ebene nimmt je nach Polarwinkel ab. Die Geschwindigkeit in der xy-Ebene kann durch

$$v'' = v' \cdot \cos(\gamma) \tag{6.49}$$

angepasst werden.
Mit dem beschriebenen Lösungsansatz können Kollisionen des Balls mit ruhenden Objekten berechnet werden. Bei einem Tischkicker sind allerdings die Spielfiguren häufig in Bewegung, wenn sie auf den Ball treffen, da so überhaupt erst der Ball geschossen wird.

Für diese Fälle kann einfach die Bewegung der Spielfiguren mit der des Balls addiert werden. Hierfür werden beide Bewegungen in ihre x, y und z Komponenten zerlegt. Für den Ball kann dies ähnlich wie in Formel 6.22 erfolgen. Bei der Spielfigur werden die Geschwindigkeiten v_{shift} und v_{rot} genutzt, wobei $v_{f,y} = v_{shift}$ ist. Aus v_{rot} lassen sich wie in Abbildung 6.48 die anderen beiden Geschwindigkeiten durch

$$v_{f,x} = v_{rot} \cdot \cos(\alpha_r) \tag{6.50}$$

$$v_{f,z} = v_{rot} \cdot \sin(\alpha_r) \tag{6.51}$$

bestimmen, wobei der Winkel α_r hier dem Winkel der jeweiligen Reihe entspricht.

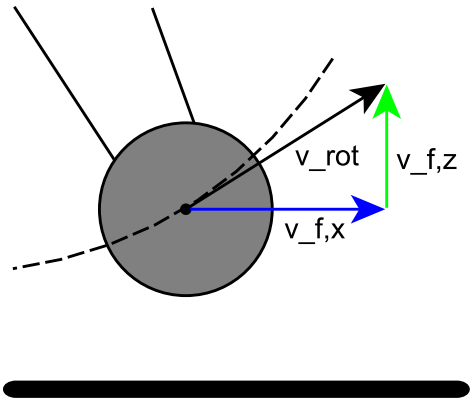


Abbildung 6.48: Zerlegung der Rotationsgeschwindigkeit der Reihe in seine x und z Komponente

Nach der Addition der Vektoren kann das Ergebnis wie in Formel 6.40 zurück in Kugelkoordinaten umgewandelt werden. Von hier aus kann wie bei einem ruhenden Ball vorgegangen werden. Es ist jedoch zu beachten, dass die addierte Geschwindigkeit der Figur nach der Korrektur wieder vom Ball abgezogen werden muss.

Bei jeder Korrektur der Position wird zunächst die Position entgegen der alten Bewegungsrichtung um einen halben Zeitschritt *dt* zurückgesetzt und anschließend mit der neuen Bewegungsrichtung und Geschwindigkeit für den nächsten halben Zeitschritt berechnet. Hierdurch wird verhindert, dass der Ball in das Objekt eindringt, aber aufgrund der geringeren Geschwindigkeit nicht bei der nächsten Berechnung das Objekt verlassen hat. Dieser Ansatz ist zwar nicht besonders genau, aber durch die kleinen Zeitschritte präzise genug.

Das hier vorgeschlagene Modell zur Berechnung der Bewegung des Balls ist nicht perfekt, da z. B. Effekte durch Ballspin nicht beachtet werden. Eine perfekte Modellierung ist jedoch ohnehin nicht möglich und nötig. Deshalb wird von weiteren Schritten abgesehen, da die Effekte komplex sind und einen eher geringen Einfluss auf den Spielverlauf nehmen [59].

6.4.8 Reibung

Zu einem realistischen System gehören auch Verluste, die z. B. durch Reibung auftreten. Bei dem Tischkicker bedeutet dies eine Verringerung der Geschwindigkeit. Auf den kurzen Strecken des Tischkickers sind diese Verluste bei den gleichzeitig recht hohen Geschwindigkeiten eher gering und weniger spielentscheidend. Dies steht im Kontrast zu den Verlusten bei einer Kollision, welche auch die Bewegungsrichtung verändern und so einen größeren Einfluss auf das Spiel nehmen können. Trotzdem kann eine einfache Funktion implementiert werden, welche die Geschwindigkeit mit einem Reibungskoeffizienten μ multipliziert. Dabei errechnet sich die neue Geschwindigkeit durch

$$v' \;=\; v \cdot (\mu + (1-\mu) \cdot (1 - \Delta t)), \tag{6.52}$$

sodass bei einem Zeitschritt von $\Delta t = 1\ s$ die Geschwindigkeit durch $v' = v \cdot \mu$ berechnet wird. Der Reibungskoeffizienten kann nach Bedarf angepasst werden.

6.5 KI

Nach dem Aufbau der Hardware und der Implementierung der Simulation kann die Entwicklung der KI erfolgen. Zunächst muss eine geeignete Umgebung für das Training geschaffen werden, welche als Schnittstelle zwischen der KI und dem Tischkicker dient. Anschließend kann das Training in verschiedenen Einsatzszenarien erfolgen, wobei unterschiedliche Ansätze und Einstellungsmöglichkeiten verglichen werden.

6.5.1 Umgebung

Im ersten Schritt muss die Umgebung für das RL geschaffen werden. Hierbei muss jedoch unterschieden werden zwischen der Umgebung für das Training mit der Simulation und

der Umgebung für den Test auf der realen Hardware.
In beiden Fällen wird die Umgebung als ein OpenAI Gym-Environment implementiert.

Gym

OpenAI Gym ist eine Sammlung unterschiedlicher Werkzeuge für RL [14]. Gym dient dabei als standardisierte Schnittstelle zwischen RL-Agenten und verschiedenen Umgebungen. Es gibt eine große Auswahl an unterschiedlichsten Umgebungen, die genutzt werden können, um RL-Verfahren zu testen. Es besteht auch die Möglichkeit eigene Umgebungen für spezifische Anwendungsfälle zu implementieren. Aus diesem Grund sind sehr viele RL-Algorithmen auf die Verwendung von Gym ausgelegt [51, 93, 88]. Deshalb eignet sich OpenAI Gym sehr gut, um bekannte RL-Verfahren in neuen Umgebungen auszuprobieren.

Allgemein

Bei der Implementierung eines Gym-Environments müssen zunächst der Zustands- und Aktionsraum festgelegt werden. Der Zustandsraum ist dabei eine deterministische Beschreibung des Zustands und kann aus einzelnen Werten oder auch Bildern bestehen. Über festgelegte Grenzen können die Werte normalisiert werden, sodass alle Eingänge einen ähnlichen Einfluss auf die Ausgabe haben können. Der Aktionsraum bestimmt,d welche Aktionen der Agent bzw. die KI ausführen kann. Der Aktionsraum kann je nach Verfahren diskret oder auch kontinuierlich sein.
Für das erfolgreiche Training eines Agenten mit der eigenen Gym-Umgebung, müssen vier Funktionen geschrieben werden, welche eine Verbindung zwischen KI und der Umgebung herstellen [16]. Im Folgenden werden diese Funktionen vorgestellt.

- *step()* ist die Schnittstelle zwischen der KI und der Umgebung. Die Funktion führt die erhaltene Aktion des Agenten in der Umgebung aus und beobachtet die Zustandsänderung. Der resultierende Zustand und die passende Belohnung wird dem Agenten übergeben. Zudem wird geprüft, ob eine Abbruchbedingung, wie z. B. das Ablaufen der Zeit, erfüllt ist.

- *reward()* berechnet die Belohnung für die aktuelle Aktion des Agenten. Eine Belohnung soll dann ausgegeben werden, wenn die ausgeführte Aktion einen positiven Einfluss auf die Umgebung hat. Unerwünschtes Verhalten kann hingegen auch

bestraft werden. Diese Funktion wird in der Regel innerhalb der *step()*-Funktion aufgerufen.

- *reset()* setzt die Umgebung auf einen vordefinierten Anfangszustand zurück. Dieser kann je nach Umgebung auch zufällig ausgewählt werden, damit unterschiedliche Bereiche des Zustandsraum überprüft werden. Aufgerufen wird diese Funktion automatisch im Training, wenn das Ende einer Episode erreicht ist.
- *render()* ist für die grafische bzw. textliche Ausgabe des aktuellen Zustands verantwortlich. Diese Funktion wird nur bei Bedarf aufgerufen. Während des Trainings wird in der Regel keine Ausgabe gemacht, damit die Rechenzeit möglichst gering gehalten werden kann.

Mithilfe dieses Konzepts lassen sich die Umgebungen für die Simulation und für den Hardwaretest beschreiben.

Simulation

Zunächst muss die Umgebung für das Training mit der Simulation implementiert werden. Es gilt also die Spielfiguren durch die Aktionen eines Agenten zu steuern. Hierfür muss der Agent die Informationen zum Zustand des Tischkickers und die passende Belohnung erhalten.

Als erstes kann eine deterministische Beschreibung des Zustands bestimmt werden. Hierfür müssen alle wichtigen Parameter zusammengefasst werden. Dazu gehören die veränderlichen Parameter des Balls und der gesteuerten Stange. Damit der gesamte Zustand des Tischkickers aus einer Abfrage hervorgeht, müssen auch die Geschwindigkeiten der beweglichen Teile berücksichtigt werden. Zu den Parametern des hier festgelegten Zustandsraum gehören:

- x Position des Balls auf der Abszisse,
- y Position des Balls auf der Ordinate,
- ϕ Polarwinkel der Bewegungsrichtung des Balls,
- v Geschwindigkeit der Bewegung des Balls,
- y_{off} Verschiebung der KI gesteuerten Stange,

- α Rotation der KI gesteuerten Stange,
- v_{shift} Verschiebegeschwindigkeit der KI gesteuerten Stange,
- v_{rot} Rotationsgeschwindigkeit der KI gesteuerten Stange.

Dies sind die minimal erforderlichen Parameter, um alle Zustände eindeutig zu definieren. Hier fehlen die Angaben zu der vom Menschen gesteuerten Stange, da in dieser Arbeit keine Auswertung der Position dieser erfolgt. Des Weiteren werden viele Daten, wie die Position der Stange auf der Abszisse nicht berücksichtigt. Dies liegt daran, dass die fehlenden Parameter dem Tischkicker angepasst sind und sich nicht verändern. Ziel kann es in späteren Arbeiten sein, auch mit anpassbaren Parametern für z. B. die Größe des Tischkickers zu arbeiten, sodass ein entwickeltes Modell auch auf anderer Hardware zuverlässig arbeitet.
Die erforderlichen Parameter können mithilfe der *get_ state()*-Funktion des Tischkickers abgefragt werden.

Die Implementierung der *step()*-Funktion dient als Schnittstelle zwischen KI und Simulation. Der Ablauf der Funktion ist in Abbildung 6.49 dargestellt.

Die Funktion wird vom Agenten aufgerufen und erhält die ausgewählte Aktion als Argument. Je nach der Geschwindigkeit des Balls und der Rotation der Stangen wird der maximale Zeitschirtt in der Simulation berechnet, damit eine zuverlässige Kollisionsdetektion möglich ist. Anschließend kann die Position und Bewegung der KI-gesteuerten Stange und des Balls aktualisiert werden. Hierzu gehört auch die Kollisionsdetektion und die mögliche Korrektur der Position und Bewegung des Balls. Da es auf dem realen System eine begrenzte Abtastrate gibt, muss diese auch in der Simulation berücksichtigt werden. Deshalb werden die ersten Schritte wiederholt bis T_s abgelaufen ist. Der Wert hierfür ist an die Anforderung N8 auf $T_s = 0,05$ s angepasst.
Nach dem Ablaufen der Abtastzeit wird die Belohnung für die letzte Aktion berechnet und der aktuelle Zustand bestimmt. Diese Daten werden mit dem Abschluss der Funktion an den Agenten zurückgegeben.

Die Berechnung der Belohnung ist abhängig von dem Ziel des Agenten und wird deshalb in den nächsten Abschnitten an die verschiedenen Anwendungsfälle angepasst. Dies gilt auch für den Aktionsraum und den Anfangszustand der Umgebung. Die *render()*-Funktion kann hingegen stets einfach die grafischen Darstellung der Simulation aufrufen.

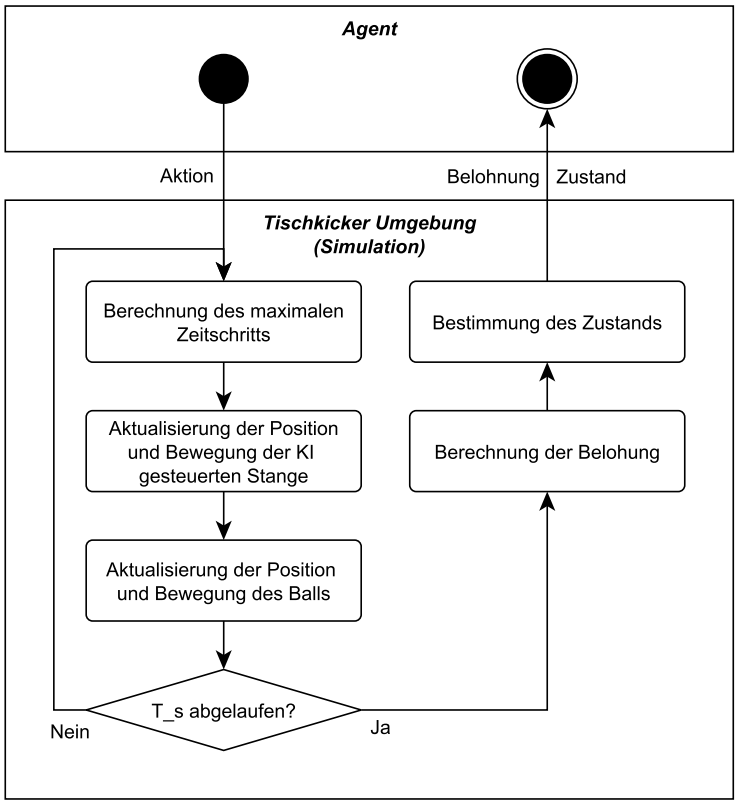


Abbildung 6.49: Ablauf der *step()*-Funktion der Umgebung in der Simulation

Hardware

Analog zu dem Gym-Environment der Simulation muss die Umgebung für den Hardwaretest entwickelt werden. Die Beschreibung des Zustands bleibt dabei gleich, da schließlich der Agent in der Umgebung für die Simulation trainiert wird. Dies gilt ebenfalls für den jeweiligen Aktionsraum.
Die den Zustand beschreibenden Parameter können hier jedoch nicht einfach mithilfe einer einfachen Funktion aus der Simulation ausgelesen werden. Stattdessen müssen sie über die in den vorigen Kapiteln entwickelte Sensordatenverarbeitung ermittelt werden. Abbildung 6.50 zeigt den Ablauf der *step()*-Funktion in der Umgebung auf der Hardware.

Anders als bei der Simulation muss auf der Hardware keine Berechnung der nächsten Position von Ball und Stange in einem iterativen Verfahren erfolgen. Stattdessen wird

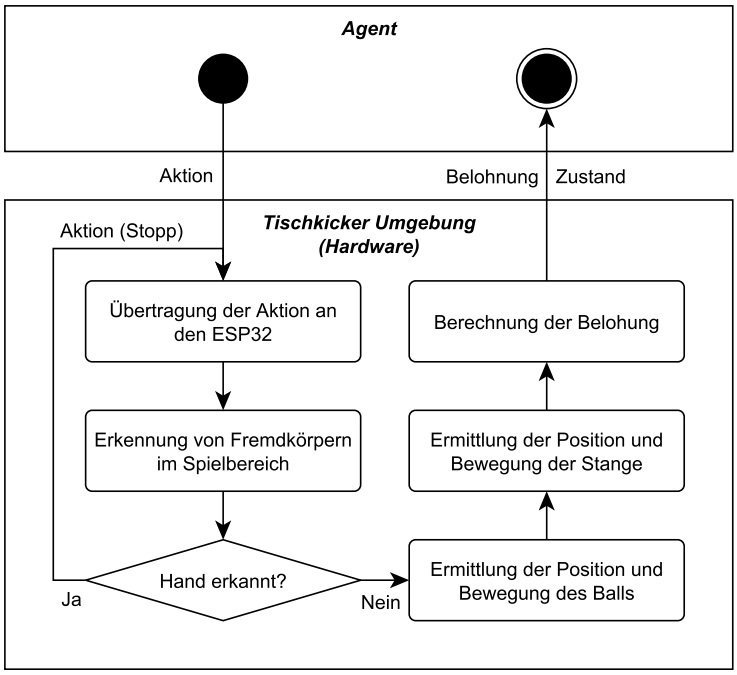


Abbildung 6.50: Ablauf der *step()*-Funktion der Umgebung auf der Hardware

die vom Agenten erhaltene Aktion direkt an den Mikrocontroller weitergegeben, welcher die gewünschte Bewegung umsetzt, indem er die Motoren entsprechend ansteuert.
Anschließend muss die Ermittlung des Zustands erfolgen. Hierbei wird mit der zuvor entwickelten 3D-Balldetektion gearbeitet. Die Position der Stange lässt sich über eine Umrechnung der Motorposition bestimmen. Bei fehlgeschlagenen Messungen kann wie zuvor beschrieben der letzte gültige Messpunkt verwendet werden.
Zusätzlich zu der Zustandsermittlung wird auch eine Erkennung von Fremdkörpern vorgenommen, damit die Motoren gestoppt werden können, wenn eine Hand im Spielbereich erkannt wird.

Die *render()*-Funktion wird in der Umgebung für die Hardware überflüssig, da das Spielgeschehen direkt auf dem Tischkicker beobachtet werden kann. Lediglich zur Überprüfung der korrekten Funktionalität der einzelnen Verarbeitungsschritte können in Anlehnung an die Visualisierung in der Simulation die ermittelten Daten in das verarbeitete Bild eingezeichnet werden (siehe Abbildung 6.51).

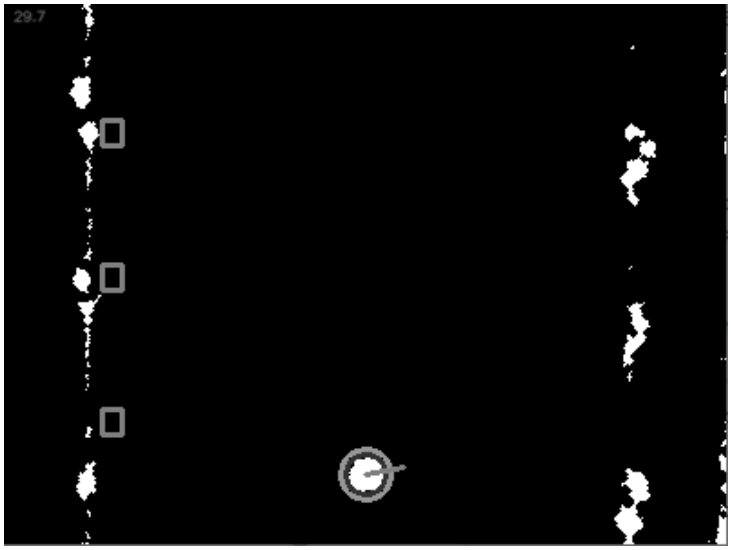


Abbildung 6.51: Visualisierung der *render()*-Funktion der Umgebung für die Ansteuerung der Hardware

6.5.2 Training

In diesem Kapitel werden ausgewählte Agenten in der Umgebung der Simulation trainiert. Hierbei werden verschiedene Einsatzfälle überprüft und so die Komplexität der Aufgaben sukzessiv erhöht. Angefangen wird mit dem Training eines Torwarts, der ausschließlich verteidigen soll. Das endgültige Ziel ist die Erstellung einer KI, die eine ganze Mannschaft eines großen Tischkickers steuern kann. In dieser Arbeit werden zunächst jedoch nur Ansätze für die Steuerung des Minikickers entwickelt, welche in zukünftigen Arbeiten dann erweitert werden können.

Tore verhindern

Der erste und vermutlich einfachste Einsatzfall ist das Verteidigen von Schüssen mithilfe einer Torwart-Reihe. Dies lässt sich auch sehr gut auf die Hardware übertragen, da die entsprechende Stange automatisiert wurde. Für das Verteidigen wird lediglich die Verschiebung der Stange benötigt. Dadurch wird keine Rotation gebraucht und das Training der KI dementsprechend vereinfacht.
Die Aufgabe entspricht so etwa dem Atari-Spiel Pong, welches in [63] mit einem DQN-Agenten gelöst wird, wobei sogar bessere Ergebnisse erzielt werden als bei der Steuerung durch einen Menschen. Hierbei wurde mit Bild- statt Metadaten gearbeitet, wodurch sich

ein guter Vergleich der beiden Trainingsdaten ergibt. Deshalb wird zunächst ein DQN-Agent verwendet.

Der DQN-Agent hat lediglich einen diskreten Aktionsraum, da die Neuronen am Ausgang des Netzes One Hot Encoded werden. Deshalb kann die Geschwindigkeit der Motoren nicht „stufenlos“ eingestellt werden. Stattdessen werden nur drei Ausgaben ermöglicht, welche der maximalen Geschwindigkeit der Motoren $v_{m,max}$ in beide Richtungen sowie dem Stillstand entsprechen. Die Aktionen und die jeweilig dazugehörige einheitenlose Zielgeschwindigkeit des Motors für die Verschiebung v_v, welche an die Motorsteuerung weitergeleitet wird, sind in Tabelle 6.6 zusammengefasst. Es wäre möglich noch höhere Geschwindigkeiten zu erlauben, jedoch kann durch die hohe Frequentierung der KI und die begrenzte Beschleunigung der Motoren eine geringere Geschwindigkeit erzeugt werden. Hierfür muss der Agent nur die verschiedenen Aktionen im Wechsel aufrufen. Gerade für diesen ersten Anwendungsfall ist ein solcher Aktionsraum ausreichend.

Tabelle 6.6: Aktionsraum des DQN-Agenten für die Steuerung der Verschiebung

Aktion	v_v
0	-70
1	0
2	70

Neben dem Aktionsraum muss auch der Startzustand über die *reset()*-Funktion festgelegt werden. Hierbei wird der Tischkicker samt Ball und Stangen initialisiert. Da auf der Hardware nur die Position der eigenen Reihe ermittelt werden kann, wird auch nur diese dem Spielfeld hinzugefügt. Die Größen der einzelnen Objekte sind dem vorhandenen Minikicker angepasst.
Die Position des Balls und die Verschiebung sowie Rotation der Stange werden über einen zufälligen Wert eingestellt, sodass die KI später möglichst gut generalisieren kann. Dadurch soll der Agent in allen möglichen Situationen eine passende Aktion finden.
Damit das Verteidigen überhaupt erst möglich wird, muss dem Ball ein Impuls in Richtung Tor gegeben werden. Dies kann direkt bei der Initialisierung erfolgen, wobei einerseits die Geschwindigkeit und andererseits die Bewegungsrichtung in Form des Polarwinkels angegeben werden. Diese Werte werden ebenfalls zufällig ausgewählt. In Abbildung 6.52 ist die mögliche Position und Bewegungsrichtung des Balls zu Beginn einer Episode zu sehen. In orange ist der Bereich gekennzeichnet, in welchem der Ball liegen kann, während in grün die mögliche Bewegungsrichtung aus der aktuellen Position markiert ist. Die Geschwindigkeit des Balls wird zwischen 0,5 $\frac{\mathrm{m}}{\mathrm{s}}$ und 2 $\frac{\mathrm{m}}{\mathrm{s}}$ festgelegt.

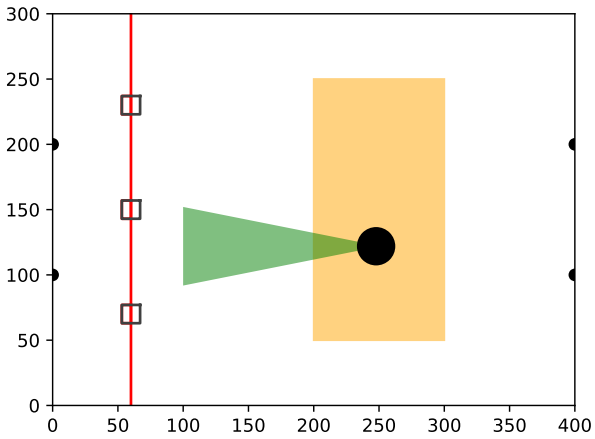


Abbildung 6.52: Mögliche Position und Bewegungsrichtung des Balls im Anfangszustand beim Training der Verteidigung

Die Belohnung des Agenten muss gut durchdacht und ausgewogen sein, da sie die Grundlage für das Training darstellt. Die Qualität des Agenten ist limitiert durch die Qualität der Belohnungsfunktion.
Das Hauptziel der Verteidigung ist, kein Tor zu fangen. Wenn der Ball doch in das eigene Tor geschossen wird, muss das Verhalten der KI bestraft werden, weshalb in diesem Fall die Belohnung auf $r = -300$ gesetzt wird. Zwar ist das Tore Schießen in diesem Anwendungsfall kein primäres Ziel, aber wenn der Ball zufällig ins gegnerische Tor trifft, kann trotzdem eine hohe Belohnung von $r = 800$ ausgeschrieben werden.
Neben dem Wert für die Tore können auch kleine Belohnungen, bei einem Verhalten, dass einen positiven Einfluss auf das Endergebnis hat, addiert werden. Dadurch lässt sich das Training beschleunigen, da z. B. nicht in jeder Episode ein Tor fällt. Ein positives Ereignis beim Verteidigen ist, wenn der Ball seine Richtung verändert, nachdem er eine Figur berührt. Deshalb wird der skalierte Unterschied der Bewegungsgeschwindigkeit auf der Abszisse

$$\Delta v_x \;=\; v_{x,end} - v_{x,start} \tag{6.53}$$

der Belohnung hinzugefügt. So wird bei einem frontalen Zusammenstoß eine hohe und bei einem seitlichen Abpraller eine niedrige Belohnung addiert. Wenn der Ball in die Richtung des eigenen Tors abgelenkt wird, erhält der Agent eine Bestrafung.
Noch kurzfristiger kann die Verschiebung der Figuren honoriert werden, indem die maxi-

male Distanz des Balls auf der Ordinate zu einer auf der Stange befindlichen Spielfiguren gemessen wird. Wenn diese Distanz nach der Aktion geringer ist als vor der Ausführung, befindet sich eine der Figuren näher an der Position des Balls auf der Ordinate. So wird das „Verfolgen“ der Stange positiv belohnt. Sollte sich die Distanz erhöhen, wird dies bestraft. Dadurch muss die KI eine Vorhersage über die Bewegung des Balls machen und vorsorglich eine Korrektur der Verschiebung durchführen.

Damit das Training effizient gestaltet werden kann, muss eine Abbruchbedingung für die Episoden eingerichtet werden. Ansonsten kann der Ball beispielsweise durch die Reibungsverluste zum Stehen kommen, sodass kein sinnvolles Training mehr möglich ist. Die Abbruchbedingung sind hier erreicht, wenn entweder eine Sekunde simuliert wurde oder ein Tor gefallen ist.

Tabelle 6.7: Aufbau des ersten Netzes des DQN-Agenten für das Verteidigen von Schüssen

Schicht	Neuronen	Art	Aktivierungsfunktion
Eingangsschicht	8	Flatten	-
1	32	Dense	ReLU
Ausgangsschicht	3	Dense	Linear

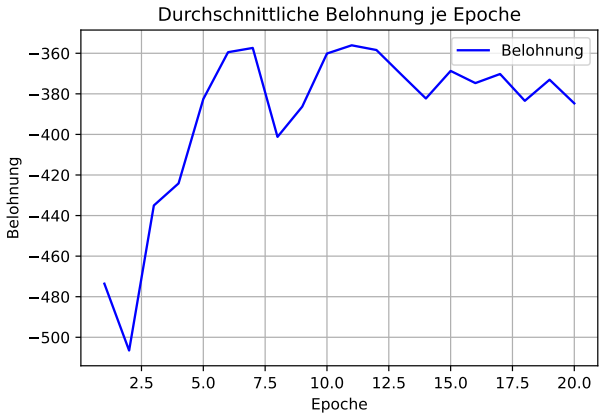


Abbildung 6.53: Training des DQN-Agenten für das Verteidigen mit dem ersten neuronalen Netz

Für den DQN-Agenten muss ein neuronales Netz festgelegt werden. Die Eingangsschicht entspricht dabei dem Zustand, während die Ausgangsschicht die Aktionen widerspiegelt. Da beide Schichten sehr schlicht sind und die Aufgabe keine hohe Komplexität hat, wird ein 32 Neuronen großes Dense Layer dazwischen eingesetzt. Diese Schicht wird mit der

ReLU-Aktivierungsfunktion versehen, da diese in der Regel gute Ergebnisse liefert und schnell berechnet werden kann. Das gesamte Netzwerk ist in Tabelle 6.7 zu sehen. Als beste Lernrate für das Netz wird in mehreren Versuchen $\eta = 5 \cdot 10^{-4}$ ermittelt.

Das Training des Netzes dauert auf einem System mit Intel Core i7-6700K 4 GHZ Prozessor und 16 GB Arbeitsspeicher bei 200.000 ausgeführten Schritten rund 20 Minuten. Die durchschnittliche Belohnung kann in Abbildung 6.53 beobachtet werden.

Es zeigt sich, dass das gewählte Netz keine ausreichende Komplexität besitzt. Zwar versucht der Agent die Stange in die Richtung des Balls zu bewegen, allerdings gelingt dies nur schlecht. Die Stangen werden stets nur in eine Richtung bewegt, eine Korrektur erfolgt z. B. bei einem Zusammenstoß mit der Wand nicht. Deshalb muss die Komplexität des Netzes erhöht werden.

Tabelle 6.8: Aufbau des zweiten Netzes des DQN-Agenten für das Verteidigen von Schüssen

Schicht	Neuronen	Art	Aktivierungsfunktion
Eingangsschicht	8	Flatten	-
1	32	Dense	ReLU
2	32	Dense	ReLU
Ausgangsschicht	3	Dense	Linear

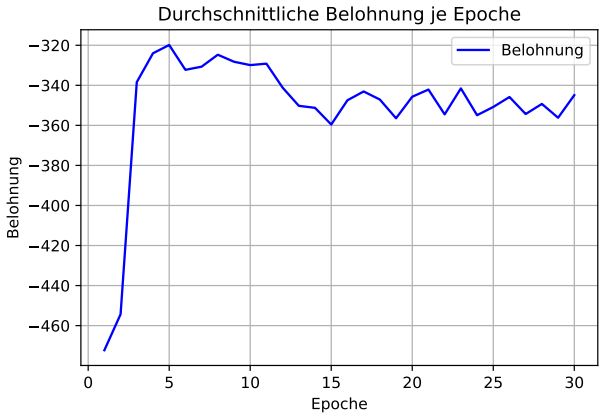


Abbildung 6.54: Training des DQN-Agenten für das Verteidigen mit dem zweiten neuronalen Netz

Im nächsten Schritt wird ein zweites Dense Layer hinzugefügt, sodass das Netz in Tabelle 6.8 entsteht. Auch hier ist die beste Lernrate bei $\eta = 5 \cdot 10^{-4}$.

Das zweite Netz hat trotz erhöhter Anzahl an Parametern etwa 35 Minuten Training für 300.000 Schritte benötigt. Die durchschnittliche Belohnung des Agenten kann in Abbildung 6.54 beobachtet werden.

Das zweite Netz ist minimal besser als das erste. Die Ergebnisse sind jedoch nicht zufriedenstellend, da der Ball häufiger ins eigene Tor geht. Aus diesem Grund wird die Komplexität erneut erhöht. Es ergibt sich das Netz in Tabelle 6.9.

Tabelle 6.9: Aufbau des dritten Netzes des DQN-Agenten für das Verteidigen von Schüssen

Schicht	Neuronen	Art	Aktivierungsfunktion
Eingangsschicht	8	Flatten	-
1	32	Dense	ReLU
2	64	Dense	ReLU
3	16	Dense	ReLU
Ausgangsschicht	3	Dense	Linear

Durch die größere Komplexität des dritten Netzes ist die optimale Lernrate eher bei $\eta = 1 \cdot 10^{-3}$. Dieses Mal wird für 400.000 Schritte trainiert, da selbst dann sich die Belohnung noch verbessert. Dies dauert etwa 50 Minuten. Die durchschnittliche Belohnung des Agenten kann in Abbildung 6.55 beobachtet werden.

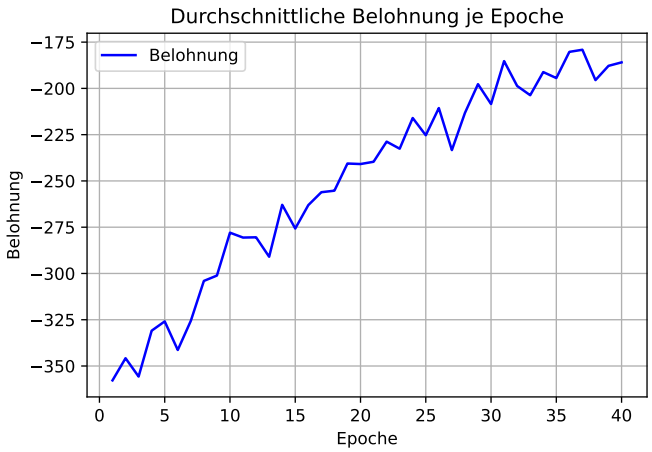


Abbildung 6.55: Training des DQN-Agenten für das Verteidigen mit dem dritten neuronalen Netz

Mit dem dritten Netz kann ein Agent trainiert werden, der agil auf den Ball reagiert und die meiste Zeit den direkten Weg zum Tor versperrt. Bei einer visuellen Überprüfung

kann festgestellt werden, dass nur in seltenen Ausnahmen das Verteidigen nicht gelingt. Die noch sehr niedrige durchschnittliche absolute Belohnung kommt durch das Belohnungssystem und die Gewichtung der einzelnen Belohnungen zustande und hat keine weitere Aussagekraft.
Da beim Training auch in den letzten Schritten noch die durchschnittliche Belohnung erhöht wird, ist davon auszugehen, dass der Agent noch weiter verbessert werden könnte, allerdings ist die entwickelte KI ausreichend genau für diesen ersten Anwendungsfall. Dazu kommt, dass die Verbesserungen in der Belohnung nicht zwingend eine Verbesserung des gewünschten Verhaltens nach sich ziehen, da z. B. nach dem Abwehren eines Balls eine Bestrafung ausgesprochen werden kann, obwohl das Verhalten in dem Moment nicht relevant ist.

Manuelle Programmierung

Interessant für die Bewertung der KI ist auch der Vergleich mit anderen Methoden. In diesem Fall kann ein manuell programmierter Algorithmus zum Abwehren der Bälle genutzt werden, um die möglichen Vor- bzw. Nachteile der KI aufzuzeigen.

Die Implementierung eines solchen Algorithmus kann mithilfe der beiden Umgebungen für das Training in der Simulation und für die Tests auf der Hardware erfolgen. Der Algorithmus agiert hierbei als Agent, der auf den Zustand reagiert und eine Aktion ausgibt. Er beachtet jedoch nicht die Belohnung, da während der Laufzeit keine Anpassungen des Algorithmus gemacht werden.
Die Auswahl der Aktion erfolgt anhand der Position des Balls auf der Abszisse und der Verschiebung der Stange. Dabei wird nur die Differenz zwischen der mittleren Figur und dem Ball berücksichtigt, da durch die begrenzte Verfahrstrecke der Stange ein Wechsel der verantwortlichen Figur nötig ist, um platzierte Bälle von der Außenbahn abzuwehren. Mit der mittleren Figur lassen sich hingegen die meisten Schüsse abwehren. Hierfür wird je nach Differenz

$$\Delta y \;=\; y_{ball} - y_{figure} \tag{6.54}$$

eine passende Aktion ausgewählt. Dies wird über eine Fallunterscheidung durchgeführt, welche in Tabelle 6.10 dargestellt ist. Hierbei wird die Geschwindigkeit der Verschiebung reduziert, wenn die Differenz kleiner ist, damit ein Überschwingen des Systems vermieden werden kann.

Tabelle 6.10: Auswahl der Aktion zum Verteidigen von Schüssen mittels eines manuell programmierten Algorithmus

Δy in mm	Aktion
$30 \leq$	70
$10 \leq \Delta y < 30$	30
$3 \leq \Delta y < 10$	5
$-3 < \Delta y < 3$	0
$-10 < \Delta y \leq -3$	-5
$-30 < \Delta y \leq -10$	-30
≤ -30	-70

Der gewählte Ansatz ist natürlich nur ein primitiver Algorithmus, der durch verschiedene Erweiterungen verbessert werden kann. Es ist allerdings ein guter Vergleich zum Agenten, da er etwa nach einem ähnlichen Prinzip wie die Belohnungsfunktion des DQN-Agenten funktioniert.
Der Vergleich der zwei Ansätze erfolgt im nächsten Kapitel.

Tore schießen

Nachdem ein erster einfacher Agent erfolgreich trainiert wurde, kann die Komplexität der Aufgabe gesteigert werden. Dies bedeutet, dass auch der Motor für die Rotation von der KI gesteuert wird, wodurch sich der Aktionsraum vergrößert. Statt wie zuvor nur drei Aktionen, müssen zwei mal drei Aktionen unabhängig voneinander aufgerufen werden. Hierfür kann in Gym ein multidiskreter Aktionsraum festgelegt werden, sodass mit sechs Neuronen in der Ausgangsschicht zwei Aktionen ausgegeben werden [71]. Der DQN-Agent unterstützt diese Funktion jedoch nicht [51]. Hier kann stets nur eine Aktion an die Umgebung weitergegeben werden. Aufgrund der noch recht geringen Komplexität kann dieses Problem umgangen werden, indem jeder Kombination der gewünschten Bewegungen ein Neuron am Ausgang des Netzes des Agenten entspricht. Die möglichen Kombinationen sind in der Tabelle 6.11 dargestellt. Hierbei entspricht v_v der Zielgeschwindigkeit der Verschiebung, während v_r für die Rotation verantwortlich ist. Die jeweiligen Werte werden an die Motorsteuerung übertragen.

Konkret soll in diesem Abschnitt ein Agent entwickelt werden, der einen vor der Stange ruhenden Ball ins gegnerische Tor schießen kann. Damit die Aufgabe nicht zu einfach ist, wird der Ball an einen zufälligen Platz auf der Ordinate gelegt und auch die Verschiebung

Tabelle 6.11: Aktionsraum des DQN-Agenten für die Steuerung der Verschiebung und Rotation

Aktion	v_v	v_r
0	-70	-70
1	0	-70
2	70	-70
3	-70	0
4	0	0
5	70	0
6	-70	70
7	0	70
8	70	70

und Rotation der Stange zufällig festgelegt. Der Bereich, in dem der Ball hingelegt wird, ist in Abbildung 6.56 markiert.
Anstatt direkt einen Agenten zu trainieren, der auch Schüsse auf das eigene Tor abwehren kann, wird nur das Schießen eines ruhenden Balls gefordert, da so die grundlegenden Funktionen (Schießen und Abwehren) einzeln überprüft werden können. Nach dem Bestätigen dieser Funktionen können auch flexiblere und komplexere Modelle auf Basis des hier ermittelten Wissens trainiert werden.

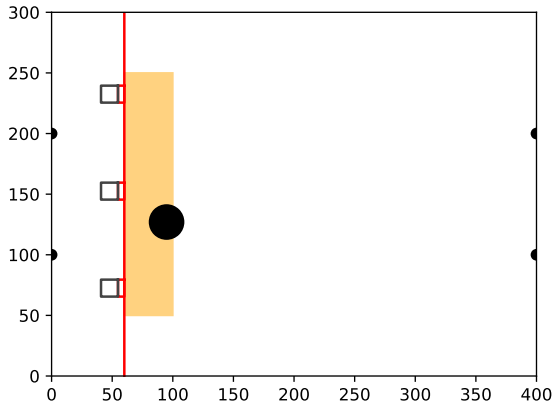


Abbildung 6.56: Mögliche Position des Balls im Anfangszustand beim Training des Angriffs

Die Belohnungen müssen für diesen Anwendungsfall ebenfalls angepasst werden. Dabei kann jedoch die Zuweisung für geschossene und gefangene Tore gleich bleiben. Dies gilt

auch für die Belohnung bei der Beschleunigung des Balls in Richtung des gegnerischen Tors.
Der erste Unterschied zu der Funktion für das Verteidigen ist eine zusätzliche Belohnung für die Reduzierung der Distanz zwischen einer beliebigen Figur und dem Ball auf der Ordinate. Damit der Agent möglichst schnell den Ball findet, kann diese Belohnung zunächst beibehalten werden. Wenn der Ball dann in Bewegung versetzt wird, muss diese Belohnung ausgesetzt werden, da z. B. schräge Schüsse, die aus seitlichen Positionen benötigt werden um das Tor zu treffen, bestraft werden.
Zusätzlich kann noch eine Belohnung eingeführt werden, die sich an der Position des Balls auf der Abszisse orientiert. Je näher am gegnerischen Tor, desto höher die Belohnung.
Zuletzt muss noch eine kleine Bestrafung von $r = -3$ standardmäßig addiert werden, damit der Agent gezwungen ist, neue Aktionen auszuprobieren. Ansonsten kann es passieren, dass der Agent den Ball nicht schießt, da z. B. durch die Verschiebung bereits eine Belohnung ausgesprochen wurde.

Das Training des DQN-Agenten für das Schießen der Tore kann zunächst mit dem dritten vortrainierten Netz aus dem ersten Anwendungsfall durchgeführt werden. Denn das Netz kann bereits zuverlässig eine Figur zum Ball verschieben, sodass nur noch der Schuss durchgeführt werden muss. In Abbildung 6.57 ist das Ergebnis des Trainings zu sehen. Dieses dauert etwa 75 Minuten für 600.000 Schritte.

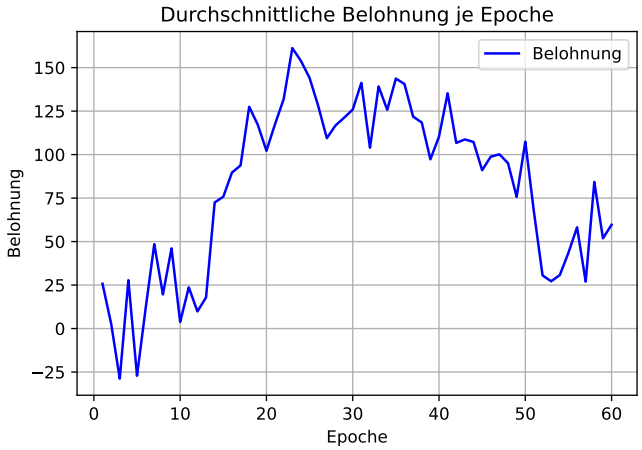


Abbildung 6.57: Training des DQN-Agenten für das Schießen mit dem Netzwerk fürs Verteidigen

Es zeigt sich, dass der Agent es durchaus schafft den Ball in die Richtung des gegnerischen Tors zu schießen. Die durchschnittliche Belohnung sinkt jedoch nach einiger Zeit wieder, was durch das Vergessen früherer eintrainierter Daten hervorgerufen wird [86].
Der Ansatz mittels DQN-Agent ist also auch für mehrere Aktionen möglich, allerdings ist das Ergebnis noch nicht zufriedenstellend. Deshalb ist davon auszugehen, dass das Netz für die neue Aufgabe eine zu geringe Komplexität bietet, sodass im nächsten Schritt ein etwas komplexeres Netz implementiert wird. Dieses zweites Netz für das Schießen des Balls wird in Tabelle 6.12 beschrieben.

Tabelle 6.12: Aufbau des zweiten Netzes des DQN-Agenten für das Schießen von Bällen

Schicht	Neuronen	Art	Aktivierungsfunktion
Eingangsschicht	8	Flatten	-
1	32	Dense	ReLU
2	128	Dense	ReLU
2	64	Dense	ReLU
3	16	Dense	ReLU
Ausgangsschicht	3	Dense	Linear

Das gewählte Netzwerk wurde durch mehrere Anpassungen für die Aufgabe optimiert. Als optimale Lernrate wird $\eta = 1 \cdot 10^{-4}$ ermittelt. Das Training des Netzes dauert für 700.000 Schritte rund 85 Minuten und kann in Abbildung 6.58 gesehen werden.

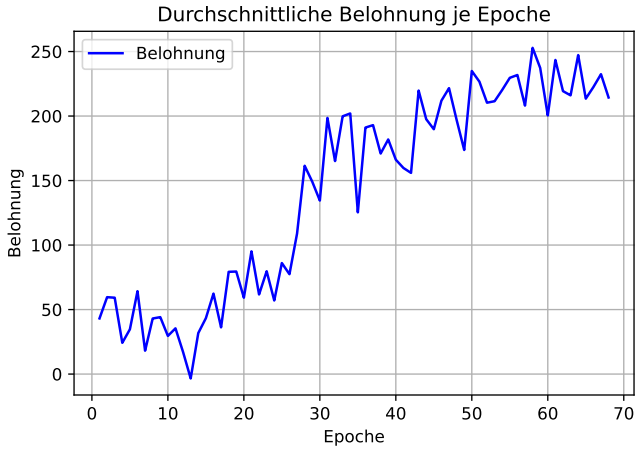


Abbildung 6.58: Training des DQN-Agenten für das Schießen mit einem angepassten Netzwerk

Das trainierte Netz kann Bälle teilweise in das gegnerische Tor befördern. Wenn der Ball mittig liegt, kann die KI regelmäßig Tore schießen. In anderen Positionen wird diese

Genauigkeit nicht erreicht. Allerdings wird die erhoffte Erfolgsrate nicht erreicht. Grund hierfür kann die limitierte Steuerung der Stangen sein, die hier nur mit einer sehr hohen Geschwindigkeit erfolgt. Wahrscheinlich ist für die Lösung dieser Aufgabe ein anderer Agent, der mehrere kontinuierliche Ausgaben hat, sinnvoller. Dies wird in dieser Arbeit aus zeitlichen Gründen jedoch nicht weiter untersucht. Stattdessen wir noch ein letzter Anwendungsfall besprochen, der die ersten zwei Einsatzfälle kombiniert.

Kombination

Der letzte Anwendungsfall, welcher auf der aktuellen Hardware eingesetzt werden kann, ist die Kombination der ersten beiden Anwendungsfälle. Der RL-Agent soll in diesem Fall Schüsse abwehren und idealerweise direkt ins gegnerische Tor schießen können. So kann am ehesten ein Spiel eines Menschen mit der KI auf dem Tischkicker ermöglicht werden. Dieser Einsatzfall ist besonders dem Schießen sehr ähnlich, da erst eine Figur in die Richtung des Balls verschoben werden muss, bevor dieser geschossen werden kann. Grundsätzlich kann deshalb in diesem Fall wie bei den vorigen Fällen vorgegangen werden. Der Aktionsraum ist identisch mit dem aus Tabelle 6.13. Dies gilt auch für den Startzustand, wobei dieser vom ersten Fall übernommen wird, damit der Agent auch einen Ball abwehren kann.
Die Belohnungen des Netzes fürs Schießen müssen nur geringfügig angepasst werden, sodass stets eine Belohnung bzw. eine Bestrafung ausgesprochen wird, wenn sich eine Figur auf der Ordinate der Position des Balls annähert.
Für das Training wird die beste ermittelte Struktur aus dem letzten Einsatzfall übernommen. Diese ist in Tabelle 6.13 nochmal aufgelistet.

Tabelle 6.13: Aufbau Netzes des DQN-Agenten für das Verteidigen und Schießen von Bällen

Schicht	Neuronen	Art	Aktivierungsfunktion
Eingangsschicht	8	Flatten	-
1	32	Dense	ReLU
2	128	Dense	ReLU
2	64	Dense	ReLU
3	16	Dense	ReLU
Ausgangsschicht	3	Dense	Linear

Das Training des Agenten wird mit einer Lernrate von $\eta = 5 \cdot 10^{-4}$ für 800.000 Schritte durchgeführt. Dies benötigt rund 90 Minuten, wobei der Trainingsfortschritt in Abbildung 6.59 gezeigt wird.

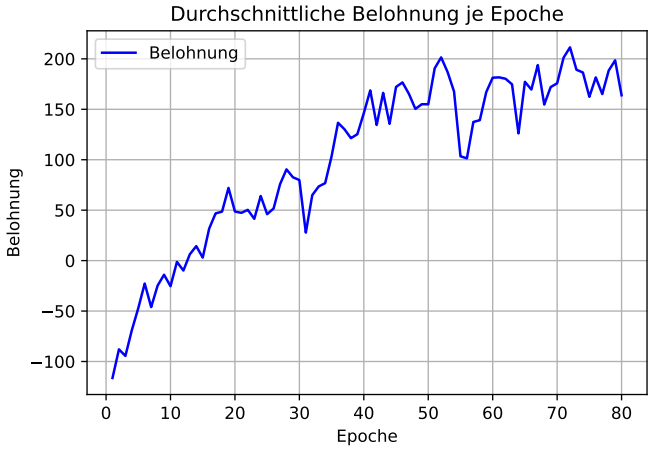


Abbildung 6.59: Training des DQN-Agenten für das Verteidigen und Schießen eines Balls

Auch hier sind die Ergebnisse nicht ganz zufriedenstellend. Allerdings ist es der KI möglich die meisten Bälle abzuwehren und häufiger auch ein Tor zu schießen.

Vollständiger Tischkicker

Für einen späteren kompletten Aufbau müssen Agenten entwickelt werden, die mehrere Stangen steuern können und dabei gute Ergebnisse erreichen. Da in den letzten Abschnitten bereits Grenzen des DQN Agenten aufgezeigt wurden, gilt es noch weitere Methoden zu überprüfen und miteinander zu vergleichen. Im Rahmen dieser Arbeit ist dies jedoch nicht mehr möglich.

7 Evaluierung

Nachdem im letzten Kapitel die Entwicklung des automatisierten KI Tischkickers beschrieben wurde, sollen hier die Ergebnisse der einzelnen Verarbeitungsschritte evaluiert werden. Dabei wird die Qualität der Lösung untersucht und bewertet, sodass ein Ausblick gegeben werden kann, an welchen Stellen noch Verbesserungspotenziale sind. Dies geschieht auch unter der Berücksichtigung der aufgestellten Anforderungen.

7.1 Aufbau der Hardware

Zunächst wird der Aufbau der Hardware untersucht, welcher in dieser Arbeit lediglich ein Provisorium darstellt.

7.1.1 Allgemein

Einige funktionale Anforderungen an den Aufbau konnten im Rahmen dieser Arbeit nicht erfüllt werden. So ist weder eine Spielstandsanzeige, noch eine Ballmaschine Teil des ersten Aufbaus. Auch ein Mechanismus zum Bewegen eines unerreichbaren Balls wurde nicht implementiert. Die genannten Funktionen sind jedoch nicht zwingend für den Erfolg der Arbeit erforderlich und deshalb im ersten Aufbau entbehrlich.
Die Montage des Kickers auf einer Holzplatte ermöglicht den leichten Transport und Aufbau des Systems und eignet sich daher gut als erste Konstruktion. Durch die Verschraubung sind jedoch kleine Anpassungen des Aufbaus nicht ganz einfach, weshalb für weitere Anlagen ein Aufbau mit Profilschienen empfohlen wird. So kann auch die Linearführung des Schlittens mit dem Motor für die Rotation tiefer gelegt werden, wodurch die Befestigung der Stange an der Motorhalterung mehr Platz bekommt. Wichtig ist jedoch auch die Steifigkeit der Anlage, die bei der Holzkonstruktion über mehrere Verschraubungen erreicht wird. Ohne die nötige Stabilität kann sich der Aufbau durch die hohe Kraft der Motoren selbst zerstören.

Der hier realisierte Aufbau ist noch kein alleinstehendes System. Hierfür wird unter anderem eine mobile Plattform benötigt, um die Bildverarbeitung und Kommunikation mit der KI zu übernehmen. Derzeit muss noch ein Anschluss der Kamera und des Mikrocontrollers an einen externen Rechner vorgenommen werden, um das System zu betreiben. Gleichzeitig wird ein externes Netzteil für die Versorgung der Komponenten verwendet, welches in einem späteren Aufbau integriert sein soll, damit lediglich ein Netzstecker benötigt wird.
Grundsätzlich funktionieren die ausgewählten Komponenten erwartungsgemäß. Ausgeschlossen hiervon ist natürlich die ursprünglich ausgewählte Linearführung, welche entgegen der Erwartungen nicht den Anforderungen entspricht. Beim Aufbau weiterer Reihen sollte zudem eine etwas breitere Energiekette gewählt werden, damit die Kabel mehr Platz haben.

7.1.2 Aufstellung der Kosten

Tabelle 7.1: Auflistung und Kosten der Hardwarekomponenten

Komponente	Anzahl	Gesamtkosten
Minikicker	1	0 €
Kamera	1	0 €
Schrittmotoren	2	220 €
Linearführung 1	1	87 €
Linearführung 2	1	60 €
Energiekette	1	10 €
Flanschlager	1	4 €
Motorhalterungen	4	0 €
Zahnriemen	3	20 €
Spielfigurenstange	1	< 5 €
Mikrocontroller	1	6 €
Transistoren	6	< 1 €
Kabel	3	0 €
Holzplatte	1	0 €
	Summe	413 €

Für die Konstruktion des Tischkickers wurden verschiedene Komponenten beschafft. Um die Einhaltung des Budgets prüfen zu können, wird in Tabelle 7.1 eine Auflistung der Komponenten und ihrer Kosten vorgenommen. Hierbei sind die Preise der einzelnen Komponenten gerundet, da die genauen Werte aufgrund gebündelten Versands nicht

ermittelbar sind. Weiter sind auch viele kostenlose Komponenten dabei, da sie aus vorigen Projekten zur Verfügung standen oder aus Resten bzw. Abfällen bestehen. Das Netzteil wird hier nicht mit aufgelistet, da es nicht direkt zum System des Minikickers zählt.

7.2 Motorsteuerung

Die Motorsteuerung ist eine wichtige Verbindung zwischen KI und Tischkicker. Ihre Funktionalität ist essentiell, um die gewünschten Bewegungen des Balls auszuführen.
Die zwei Bewegungsrichtungen der Stangen müssen gleichzeitig und unabhängig voneinander steuerbar sein, was durch den Aufbau mittels Linearführung möglich ist. Auch softwareseitig ist es möglich verschiedene Befehle an die Steuerung zu übergeben, welche dann gleichzeitig ausgeführt werden können. Die automatisierte Stange lässt sich so entsprechend der Anforderungen bewegen und stoppen.

7.2.1 Verschiebung

Neben der grundsätzlichen Funktionalität ist auch die Qualität der Motorsteuerung zu untersuchen. Einerseits ist eine gewisse Genauigkeit erforderlich, während andererseits auch die Geschwindigkeit der Bewegungen interessant ist.
Die Genauigkeit der Motoren geht vorrangig aus der Schrittweite hervor. Diese ist bei beiden Motoren auf 6400 Schritte pro Umdrehung festgelegt.
Bei der Rotation lässt sich dieser Wert fast direkt übertragen, wobei hier durch die zwei verwendeten Zahnräder ein Übertragungsverhältnis von 1:2 entsteht. Dadurch entsprechen 3200 Schritten einer Umdrehung der Stangen, sodass die Position in einem Bereich von

$$\Delta\alpha_r \;=\; \frac{360^\circ}{3200} \;=\; 0,1125^\circ \tag{7.1}$$

vergeben werden kann.
Die Verschiebung ist neben der Schrittweite auch vom Durchmesser des Zahnrads, mit welchem der Riemen bewegt wird, abhängig. Dieser liegt bei $d = 25$ mm, wodurch eine ganze Umdrehung des Motors eine Verschiebung um $s = d \cdot \pi = 78,5$ mm bedeutet.

Hieraus ergibt sich eine Genauigkeit von

$$\Delta s_v \;=\; \frac{78,5 \text{ mm}}{6400} \;=\; 12,27\ \mu\text{m}. \tag{7.2}$$

Die beiden Genauigkeiten gelten dabei sowohl für die Positionierung als auch für die Ermittlung der Verschiebung und Rotation der Stange.
Die erreichten Werte sind sehr klein und dementsprechend ist die Genauigkeit hoch. Es ist allerdings zu beachten, dass stets ein kleiner Offset zu den Werten addiert werden muss, da die Startposition vom Benutzer eingestellt wird und hierbei kleine Fehler entstehen.

Die Geschwindigkeit der Bewegungen sind mindestens ebenso wichtig wie die Genauigkeit. Die maximale Geschwindigkeit der Verschiebung lässt sich über die Formel 6.1 ausrechnen. Hierbei wird die maximal einstellbare Geschwindigkeit $v_{v,max} = 70$ eingesetzt, sodass mit einer maximalen Frequenz von

$$f_{v,max} \;=\; \frac{1}{T_{v,max}} \;=\; 2 \cdot \frac{(w_{100} - w_0) \cdot |v_{v,max}|}{100} \cdot 10^{12} \;=\; 87,15 \text{ kHz} \tag{7.3}$$

Schritte ausgeführt werden. Hieraus kann die maximale Geschwindigkeit der Verschiebung auf der Hardware

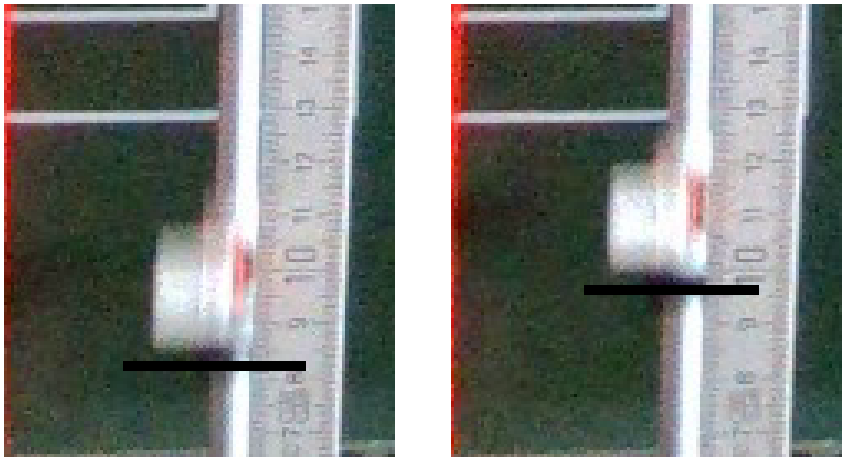
$$v_{v,hw} \;=\; f_{v,max} \cdot \Delta s_v \;=\; 1,07\ \frac{\text{m}}{\text{s}} \tag{7.4}$$

berechnet werden. Da die Geschwindigkeit in der Realität durch die Berechnungszeit auf der Hardware etwas reduziert wird, wird mittels eines Tests in Abbildung 7.1 die tatsächliche maximale Geschwindigkeit ermittelt.

Die abgelesenen Werte auf dem Maßband neben dem Ball betragen $s_{start} = 8,3$ cm und $s_{end} = 9,7$ cm. Zwischen den Bildern ist ein Periode von $T = \frac{1}{30}$ s. Die Geschwindigkeit der Verschiebung liegt also tatsächlich bei

$$v_{Stange} \;=\; \frac{s_{start} - s_{end}}{T} \;=\; 0,42\ \frac{\text{m}}{\text{s}}. \tag{7.5}$$

Diese ist kleiner als in den Anforderungen angegeben, was jedoch für den Minikicker noch ausreichend ist. Eine höhere Geschwindigkeit wäre über die Reduzierung der Schritte für eine Umdrehung durchaus möglich, jedoch kommt es dann beim Motor durch die zu hohe Belastung zu Schrittverlusten, die nicht mehr ausgeglichen werden können. Durch



(a) Startposition (b) Endposition

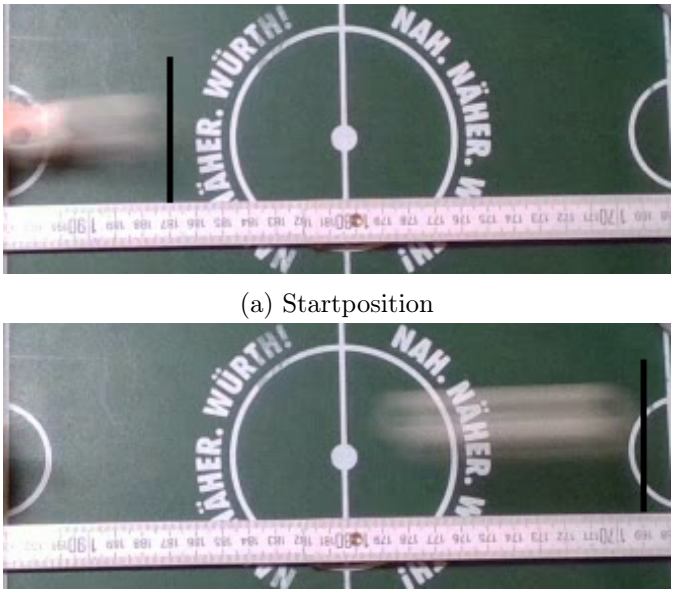
Abbildung 7.1: Messung der Position der Spielfiguren zur Bestimmung der Geschwindigkeit der Verschiebung

den Einsatz eines kräftigeren Modells könnte dem entgegengewirkt werden, sodass die gewünschte Geschwindigkeit erreichbar wird. Hierfür muss jedoch auch die Stabilität des Tischkickers angepasst werden, da sehr hohe Kräfte wirken.
Neben der Geschwindigkeit ist auch die Beschleunigung der Verschiebung sehr wichtig. Deshalb muss die Zeit gemessen werden, innerhalb welcher die Beschleunigung auf die hier maximal erreichbare Geschwindigkeit möglich ist.
Vom Stillstand bis zum Erreichen der maximalen Geschwindigkeit wird eine Zeit von $T_a = \frac{3}{30}$ s $= 0,1$ s benötigt, was sehr gut mit den Angaben in Abbildung 6.9 zusammenpasst.

7.2.2 Rotation

Für die Rotation gibt es keine direkte Vorgabe an die Geschwindigkeit. Stattdessen soll der Ball möglichst schnell geschossen werden. Dies lässt sich nicht auf die gleiche Weise wie bei der Verschiebung ausrechnen, sondern nur im Versuch ermitteln. Deshalb wird ein Ball mittels Rotation der Stange geschossen und in Abbildung 7.2 die Geschwindigkeit hieraus berechnet.

Die abgelesenen Werte auf dem Maßband neben dem Ball betragen $s_{start} = 187,0$ cm und $s_{end} = 169,0$ cm. Zwischen den Bildern ist ein Periode von $T = \frac{2}{30}$ s. Die Geschwindigkeit



(a) Startposition

(b) Endposition

Abbildung 7.2: Messung der Position des Balls zur Bestimmung seiner Geschwindigkeit nach einem Schuss

des Balls liegt dementsprechend bei

$$v_{Ball} = \frac{s_{start} - s_{end}}{T} = 2,7 \frac{\text{m}}{\text{s}}. \tag{7.6}$$

Dieser Wert ist recht hoch und entspricht fast der Anforderung N7 für die maximale Schussgeschwindigkeit. Es ist jedoch auch zu beachten, dass die Messung nicht ganz perfekt ist, da die genaue Position des Balls durch die Bewegungsunschärfe nicht zu erkennen ist. Andererseits variiert die Geschwindigkeit je nachdem wie der Ball getroffen wird, sodass die maximale Geschwindigkeit noch höher liegen kann. Deshalb wird die Anforderung als erfüllt betrachtet. Bei einem großen Tischkicker kann zudem die Geschwindigkeit durch die längeren Spielfiguren weiter erhöht werden. Allerdings ist hierfür vermutlich ein stärkerer Motor nötig.

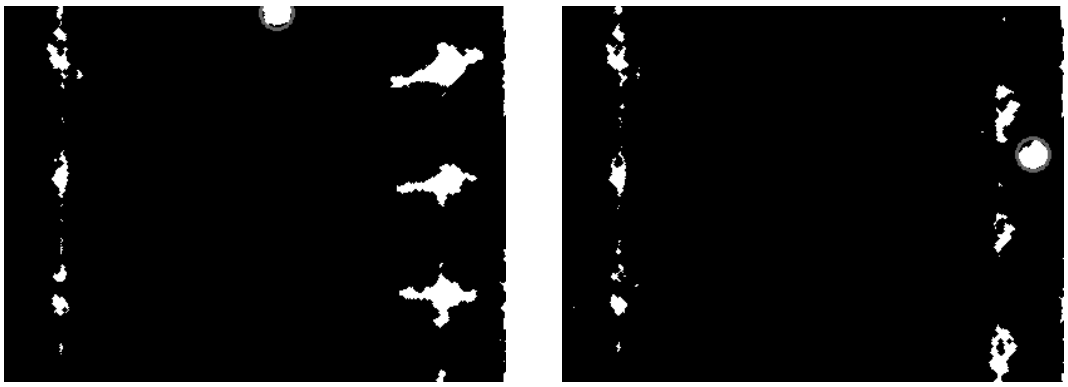
7.3 Kameradatenauswertung

Einerseits soll über die Kamera der Ball detektiert und hieraus seine Bewegungsrichtung ermittelt werden, andererseits wird auch die Erkennung von Fremdkörpern bzw. Händen

im Spielbereich über die Kameradaten durchgeführt. Für beide Verarbeitungsschritte muss eine gewisse Genauigkeit sowie eine hohe Zuverlässigkeit gegeben sein. Die hier implementierte Lösung ist lediglich ein erster Ansatz und unterliegt deshalb geringeren Anforderungen.

7.3.1 Zuverlässigkeit

Grundsätzlich obliegt die Balldetektion über die Tiefenkamera hohen Anforderungen an die Genauigkeit, gerade weil sie durch verschiedene Faktoren, wie das Verdecken des Balls durch die Spielfiguren, gestört wird. Zunächst lohnt sich die Überprüfung der Limitierungen der entwickelten Erkennung. Hierfür wird der Ball an verschiedenen Stellen auf dem Kicker platziert und überprüft, ob die Detektion möglich ist.
Auf den freien Flächen des Kickers ist die Erkennung kein Problem und gelingt in fast 100 % der Fälle. Etwas schwieriger wird es, wenn der Ball eine Figur oder Wand berührt. In der Regel kann auch hier eine erfolgreiche Detektion erfolgen, wobei dies auch von der genauen Position abhängig ist. Wie in Abbildung 7.3 gezeigt, kann der Ball auch im schmalen Bereich zwischen Tor und Spielfiguren erkannt werden. Dies gelingt jedoch nur auf der rechten Seite des Bilds, da der linke Bereich, wie in Abschnitt 6.3.1 beschrieben, gestört ist.

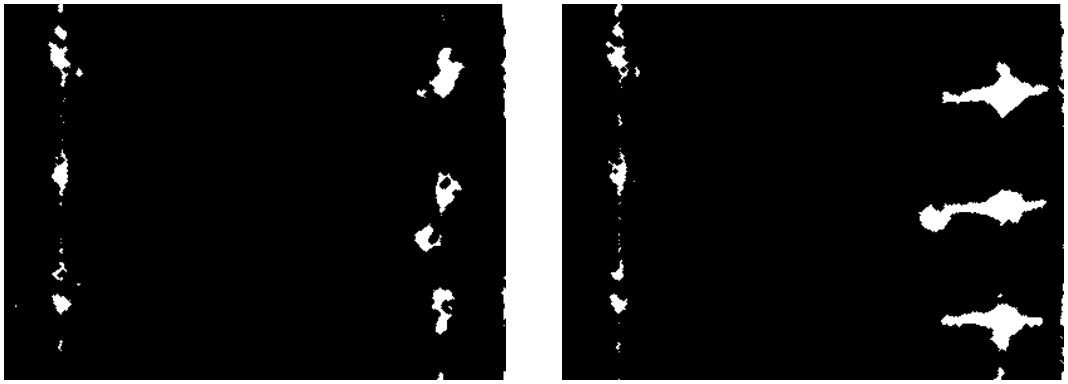


(a) Ball berührt eine Wand

(b) Ball zwischen den rechten Figuren und dem Tor

Abbildung 7.3: Beispiele erfolgreicher Balldetektion

Die Erkennung kommt jedoch an ihre Grenzen, wenn der Ball durch eine Stange oder Figur verdeckt ist. Dies ist z. B. in Abbildung 7.4 zu sehen. In seltenen Fällen kann es jedoch auch hier zu einer erfolgreichen Ermittlung der Position kommen.



(a) Ball liegt unter einer Stange (b) Ball ist durch Spielfigur verdeckt

Abbildung 7.4: Beispiele erfolgloser Balldetektion

Es kann in Abbildung 7.5 grob eine Einteilung der Bereiche auf dem Spielfeld erfolgen, wie gut die Balldetektion an der jeweiligen Stelle funktioniert. Im rot markierten Bereich gelingt die Erkennung nie, während im orangen Bereich je nach Position der Stangen teilweise die Detektion gelingt. Wenn sich der Ball im grünen Bereich befindet, ist die Erkennung leicht möglich.
Eine fälschliche Detektion eines anderen Objektes als Ball kommt fast nie vor.

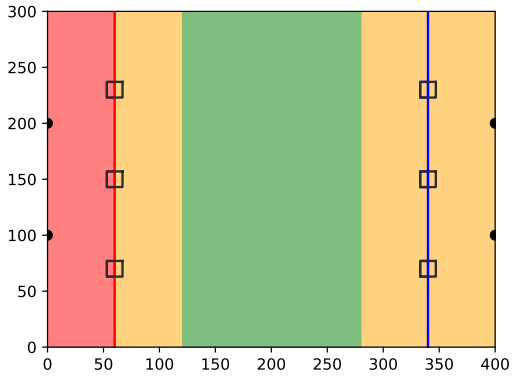


Abbildung 7.5: Einteilung der Bereiche des Spielfelds nach Stabilität der Balldetektion

7.3.2 Genauigkeit

Neben der Erfolgsquote der Detektion ist auch die Präzision entscheidend. In einem Versuch wird ein ruhender Ball beobachtet und seine Position auf der Abszisse aufgezeichnet

(siehe Abbildung 7.6). Hieraus wird die Standardabweichung der Messung $\sigma = 0,651$ mm berechnet. Dementsprechend befinden sich über 95 % der Messwerte in einem Bereich von $\pm 2 \cdot \sigma = \pm 1,302$ mm, sodass die aufgestellte Anforderung für die Genauigkeit der Auswertung erfüllt ist.

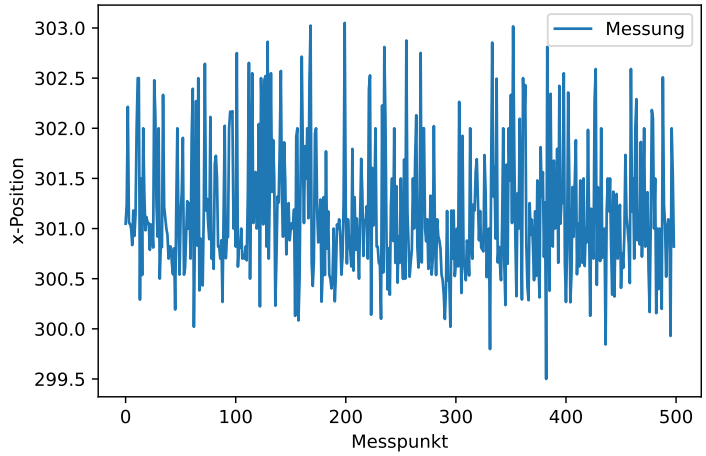
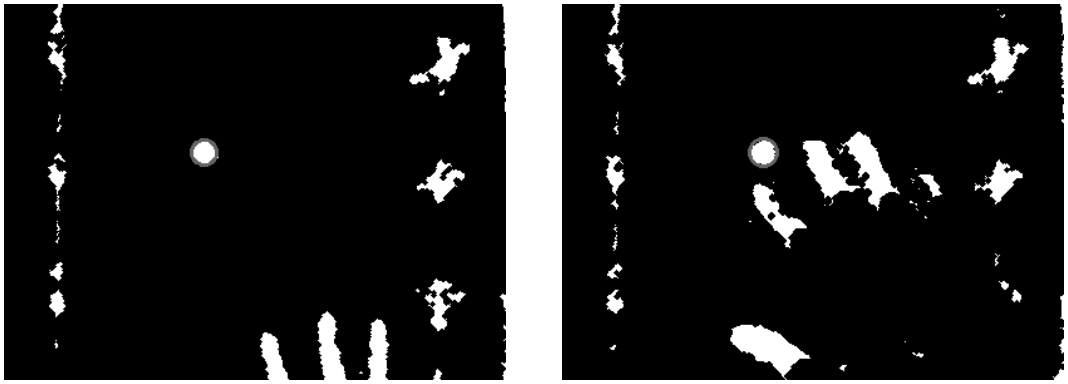


Abbildung 7.6: Rauschen der Messung der Position des Balls

Die Balldetektion erfüllt bereits hohe Anforderungen, obwohl es lediglich eine vorläufige Version ist. Die Genauigkeit und Präzision können durch das Einarbeiten des Kalmanfilters und die Erhöhung der Auflösung der Kamera noch verbessert werden, wodurch jedoch die Abtastrate verringert wird. Diese beträgt auf dem verwendeten System mit Intel Core i7-6700K 4 GHZ Prozessor und 16 GB Arbeitsspeicher maximal $f_s = 55$ Hz.

7.3.3 Handerkennung

Die Erkennung von Händen funktioniert ebenfalls sehr zuverlässig, obwohl sie sehr einfach ist. Die einzigen Limitierungen entstehen hier, wenn ein Mensch wie in Abbildung 7.7 nur einzelne Finger in den Spielbereich hält oder die Hand zu hoch hält und so außerhalb des Arbeitsbereichs der Tiefenkamera ist. Diese Fälle sind jedoch eher selten bzw. ungefährlich, da z. B. eine zu hohe Hand nicht mit den Spielfiguren in Berührung kommen kann.



(a) Einzelne Finger (b) Hand außerhalb des Arbeitsbereichs

Abbildung 7.7: Beispiele erfolgloser Detektion von Fremdkörpern im Spielbereich

7.4 Simulation

Das Training der KI wird mit Hilfe der Simulation durchgeführt, wodurch sich gewisse Anforderungen an diese ergeben. Einerseits müssen verschiedene Funktionen gewährleistet werden, andererseits ist natürlich auch die Genauigkeit der Abbildung verschiedener realer physikalischer Prozesse wichtig.

7.4.1 Allgemein

Die Simulation enthält alle beschriebenen Funktionen. Es ist möglich eine Simulation der Bewegung des Balls unter Berücksichtigung möglicher Kollisionen durchzuführen, wobei nach einen Kontakt die Bewegungsrichtung und Geschwindigkeit angepasst wird. Hierbei kann durch das Objekt, mit dem der Ball kollidiert, ein Impuls übertragen werden. Über die Steuerung der Reihen mit einer KI kann so der Ball aktiv bewegt und geschossen werden. Zudem können Tore ermittelt und der aktuelle Zustand in Form eines Tupels oder grafisch ausgegeben werden. Einzig abgeschrägte Ecken und Ränder können nicht dem System hinzugefügt und berücksichtigt werden. Da der ausgewählte Minikicker nicht über solche Ecken verfügt, ist diese Funktion nicht weiter relevant.
Über die Grundfunktionen hinaus lassen sich die verschiedenen Eigenschaften des Tischkickers anpassen. So sind Form, Größe und auch Eigenschaften wie die maximale Beschleunigung und Geschwindigkeit der Stangen einstellbar. Die Simulation kann dementsprechend an verschiedene Tischkicker und Hardwarekomponenten angepasst werden. Zudem können z. B. Verluste durch Reibung eingestellt werden.

7.4.2 Limitierungen

Die Funktionen der Simulation dürfen nach Möglichkeit kein unerwartetes und unrealistisches Verhalten zulassen, sodass die Übertragung der KI auf das reale System möglich ist. Ein wichtiger Faktor hierfür ist die Kollisionsdetektion und die mögliche anschließende Korrektur der Bewegung. Durch die diskrete Abtastung des Zustands der Simulation kann keine perfekte Kollisionsdetektion durchgeführt werden. Eine Berührung zweier Objekte lässt sich immer erst nach einer gewissen Überschneidung feststellen. Mit der flexiblen Anpassung der Abtastrate an die Bewegungsgeschwindigkeiten können der Ball und die Figuren nur eine zuvor festgelegte maximale Strecke zurücklegen. In den allermeisten Fällen wird so rechtzeitig eine Korrektur der Bewegung vorgenommen. Wenn der Ball eine Figur jedoch nur leicht streift, kann ein Kontakt wie in Abbildung 7.8 übersehen werden.

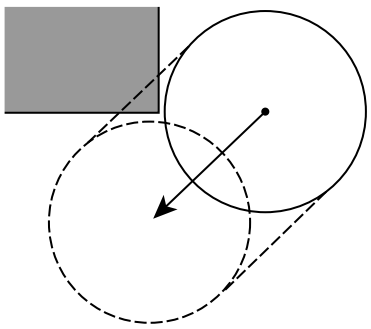


Abbildung 7.8: Nicht erkannter Kontakt beim Streifen zweier Objekte

Durch die Vernachlässigung der Gravitation und damit auch die Bewegung auf der Applikate können weitere Probleme entstehen. Wenn eine Spielfigur beispielsweise von oben auf den Ball trifft, wird der Ball gar nicht oder nur sehr wenig angestoßen. Die Reihe dreht sich jedoch weiter, sodass sich innerhalb einiger Schritte die Figur durch den Ball bewegt. Wahrscheinlich wird der Ball in diesen Fällen dann sogar entgegen der Bewegungsrichtung der Figur beschleunigt.
Die genannten Kritikpunkte sind im Kontext dieser Arbeit nicht weiter zu beachten, da sie sehr selten auftreten und dadurch nicht gezielt durch die KI ausgenutzt werden können.

7.4.3 Bewertung

Nach einer Kollision des Balls mit anderen Objekten muss die Bewegungsrichtung verändert werden. Die Korrektur der Bewegung ist vor allem vom Aufprallwinkel abhängig, welcher bei kleinen Änderungen der Position des Balls und der Spielfiguren bereits stark verändert wird. Deshalb ist ein messtechnischer Vergleich der Simulation mit dem Verhalten auf der Hardware nicht möglich. Stattdessen werden lediglich sehr viele Testfälle in der Simulation beobachtet und geprüft, ob die Ergebnisse auf dem echten Tischkicker ähnlich sind. Hierbei kann festgestellt werden, dass Simulation und Hardware gut übereinstimmen. Grundsätzlich entspricht das Verhalten in der Simulation meist der Erwartung. Nur in seltenen Fällen treten Fehler auf, die dann wahrscheinlich durch die oben beschriebene Problematik in der Kollisionsdetektion hervorgerufen werden.

Da die Simulation als Grundlage für das Training der KI dient und hierfür viele Trainingsdurchläufe benötigt werden, ist die Geschwindigkeit der Simulation ein wichtiger Faktor. Auf einem System mit einem Intel Core i7-6700K 4 GHZ Prozessor und 16 GB Arbeitsspeicher kann $T_{real} = 1$ s des Spiels je nach den Zuständen innerhalb von $T_{sim} < 0,1$ s simuliert werden, wobei auch die Berechnungen einer KI noch mit einberechnet sind. Dadurch ist die Simulation um einen Faktor 10 schneller als die simulierte Zeit.

7.4.4 Anforderungen

Nach der generellen Evaluierung werden nochmal die funktionalen und nicht-funktionalen Anforderungen an die Simulation hinsichtlich ihrer Erfüllung in dieser Arbeit untersucht. Dabei gibt es drei unterschiedliche Einordnungen: erfüllt, teilweise erfüllt und nicht erfüllt. Erfüllt bedeutet hierbei, dass die Anforderung komplett erfüllt wird, während eine teilweise erfüllte Anforderung nicht ganz den Erwartungen entspricht, aber dennoch ein akzeptables Ergebnis liefert. Nicht erfüllt sind Anforderungen, die nicht berücksichtigt wurden oder keine passende Lösung darstellen.

Funktionale Anforderungen

Tabelle 7.2 zeigt, dass die funktionalen Anforderungen an die Simulation in dieser Arbeit fast alle erfüllt sind. Lediglich die Implementierung von Gravitation und abgeschrägter

Ecken und Kanten wurde nicht berücksichtigt. Im Rahmen dieser Arbeit ist dies jedoch nicht weiter relevant und daher vertretbar.

Tabelle 7.2: Evaluierung der funktionalen Anforderungen der Simulation

ID.	Erfüllt	Anforderung
SF1	Ja	Simulation der Ballbewegung
SF2	Ja	Kollisionsdetektion zwischen Ball und Spielfiguren
SF3	Ja	Kollisionsdetektion zwischen Ball und Spielfeld
SF4	Ja	Erkennung von Toren
SF5	Ja	Steuerung der Rotation der Spielfiguren
SF6	Ja	Steuerung der Verschiebung der Spielfiguren
SF7	Teils	Einstellbarkeit der Größe, Beschaffenheit und des Gewichts des Spielballs
SF8	Teils	Einstellbarkeit der Größe und Form des Tischkickers
SF9	Ja	Einstellbarkeit der Anzahl und Position der Reihen und Figuren
SF10	Ja	Einstellbarkeit der maximalen Verfahrstrecke der Verschiebung
SF11	Ja	Einstellbarkeit der maximalen Geschwindigkeit der Rotation und Verschiebung
SF12	Ja	Anbindungsmöglichkeit zur Steuerung durch die KI
SF13	Ja	Ausgabe relevanter Informationen an die KI
SF14	Ja	Einstellbare visuelle Ausgabe des aktuellen Zustands

Nicht-funktionale Anforderungen

Auch die nicht-funktionalen Anforderungen der Simulation sind zum Großteil erfüllt. Hier ist lediglich anzumerken, dass die Simulation zwar grundsätzlich im 3-dimensionalen Raum erfolgt, jedoch Gravitation nicht berücksichtigt und deshalb Bewegungen des Balls entlang der Applikate nicht erfolgen. Hierdurch kann es in seltenen Fällen zu unerwünschtem Verhalten kommen, das aufgrund der Seltenheit für das Training nicht relevant ist.

Tabelle 7.3: Evaluierung der nicht-funktionalen Anforderungen der Simulation

ID.	Erfüllt	Anforderung
SN1	Ja	Objektorientierte Programmierung
SN2	Ja	Programmierung in Python
SN3	Ja	Rechenzeit der Simulation kleiner als simulierte Zeit
SN4	Ja	Kollisionsdetektion nach maximal $\Delta s = 2\ mm$ Bewegung des Balls
SN5	Teils	3-dimensionale Simulation
SN6	Ja	Berücksichtigung von Verlusten durch Reibung

7.5 KI

Mit der Simulation ist eine gute Grundlage für das Training von KI Modellen geschaffen. In dieser Arbeit wurden mehrere Modelle für unterschiedliche Anwendungsfälle präsentiert. Die Evaluierung dieser Ergebnisse in der Simulation und auf der Hardware dient der Abschätzung der nächsten Entwicklungsschritte des automatisierten Tischkickers.

7.5.1 Allgemeine Funktionen

Mit der KI können die Berechnungen der bestmöglichen Aktionen in einem gegebenen Zustand des Tischkickers durchgeführt werden. Hierbei kann durch die zwei entwickelten Umgebungen sowohl mit der Simulation als auch mit der Hardware interagiert werden.
Aktuell wurden lediglich Modelle für den Testaufbau des Minikickers entwickelt. Ein vollständiges Tischkickerspiel mit mehreren Reihen pro Spieler oder KI wurde bisher nicht untersucht. Das Training der KI erfolgt momentan ohne Interaktion mit einem Menschen oder einer anderen KI, wodurch sich z. B. die Bewegungsrichtung von Schüssen nicht vorhersagen lässt und deshalb erst beobachtet werden muss.
Bei extrem spielstarken Modellen kann es für einen Menschen nahezu unmöglich werden, ein Spiel gegen die KI zu gewinnen, weshalb eine Möglichkeit zur Einstellung der Spielstärke der KI sinnvoll sein kann. Diese Funktion wurde hier jedoch noch nicht eingearbeitet, da dies erst bei der Implementierung einer KI, die einen Menschen zuverlässig besiegen kann, interessant wird.

7.5.2 Tore verhindern

In einem ersten Anwendungsfall wurde eine KI zum Verhindern von Toren trainiert. Diese kann lediglich die Verschiebung der Stange steuern. Die Bewertung der KI wird sowohl in der Simulation als auch auf der Hardware durchgeführt. Hierbei kann sie auch mit dem programmierten Algorithmus verglichen werden.

Simulation

In der Simulation kann versucht werden eine Erfolgsrate der KI zu ermitteln. Hierfür werden 10.000 Episoden durchlaufen und die Anzahl der geschossenen Tore überprüft.

Dies wird auch für den einfachen manuell programmierten Algorithmus durchgeführt, sodass in Tabelle 7.4 ein Vergleich gezogen werden kann.
Da der Ball in der *reset()*-Funktion eine zufällige Position und Bewegungsrichtung erhält, kann dieser auch ohne Fremdeinwirkung das Tor verfehlen. Deshalb muss zunächst geprüft werden, wie viele Bälle ohne Torwart das Tor treffen, um eine qualitative Aussage über den Erfolg der KI zu machen.
Die bereinigte Erfolgsrate der Methoden wird aus der Division der gefallenen Tore der jeweiligen Methode und der der gefallenen Tore ohne Torwart berechnet.

Tabelle 7.4: Bestimmung der Erfolgsrate verschiedener Methoden zum Verteidigen von Bällen

Methode	Anzahl zugelassener Tore bei 10.000 Episoden	Erfolgsrate	
		unbereinigt	bereinigt
Kein Torwart	4205	57,95 %	0 %
Algorithmus	2468	75,32 %	41,31 %
KI	346	96,54 %	91,77 %

Die KI erreicht im Test eine sehr hohe unbereinigte, aber auch bereinigte Erfolgsrate von über 90 %. Der manuell programmierte Algorithmus hingegen kommt nicht annähernd auf diese Ergebnisse, sodass die bereinigte Erfolgsrate knapp über 40 % liegt. Es zeigt sich also, dass der Einsatz einer KI für diese Aufgabe geeignet ist.

Hardware

Ein interessanter Schritt ist die Übertragung der KI auf die Hardware. Die Simulation ist zwar auf den Tischkicker angepasst, doch ein perfektes Abbild ist nicht möglich, sodass mit einer niedrigeren Erfolgsrate zu rechnen ist.
Bei 100 auf das Tor geschossenen Bällen kann die KI 42 % halten. Dadurch ist die Genauigkeit auf der Hardware deutlich verringert. Dies liegt möglicherweise an kleinen Unterschieden der an die KI übergebenen Zustandsdaten in der Simulation und auf der Hardware.

7.5.3 Tore schießen

Der zweite Anwendungsfall der KI ist das Tore Schießen. Hierbei wird von der KI versucht einen ruhenden Ball in das gegnerische Tor zu schießen. Dabei kann in der Simulation eine

Erfolgsrate von 39,6 % erreicht werden. Dies ist im Vergleich zum Abwehren eines Balls deutlich geringer. Es gibt je nachdem wo der Ball liegt jedoch deutliche Unterschiede in der Erfolgsrate. Mittig platzierte Bälle können von der KI deutlich zuverlässiger ins Tor geschossen werden als Bälle, die seitlich liegen. Dies liegt daran, dass ein seitlich positionierter Ball mit der Kante der Spielfiguren getroffen werden muss, um schräg auf das Tor zu treffen. Kleine Unterschiede beim Winkel der Kollision verursachen deutliche Abweichungen in der Bewegungsrichtung. Abbildung 7.9 zeigt wie erfolgreich der Ball in den verschiedenen Bereichen geschossen wird. Grün bedeutet eine hohe Qualität, während in den orangefarbenen Bereichen nur selten ein Tor geschossen wird.
Wenn der Ball lediglich im mittleren Bereich platziert wird, kann sogar eine Erfolgsrate von 60,0 % erreicht werden.

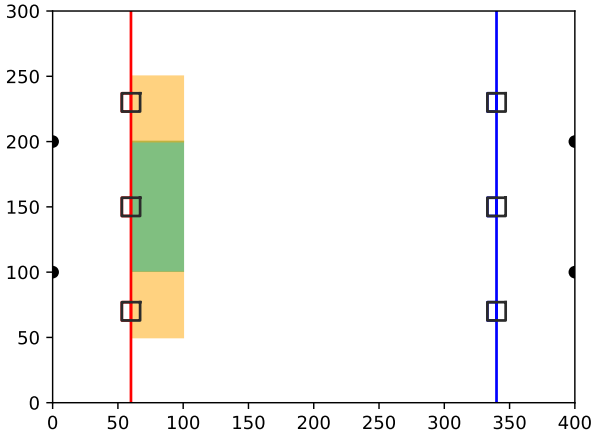


Abbildung 7.9: Erfolgsrate der Schüsse je nach Positionierung des Balls

Bei der Übertragung des Agenten auf die Hardware können keine zuverlässigen Ergebnisse erzeugt werden. Für diesen Awendungsfall ist scheinbar der Unterschied zwischen Simulation und Hardware zu groß, wobei ohnehin im Training eher schlechtere Ergebnisse erreicht werden.

7.5.4 Kombination

Nach den ersten beiden Anwendungsfällen gibt es noch die Kombination aus verteidigen und angreifen. Dies kommt einem richtigen Spiel zwischen zwei Spielern am nächsten.
Die entwickelte KI kann in den Testfällen zu 41,9 % ein Tor schießen, während sie in 7,9

% der Fälle ein Gegentor kassiert. Diese Werte sind recht beachtlich, da eine ähnliche (sogar etwas bessere) Erfolgsrate für das Schießen von Toren erreicht wird als bei der eigens hierfür entwickelten KI.
Die hier gemessene Leistung des Agenten muss jedoch noch eingeordnet werden, da z. B. die Anzahl der Gegentore durch die Wahl der Anfangszustände reduziert ist. Auch die geschossenen Tore sind ohne Berücksichtigung eines Gegenspielers, sodass die Erfolgsrate der KI in einem realen Spiel deutlich geringer sein dürfte.
Der hier trainierte Agent ist optimal für die Verwendung des Gesamtsystems auf der Hardware, da er dem realen Einsatzfall am nächsten kommt. Deshalb wird dieser ebenfalls auf dem Minikicker getestet. Hierbei zeigt sich, dass die KI noch lange nicht ihr volles Potential ausgeschöpft hat. Zudem wird die Stange fast durchgehend bewegt, sodass das Spiel sehr unruhig ist. Allerdings lässt sich durchaus ein unterhaltsames Spiel mit der KI realisieren, wobei der menschliche Benutzer in der Regel noch gewinnen sollte.

7.6 Anforderungen

Nachdem die Ergebnisse der einzelnen Entwicklungsschritte untersucht wurden, kann eine Diskussion der funktionalen und nicht-funktionalen Anforderungen an das Gesamtsystem erfolgen. Dabei gibt es erneut drei unterschiedliche Einordnungen: erfüllt, teilweise erfüllt und nicht erfüllt.

7.6.1 Funktionale Anforderungen

Zunächst werden die funktionalen Anforderungen in Tabelle 7.5 untersucht. Hierbei sind die wichtigsten Funktionen erfüllt. Nicht bearbeitet wurden verschiedene optionale Funktionen wie die Spielstandsanzeige oder die Ballmaschine, welche jedoch keinen direkten Einfluss auf das Spiel haben und deshalb vernachlässigt werden können.
Teilweise erfüllt werden die Anforderungen an die KI, welche gegen einen menschlichen Benutzer spielen können soll. Dies ist zwar grundsätzlich möglich, allerdings ist durch die geringere Erfolgsrate der KI auf der Hardware kein gleichwertiges Spiel zwischen Mensch und Maschine möglich.

Tabelle 7.5: Evaluierung der funktionalen Anforderungen des Tischkickers

ID	Erfüllt	Anforderung
F1	Ja	Erkennung der Position und Bewegung des Balls
F2	Ja	Erkennung der Verschiebung und Rotation der Spielfiguren
F3	Ja	Berechnung der nächsten Aktion durch eine KI
F4	Teils	Die KI kann gegen einen menschlichen Gegner Tore verhindern
F5	Teils	Die KI kann gegen einen menschlichen Gegner Tore schießen
F6	Nein	Einstellbarkeit des Schwierigkeitsgerades der KI
F7	Ja	Ansteuerung der Rotation der Spielfiguren
F8	Ja	Ansteuerung der Verschiebung der Spielfiguren
F9	Ja	Anhalten aller elektrisch beweglichen Teile
F10	Nein	Anzeige des Spielstands
F11	Nein	Automatische Ausgabe eines neuen Balls nach Knopfdruck
F12	Nein	Erkennung eines liegen gebliebenen Balls
F13	Nein	Automatisches ins Rollen bringen eines Balls
F14	Ja	Zugriff auf das System durch ein externes Gerät

7.6.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen werden in Tabelle 7.6 überprüft. Auch hier sind viele Punkte erfolgreich auf der Hardware implementiert, wobei jedoch zu beachten ist, dass nur ein erster einfacher Aufbau realisiert wurde, der z. B. die Auswertung der Position des Balls vereinfacht. Durch die Hindernisse auf dem Spielfeld kann die Balldetektion z. B. nicht immer erfolgreich durchgeführt werden, sodass hier keine Abtastrate von 50 Hz erreicht wird. Zudem wird die Bildverarbeitung aktuell auf einem externen Rechner durchgeführt, wodurch keine Aussage über die Geschwindigkeit der Auswertung auf einer mobilen Plattform gemacht werden kann.
Teilweise erfüllt sind einige Anforderungen an die Motorsteuerung, welche vor allem bei der Verschiebung nicht die gewünschten Werte erreicht. Für den Minikicker ist die aktuelle Steuerung noch ausreichend, da hier auch kürzere Wege zurückgelegt werden müssen. Zudem können höhere Geschwindigkeiten die Stabilität des Aufbaus reduzieren.
Nicht erfüllt werden die Anforderungen an die KI, welche die geforderten Erfolgsraten auf der Hardware nicht einhalten kann. Lediglich das Verteidigen gelingt mit über 90 %, jedoch nur in der Simulation.

Tabelle 7.6: Evaluierung der nicht-funktionalen Anforderungen des Tischkickers

ID	Erfüllt	Anforderung
N1	Ja	Bestimmung der Position des Balls mit ±2 mm Genauigkeit
N2	Ja	Bestimmung der Verschiebung der Stangen mit ±1 mm Genauigkeit
N3	Ja	Bestimmung der Rotation der Stangen mit ±1° Genauigkeit
N4	Ja	Verschiebung der Stangen mit einer maximalen Abweichung von ±1 mm
N5	Ja	Rotation der Stangen mit einer maximalen Abweichung von ±1°
N6	Teils	Verschiebung der Stangen mit einer Geschwindigkeit von bis zu $v_{Stange,max} = 2\ \frac{\mathrm{m}}{\mathrm{s}}$
N7	Ja	Schießen des Balls mit einer Geschwindigkeit von bis zu $v_{Ball,test} = 3\ \frac{\mathrm{m}}{\mathrm{s}}$
N8	Teils	Minimale Abtastrate von $f_{s,min} = 50\ \frac{1}{\mathrm{s}}$ zur Erfassung des Zustands
N9	Teils	Beschleunigung der Stangen auf $v_{Reak} = 1,5\ \frac{\mathrm{m}}{\mathrm{s}}$ innerhalb von $T_{Verschiebung} = 200$ ms
N10	Ja	Stoppen der Motoren innerhalb $T_{Krit} = 200$ ms bei der Erkennung einer menschlichen Hand
N11	Ja	Maximale Rechenzeit der KI kleiner als $T_{KI} = 20$ ms
N12	Teils	Die KI kann 80 % der Torschüsse halten
N13	Nein	Die KI kann 80 % der Schüsse auf ein leeres Tor treffen
N14	Nein	Die KI kann 50 % der Spiele gegen einen Menschen gewinnen
N15	Nein	Aufrecht positionierbarer Tisch
N16	Ja	Leichte Hardware mit einem Gewicht unter $m_{max} = 60$ kg
N17	Ja	Netzanschluss über einen Stecker
N18	Nein	Tischkicker als Standalone-System
N19	Ja	Einhaltung des Budgets von 500 €

8 Fazit und Ausblick

Im Rahmen dieser Arbeit konnte ein erster Aufbau eines KI gesteuerten Tischkickers realisiert werden. Dieser ermöglicht eine Interaktion zwischen KI und Mensch auf einem Minikicker, bei dem beide Seiten eine Reihe an Spielfiguren bewegen. Zwar erreichen die hier vorgestellten Modelle noch nicht die Leistung eines Menschen, jedoch kann trotzdem ein unterhaltsames Spiel durchgeführt werden.
Ein vollständiges und alleinstehendes System wurde in dieser Arbeit noch nicht entwickelt. Zum einen fehlen verschiedene Funktionalitäten wie die Torerkennung und eine Ballmaschine, während zum anderen auch noch keine alleinstehende Plattform implementiert ist, auf der das Gesamtsystem läuft. Dennoch können verschiedene KI-Modelle über einen externen Rechner an die Hardware angeschlossen werden, sodass ein potentieller Vergleich unterschiedlicher Ansätze durchführbar ist.

Die Bewegung der Spielfiguren erfolgt auf der Hardware mittels Schrittmotoren, die durch die KI gesteuert werden. Hierbei zeigt sich, dass der gewählte Motor für die Verschiebung zu wenig Kraft besitzt, um die gewünschten Geschwindigkeiten auf einem großen Tischkicker zu erreichen. Für den hier entwickelten Minikicker sind die erreichten Geschwindigkeiten jedoch hoch genug, um in einem Spiel gute Ergebnisse zu erzielen. Grundsätzlich wird der erste vorläufige Aufbau den Anforderungen an die Hardware gerecht. Bei folgenden größeren und vollständigen Konstruktionen sollte jedoch die Verwendung von kräftigeren Schrittmotoren für die Verschiebung berücksichtigt werden. Gleichzeitig muss auch die Stabilität des Aufbaus beibehalten werden, da die hohen Kräfte der Motoren sonst den gesamten Tisch in Bewegung versetzen können.

Da die KI nicht direkt das Kamerabild für die Auswahl seiner Entscheidung verwendet, musste in dieser Arbeit eine Auswertung der Tiefenkamera vorgenommen werden, um den Ball zu detektieren. Die Verarbeitung der Kameradaten ist jedoch nur ein Provisorium, damit der Fokus eher auf dem Training der KI liegen kann. Die entwickelte Lösung erreicht auf dem Minikicker eine recht hohe Zuverlässigkeit und Genauigkeit, die

für das Spielen auf dem System ausreichend ist. Dies liegt jedoch am reduzierten Aufbau, bei dem nur zwei von vier Stangen verwendet werden. Das Hauptproblem stellen hierbei die verschiedenen Figuren dar, da sie den Ball teilweise verdecken können und so die Balldetektion scheitern lassen. Für spätere Versionen des Tischkickers muss die Stabilität der Auswertung deutlich erhöht werden.
Die Auswertung der Verschiebung und Rotation einer Stange wurde nur für die von den Schrittmotoren gesteuerten Stangen vorgenommen. Für die Verbesserung der KI Algorithmen werden diese Daten jedoch benötigt, damit frühzeitig ein Schuss der gegnerischen Mannschaft vorhergesagt werden kann. Dies erlaubt auch eine geringere Abtastrate. Die derzeitig erreichten $f_s = 50$ Hz reichen ansonsten bei kleinen Abständen zwischen den Reihen nicht, um auf einen Schuss zu reagieren. Bei der verbesserten Auswertung des Zustands des Tischkickers muss also auch dieser Schritt noch hinzugefügt werden.

Das Training von verschiedenen RL-Methoden kann in der Simulation erfolgen. Hierbei wird zuverlässig das reale Verhalten eines einstellbaren Tischkickers angenähert. Zwar wird in der Simulation keine Gravitation berücksichtigt, doch lässt sich die Bewegungsrichtung des Balls plausibel annähern. Durch die begrenzte Aufrufrate der Kollisionsdetektion kann es gelegentlich zu unerwünschtem Verhalten kommen. Dies ist aufgrund der Seltenheit nicht weiter relevant.
Die Simulation ist eine passende Grundlage für das Training von künstlichen Intelligenzen. Eine Anpassung scheint zunächst nicht weiter nötig zu sein. Teilweise ist die Simulation vermutlich schon genauer als nötig, da die Übertragung der KI auf die Hardware noch mit gewissen Verlusten behaftet ist.

Die verschiedenen Agenten für die Interaktion mit dem Tischkicker erreichen in der Simulation bereits recht hohe Erfolgsraten. Die KI zum Verteidigen von Schüssen erreicht mit über 90 % Erfolgsrate ähnliche oder sogar höhere Werte als in verschiedenen verwandten Systemen [43]. Hierbei ist das Training in der Simulation wesentlich schneller, als bei der Lösung von ähnlichen Aufgaben unter Verwendung von Bildern als Eingang [70]. Dadurch ist das Entwickeln und Vergleichen verschiedener Lösungen einfacher und schneller möglich.
Die Übertragung des ersten Agenten auf die Hardware ist noch recht verlustbehaftet. Die Erfolgsrate sinkt deutlich, sodass die KI nicht mehr zuverlässig Bälle abwehren kann. Dies liegt vermutlich an kleinen Unterschieden in den Daten aus der Simulation und von der Hardware. Im nächsten Schritt muss dieser Abfall der Genauigkeit untersucht und gegebenenfalls behoben werden. Möglicherweise kann die Erfolgsrate der KI nur über das

Training direkt auf der Hardware verbessert werden.
Das Schießen des Balls durch eine KI erreicht schon in der Simulation nicht die in gewünschte Qualität. Die Limitierung scheint der verwendete Agent zu sein, welcher in nachfolgenden Arbeiten mit anderen Agenten und Lernmethoden verglichen werden muss. Die erreichten 40 % sind nicht vergleichbar mit ähnlichen Arbeiten auf diesem Gebiet [26]. Es ist jedoch zu beachten, dass erst die ersten Schritte der Entwicklung gemacht wurden und so noch kein fairer Vergleich möglich ist.
Auf der Hardware können die Schüsse nicht mehr zuverlässig das gegnerische Tor treffen. Trotzdem lässt sich ein unterhaltsames Spiel zwischen Mensch und KI durchführen, bei dem der Mensch jedoch noch recht überlegen ist.

Grundsätzlich konnten in dieser Arbeit die meisten Anforderungen erfüllt werden, sodass ein einfacher KI gesteuerter Tischkicker entwickelt wurde, mit dem ein Mensch interagieren kann. Die meisten Entwicklungsschritte benötigen jedoch noch Verbesserungen bzw. Weiterentwicklungen bevor das Gesamtsystem mit der gewünschten Genauigkeit arbeitet und als abgeschlossenes Projekt angesehen werden kann.

Literaturverzeichnis

[1] *Amazon Onlineshop.* – URL https://www.amazon.de/. – [Online; Stand 28. Januar 2023]

[2] *Bosch Laser Entfernungsmesser Zamo Set mit drei Adaptern.* – URL https://www.amazon.de/Bosch-Garden-Premium-Laser-Entfernungsmesser-Messbereich/dp/B07HYH9BW2/. – [Online; Stand 29. Januar 2023]

[3] *DOLD Mechatronik - Alles für Maker, Ingenieure und Visionäre.* – URL https://www.dold-mechatronik.de/. – [Online; Stand 28. Januar 2023]

[4] Aeberhard, Michael ; Connelly, Shane ; Tarr, Evan ; Walker, Nardis: *Single Player Foosball Table with an Autonomous Opponent*, Bachelorarbeit, 12 2007

[5] Allain, Rhett: *We Swear There's a Reason to Model This Ball Bouncing Off a Wall.* – URL https://www.wired.com/2016/05/swear-theres-reason-model-ball-bouncing-off-wall/. – [Online; Stand 12 Januar 2023]

[6] Anamika: *Program to find equation of a plane passing through 3 points.* – URL https://www.geeksforgeeks.org/program-to-find-equation-of-a-plane-passing-through-3-points/. – [Online; Stand 22. Januar 2023]

[7] Arbeitsrechte.de: *Lastenhandhabungsverordnung (LasthandhabV): Was ist zu beachten?* 2022. – URL https://www.arbeitsrechte.de/lastenhandhabungsverordnung/. – [Online; Stand 27. Januar 2023]

[8] AZ-Delivery: *Betriebsanleitung AZ-Delivery ESP32 DevKitC V2.* 2019. – Datenblatt

[9] Badia, Adrià P. ; Piot, Bilal ; Kapturowski, Steven ; Sprechmann, Pablo ; Vitvitskyi, Alex ; Guo, Zhaohan D. ; Blundell, Charles: Agent57: Outperforming the atari human benchmark. In: *International conference on machine learning* PMLR (Veranst.), 2020, S. 507–517

[10] Balzer, Paul: *Das Kalman-Filter einfach erklärt.* – URL https://www.cbcity.de/das-kalman-filter-einfach-erklaert-teil-1. – [Online; Stand 23. Januar 2023]

[11] Bambach, Sven ; Lee, Stefan: Real-Time Foosball Game State Tracking, 2012

[12] Becker, Alex: *About the Kalman filter.* 2022. – URL https://www.kalmanfilter.net/. – [Online; Stand 24. Januar 2023]

[13] Bošnak, Matevž ; Klančar, Gregor: Fast and Reliable Alternative to Encoder-Based Measurements of Multiple 2-DOF Rotary-Linear Transformable Objects Using a Network of Image Sensors with Application to Table Football. In: *Sensors* 20 (2020), 06, S. 3552

[14] Brockman, Greg ; Cheung, Vicki ; Pettersson, Ludwig ; Schneider, Jonas ; Schulman, John ; Tang, Jie ; Zaremba, Wojciech: *OpenAI Gym.* 2016

[15] Brownlee, Jason: *A Gentle Introduction to the Rectified Linear Unit (ReLU).* 2020. – URL https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. – [Online; Stand 30. Januar 2023]

[16] Caldwell, Shane: *How to Roll Your Own OpenAI Gym Environment.* – URL https://sjcaldwell.github.io/2020/05/21/openai-gym-intro.html. – [Online; Stand 06. Februar 2023]

[17] Chandra, Sudipto ; Nowshad, Md ; Islam, Mohammed J. ; Mukta, Marium-E-Jannat: An automated system to detect and recognize vehicle license plates of Bangladesh, 12 2017, S. 1–6

[18] Chernov, Nikolai ; Markarian, Roberto: *Chaotic billiards.* American Mathematical Soc., 2006 (127)

[19] Chong, Kah S.: *Collision Detection Using the Separating Axis Theorem.* 2012. – URL https://gamedevelopment.tutsplus.com/tutorials/collision-detection-using-the-separating-axis-theorem--gamedev-169. – [Online; Stand 25. Januar 2023]

[20] Connelly, Shane: *Autonomous Foosball Table*. 2007. – URL https://www.instructables.com/Autonomous-Foosball-Table/. – [Online; Stand 30. Januar 2023]

[21] Dahl, Øyvind N.: *How Transistors Work – A Simple Explanation*. 2021. – URL https://www.build-electronic-circuits.com/how-transistors-work/. – [Online; Stand 16. Januar 2023]

[22] Dahlkemper, Jörg: *Deep Learning in der Bildverarbeitung | Deep Learning*. 2020. – Vorlesung, HAW Hamburg

[23] Dahlkemper, Jörg: *Deep Learning in der Bildverarbeitung | Einführung in das Thema*. 2020. – Vorlesung, HAW Hamburg

[24] Dahlkemper, Jörg: *Deep Learning in der Bildverarbeitung | Training von Künstlichen Neuronalen Netzen*. 2020. – Vorlesung, HAW Hamburg

[25] Dawkins, Paul: *Equations Of Planes*. – URL https://tutorial.math.lamar.edu/classes/calciii/eqnsofplanes.aspx. – [Online; Stand 22. Januar 2023]

[26] De Blasi, Stefano ; Klöser, Sebastian ; Müller, Arne ; Reuben, Robin ; Sturm, Fabian ; Zerrer, Timo: KIcker: An Industrial Drive and Control Foosball System automated with Deep Reinforcement Learning, 04 2020

[27] DOLD Mechatronik: *Linear Motion Technology | ARC und HRC Linearführungen*. 6 2020. – Datenblatt

[28] DOLD Mechatronik: *iHSS57, iHSS60 Integrierte Schrittmotoren CL*. 12 2021. – Datenblatt

[29] dyn4j.org: *SAT (Separating Axis Theorem)*. 2010. – URL https://dyn4j.org/2010/01/sat/.. – [Online; Stand 25. Januar 2023]

[30] Fellers, Thomas J. ; Davidson, Michael W.: *Introduction to the Reflection of Light*. – URL https://www.olympus-lifescience.com/en/microscope-resource/primer/lightandcolor/reflectionintro/. – [Online; Stand 12 Januar 2023]

[31] Gasteiger, J. ; Zupan, J.: Neuronale Netze in der Chemie, 1993, S. 510–536

[32] Gollor, Friederike C.: *Entwicklung eines Systems für Oberflächenscans und die Erstellung von 3D-Modellen von Personen*, Masterarbeit, 12 2007

[33] Grace, Katja ; Salvatier, John ; Dafoe, Allan ; Zhang, Baobao ; Evans, Owain: When will AI exceed human performance? Evidence from AI experts. In: *Journal of Artificial Intelligence Research* 62 (2018), S. 729–754

[34] Gridin, Ivan: *Practical Deep Reinforcement Learning with Python*. BPB Publications, 2022

[35] gymlibrary.dev: *Cart Pole*. – URL https://www.gymlibrary.dev/environments/classic_control/cart_pole/. – [Online; Stand 28. Januar 2023]

[36] Götz, Paul: *8 Tischkicker Tipps Für Mehr Erfolg Beim Tischfussball*. – URL https://packeroo.com/tischkicker-tipps-erfolg-tischfussball/. – [Online; Stand 12. Februar 2023]

[37] Götz, Paul: *Spielregeln im Tischfußball | Eine Übersicht der wichtigsten Regeln im Tischfußball*. – URL https://www.kickerkult.de/Blog/Spielregeln-im-Tischfussball. – [Online; Stand 12. Februar 2023]

[38] hack.com next: *How to interface a 3.3V output to a 5V input*. 2020. – URL https://next-hack.com/index.php/2020/02/15/how-to-interface-a-3-3v-output-to-a-5v-input/. – [Online; Stand 15. Januar 2023]

[39] Hatfield, Scott: *What Type of Motor is Best for My Project*. 2018. – URL https://maker.pro/custom/tutorial/what-type-of-motor-is-best-for-my-project. – [Online; Stand 14. November 2022]

[40] Human Benchmark: *Reaction Time Test*. – URL https://humanbenchmark.com/tests/reactiontime. – [Online; Stand 22. Oktober 2022]

[41] Hunt, Trevor M. ; Hunt, T ; Vaughan, ND ; Vaughan, N: *The hydraulic handbook*. Elsevier, 1996

[42] Intel RealSense: *Product Family D400 Series*. 11 2022. – Datenblatt

[43] Janssen, Christina ; Gerlitz, Paul-Anton: *Elfmeterschießen in der Pförtnerloge*. 2021. – URL https://impact.h-da.de/elfmeterschiessen-in-der-pfoertnerloge. – [Online; Stand 28. Januar 2023]

[44] Janssen, Rob ; Best, Jeroen ; Molengraft, M.J.G.: Real-Time Ball Tracking in a Semi-automated Foosball Table, 06 2009, S. 128–139. – ISBN 978-3-642-11875-3

[45] Janssen, Rob ; Best, Jeroen de ; Molengraft, René van de ; Steinbuch, Maarten: The design of a semi-automated football table. In: *2010 IEEE International Conference on Control Applications*, 2010, S. 89–94

[46] Joseph, Jobit: *ESP32 Timers and Timer Interrupts*. 2022. – URL https://circuitdigest.com/microcontroller-projects/esp32-timers-and-timer-interrupts. – [Online; Stand 18. Januar 2023]

[47] Kálmán, Rudolf E.: A new approach to linear filtering and prediction problems"transaction of the asme journal of basic, 1960

[48] Kang ; Atul: *Perspective Transformation*. 2020. – URL https://theailearner.com/tag/cv2-getperspectivetransform/. – [Online; Stand 18. Januar 2023]

[49] Kathuria, Ayoosh: *The Machine Learning Practitioner's Guide to Reinforcement Learning: All About Markov Decision Processes*. – URL https://blog.paperspace.com/reinforcement-learning-for-machine-learning-folks/. – [Online; Stand 03. Februar 2023]

[50] Kempke, Jonas: *Videobasierte Linienbus- und Fahrtzielerkennung mittels Verfahren des Maschinellen Lernens für einen elektronischen Blindenführhund*, Bachelorarbeit, 2021

[51] Keras: *Keras-RL Dokumentation*. – URL https://keras-rl.readthedocs.io/en/latest/. – [Online; Stand 06. Februar 2023]

[52] Khandelwal, Renu: *Supervised, Unsupervised, and Reinforcement Learning*. – URL https://arshren.medium.com/supervised-unsupervised-and-reinforcement-learning-245b59709f68. – [Online; Stand 03. Februar 2023]

[53] Kim, Youngjoo ; Bang, Hyochoong: *Introduction to Kalman Filter and Its Applications*, 11 2018. – ISBN 978-1-83880-536-4

[54] Kohler, Florian: *How to make kick and bank shots*. – URL https://www.pooldawg.com/article/pooldawg-library/how-to-make-kick-bank-shots. – [Online; Stand 12 Januar 2023]

[55] Li, Yuxi: *Deep Reinforcement Learning: An Overview*. 2017. – URL https://arxiv.org/abs/1701.07274

[56] Maschinenbau-Wissen: *Den realen Stoß berechnen - Teilelastisch / Teilplastisch*. – URL https://www.maschinenbau-wissen.de/skript3/mechanik/kinetik/342-realer-stoss. – [Online; Stand 12 Januar 2023]

[57] Mateer, Patrick: *Understanding (and fixing) lens distortion*. – URL https://shotkit.com/lens-distortion/. – [Online; Stand 20. Januar 2023]

[58] matematica.pt: *What is the difference between a convex and a non-convex polygon?*. – URL https://www.matematica.pt/en/faq/convex-polygon.php. – [Online; Stand 26. Januar 2023]

[59] Mathavan, Senthan ; Jackson, M ; Parkin, Robert: A theoretical analysis of billiard ball dynamics under cushion impacts. In: *Proceedings of The Institution of Mechanical Engineers Part C-journal of Mechanical Engineering Science - PROC INST MECH ENG C-J MECH E* 1 (2010), 09, S. 1–10

[60] Matplotlib: *Matplotlib: Visualization with Python*. – URL https://matplotlib.org/. – [Online; Stand 9. Januar 2023]

[61] Matplotlib: *Pong*. – URL https://matplotlib.org/stable/gallery/event_handling/pong_sgskip.html. – [Online; Stand 9. Januar 2023]

[62] Mei, Gang: *RealModel-a system for modeling and visualizing sedimentary rocks*, Dissertation, 07 2014

[63] Mnih, Volodymyr ; Kavukcuoglu, Koray ; Silver, David ; Graves, Alex ; Antonoglou, Ioannis ; Wierstra, Daan ; Riedmiller, Martin: Playing atari with deep reinforcement learning. In: *arXiv preprint arXiv:1312.5602* (2013)

[64] Mnih, Volodymyr ; Kavukcuoglu, Koray ; Silver, David ; Rusu, Andrei ; Veness, Joel ; Bellemare, Marc ; Graves, Alex ; Riedmiller, Martin ; Fidjeland, Andreas ; Ostrovski, Georg ; Petersen, Stig ; Beattie, Charles ; Sadik, Amir ; Antonoglou, Ioannis ; King, Helen ; Kumaran, Dharshan ; Wierstra, Daan ; Legg, Shane ; Hassabis, Demis: Human-level control through deep reinforcement learning. In: *Nature* 518 (2015), 02, S. 529–33

[65] Mohebi, Dani: The Study of Semi-Automated Football Table, 2022

[66] Mota, Matheus: *JupyterLab is the data science UI we have been looking for.* – URL https://towardsdatascience.com/jupyterlab-you-should-try-this-data-science-ui-for-jupyter-right-now-a799f8914bb3. – [Online; Stand 06. Februar 2023]

[67] Nielsen, Michael: *Improving the way neural networks learn.* 2019. – URL http://neuralnetworksanddeeplearning.com/chap3.html. – [Online; Stand 30. Januar 2023]

[68] NVIDIA: *Jetson Nano Developer Kit.* – URL https://developer.nvidia.com/embedded/jetson-nano-developer-kit. – [Online; Stand 28. Januar 2023]

[69] Nzuzi, V. Poltavets E. Graziani D.: *Algorithmische Anwendungen | Projekt Bildsegmentierung.* 2006. – URL http://www.gm.fh-koeln.de/~hk/lehre/ala/ws0506/Praktikum/Projekt/F_rot/Bildsegmentierung_Otsu.pdf. – [Online; Stand 12. Februar 2023]

[70] Omkar, V.: *Playing Pong using Reinforcement Learning.* 2019. – URL https://towardsdatascience.com/intro-to-reinforcement-learning-pong-92a94aa0f84d. – [Online; Stand 28. Januar 2023]

[71] OpenAI Gym: *Gym Documentation | Spaces.* – URL https://www.gymlibrary.dev/api/spaces/. – [Online; Stand 06. Februar 2023]

[72] OpenCV: *Geometric Image Transformations.* – URL https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html. – [Online; Stand 18. Januar 2023]

[73] OpenCV: *Image Thresholding.* – URL https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html. – [Online; Stand 12. Februar 2023]

[74] OpenCV: *About Open Cv.* 2021. – URL https://opencv.org/about/. – [Online; Stand 4. April 2021]

[75] Oppermann, Artem: *Aktivierungsfunktionen in neuronalen Netzen: Sigmoid, tanh, ReLU.* – URL https://artemoppermann.com/de/aktivierungsfunktionen/. – [Online; Stand 06. Februar 2023]

[76] Oppermann, Artem: *Deep Q-Learning.* – URL https://artemoppermann.com/de/deep-q-learning/. – [Online; Stand 03. Februar 2023]

[77] O'Shea, Keiron ; Nash, Ryan: An introduction to convolutional neural networks. In: *arXiv preprint arXiv:1511.08458* (2015)

[78] Parák, Bc. R.: Robotic table football - Game strategy, 2017

[79] Pascanu, Razvan ; Mikolov, Tomas ; Bengio, Yoshua: On the difficulty of training recurrent neural networks. In: *International conference on machine learning* Pmlr (Veranst.), 2013, S. 1310–1318

[80] Pharr, Matt ; Jakob, Wenzel ; Humphreys, Greg: *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016

[81] Pilz GmbH: *Unschlagbar mit der automatisierten Elf von Pilz!*. – URL https://www.youtube.com/watch?v=QUJEM__JFWQ. – [Online; Stand 30. Januar 2023]

[82] Piovoso, Michael ; Laplante, Phillip A.: Kalman Filter Recipes for Real-Time Image Processing. In: *Real-Time Imaging* 9 (2003), dec, Nr. 6, S. 433–439. – URL https://doi.org/10.1016/j.rti.2003.09.005. – ISSN 1077-2014

[83] Project Jupyter: *Jupyter kernels*. – URL https://github.com/jupyter/jupyter/wiki/Jupyter-kernels. – [Online; Stand 06. Februar 2023]

[84] PyQtGraph: *PyQtGraph | Scientific Graphics and GUI Library for Python | Introduction*. – URL https://pyqtgraph.readthedocs.io/en/latest/getting_started/introduction.html. – [Online; Stand 9. Januar 2023]

[85] Quinones, Jose I.: *Applying acceleration and deceleration profiles to bipolar stepper motors*. 2012

[86] Ratcliff, Roger: Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. In: *Psychological review* 97 (1990), Nr. 2, S. 285

[87] Ridden, Paul: *It's only a game: Robots defeat humans on foosball playing field*. – URL https://newatlas.com/epfl-robot-table-soccer-foosball/44863/. – [Online; Stand 28. Januar 2023]

[88] RTE France: *Welcome to Grid2Op's technical documentation!*. – URL https://grid2op.readthedocs.io/en/latest/index.html. – [Online; Stand 06. Februar 2023]

[89] SCIPY.ORG: *scipy.linalg.lstsq()*. – URL https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lstsq.html. – [Online; Stand 22. Januar 2023]

[90] SENDEN, Jordy ; JEBBINK, Kevin ; BRUYNINCKX, Herman ; MOLENGRAFT, René van de: Invariant-Based World Models for Robust Robotic Systems Demonstrated on an Autonomous Football Table. In: *IEEE Robotics and Automation Letters* 7 (2022), Nr. 3, S. 8542–8549

[91] SHARMA, Sagar ; SHARMA, Simone ; ATHAIYA, Anidhya: Activation functions in neural networks. In: *Towards Data Sci* 6 (2017), Nr. 12, S. 310–316

[92] SOUTO, Nilson: *Collision Detection for Solid Objects*. – URL https://www.toptal.com/game/video-game-physics-part-ii-collision-detection-for-solid-objects. – [Online; Stand 25. Januar 2023]

[93] STABLE BASELINES: *Welcome to Stable Baselines docs! - RL Baselines Made Easy*. – URL https://stable-baselines.readthedocs.io/en/master/index.html. – [Online; Stand 06. Februar 2023]

[94] SUBRAMANYAM, Vineeth S.: *Basics of Bounding Boxes*. – URL https://medium.com/analytics-vidhya/basics-of-bounding-boxes-94e583b5e16c. – [Online; Stand 26. Januar 2023]

[95] SZYK, Bogna: *Spherical Coordinates Calculator*. – URL https://www.omnicalculator.com/math/spherical-coordinates. – [Online; Stand 9. Januar 2023]

[96] TENSORFLOW.ORG: *Playing CartPole with the Actor-Critic method*. – URL https://www.tensorflow.org/tutorials/reinforcement_learning/actor_critic. – [Online; Stand 28. Januar 2023]

[97] ULLRICH PROSPORT: *Ullrich-ProSport-Tisch*. – URL https://shop.ullrich-sport.de/product.php?id_product=115. – [Online; Stand 28. Januar 2023]

[98] UNIVERSITÄT STUTTGART: *Automatisierter Tischkicker*. – URL https://www.ias.uni-stuttgart.de/forschung/demonstratoren/tischkicker/. – [Online; Stand 30. Januar 2023]

[99] Van Hasselt, Hado ; Guez, Arthur ; Silver, David: Deep reinforcement learning with double q-learning. In: *Proceedings of the AAAI conference on artificial intelligence* Bd. 30, 2016

[100] Veturi, Yoga A. ; Woof, William ; Lazebnik, Teddy ; Moghul, Ismail ; Wagner, Siegfried K. ; Guimarães, Thales Antonio C. de ; Varela, Malena D. ; Liefers, Bart ; Patel, Praveen J. ; Beck, Stephan u. a.: SynthEye: Investigating the Impact of Synthetic Data on Artificial Intelligence-assisted Gene Diagnosis of Inherited Retinal Disease. In: *Ophthalmology Science* 3 (2023), Nr. 2

[101] Weigel, Thilo ; Nebel, Bernhard: KiRo – An Autonomous Table Soccer Player, 11 2003, S. 384–392. – ISBN 978-3-540-40666-2

[102] Werbos, Paul J.: Backpropagation through time: what it does and how to do it. In: *Proceedings of the IEEE* 78 (1990), Nr. 10, S. 1550–1560

[103] Wikimedia Commons: *File:Typical cnn.png — Wikimedia Commons, the free media repository*. 2021. – URL https://commons.wikimedia.org/w/index.php?title=File:Typical_cnn.png&oldid=535187198. – [Online; Stand 30. Januar 2023]

[104] Wikimedia Commons: *File:Verzeichnung3.png — Wikimedia Commons, the free media repository*. 2021. – URL https://commons.wikimedia.org/w/index.php?title=File:Verzeichnung3.png&oldid=524171061. – [Online; accessed 20-January-2023]

[105] Wikimedia Commons: *File:3D Spherical.svg — Wikimedia Commons, the free media repository*. 2022. – URL https://commons.wikimedia.org/w/index.php?title=File:3D_Spherical.svg&oldid=708087267. – [Online; Stand 11 Januar 2023]

[106] Wuttke, Laurenz: *Reinforcement Learning: Wenn KI auf Belohnungen reagiert*. – URL https://datasolut.com/reinforcement-learning/. – [Online; Stand 03. Februar 2023]

[107] XIMEA: *Human versus Robot using high speed XIMEA camera*. – URL https://www.ximea.com/en/corporate-news/foosball-table-robot-camera. – [Online; Stand 25. Oktober 2022]

[108] Ye, Andre: *Markov Decision Process in Reinforcement Learning: Everything You Need to Know*. – URL https://neptune.ai/blog/markov-decision-process-in-reinforcement-learning. – [Online; Stand 03. Februar 2023]

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original