Entwicklung eines Schachspielenden Roboters

Studienarbeit

INF2IN

Linus Wilkens

I

Inhalt

[Abstract 4](#_Toc168238285)

[Motivation 4](#_Toc168238286)

[Aufgabenstellung 4](#_Toc168238287)

[Projektplanung 4](#_Toc168238288)

[Funktionale Anforderungen 4](#_Toc168238289)

[Nicht Funktionale Anforderungen 6](#_Toc168238290)

[Out of Scope/Abgrenzung 6](#_Toc168238291)

[Zeitplan 7](#_Toc168238292)

[Theoretische Grundlagen 9](#_Toc168238293)

[Ähnliche Projekte/Umsetzungen 9](#_Toc168238294)

[3D Druck Verfahren 9](#_Toc168238295)

[3D Druck Materialen 10](#_Toc168238296)

[Aktoren 11](#_Toc168238297)

[Linearmotor 11](#_Toc168238298)

[Schrittmotor 12](#_Toc168238299)

[Direkte Kinematik 14](#_Toc168238300)

[CAD-Modellierung 14](#_Toc168238301)

[TinkerCad 14](#_Toc168238302)

[Erster Linearmotor 15](#_Toc168238303)

[Zweiter Linearmotor 15](#_Toc168238304)

[Drehmechanismus 15](#_Toc168238305)

[Magnethalterung 15](#_Toc168238306)

[Überblick CAD Modellierung 15](#_Toc168238307)

[Elektronik/Schaltungsaufbau/Hardware 15](#_Toc168238308)

[Figuren Erkennung 15](#_Toc168238309)

[Theoretische Grundlagen 15](#_Toc168238310)

[Schachbrett 15](#_Toc168238311)

[Digitale Bilder und RGB-Werte 17](#_Toc168238312)

[Konzept 18](#_Toc168238313)

[Methodik zur Spielzugerkennung 18](#_Toc168238314)

[Bestimmung des gespielten Zuges 20](#_Toc168238315)

[Methodik zum Bilderfassungszeitpunkt 21](#_Toc168238316)

[Umsetzung 22](#_Toc168238317)

[Hardware 22](#_Toc168238318)

[Software-Implementierung 22](#_Toc168238319)

[Tests und Probleme 27](#_Toc168238320)

[Verbesserungen 29](#_Toc168238321)

[Zuggenerierung und -ausführung 30](#_Toc168238322)

[Konzept 30](#_Toc168238323)

[Umsetzung 32](#_Toc168238324)

[Herausforderungen 35](#_Toc168238325)

[Heben aus engen Stellen heraus 35](#_Toc168238326)

[Figurenerkennung 35](#_Toc168238327)

[Fazit 35](#_Toc168238328)

[Kritische Reflektion 35](#_Toc168238329)

[Ausblick 35](#_Toc168238330)

[Literaturverzeichnis 36](#_Toc168238331)

# Abstract

# Motivation

# Aufgabenstellung

In dieser Arbeit wird die Realisierung eines Schachs spielenden Roboters behandelt werden. Hierbei sollen sowohl die mechanische Umsetzung des Roboters als auch die Softwareseitige Entwicklung betrachtet. In der Projektplanung zu beginn der Arbeit wird außerdem ein Konzept für das Projekt entwickelt.

# Projektplanung

## Funktionale Anforderungen

Die Anforderungen an das Projekt können in Funktionale und Nicht Funktionale Anforderungen unterteilt werden. Unter die Funktionalen Anforderungen fallen hierbei die Funktionen des Projektes, während es sich bei den nicht funktionalen Anforderungen um die Qualität dieser Funktionen handelt.

Die Funktionalen Anforderungen des Projektes können in die folgenden drei Teilbereiche unterteilt werden:

Hardware

Für die Hardware des Roboter Arms soll sich an einem Bagger Arm orientiert werden. Der Roboter Arm soll später 2 Gelenkstellen haben sowie einen Elektromagnet am vorderen Teil des Arms. Um eine Drehung zu gewährleisten, soll der Arm auf einer Drehscheibe stehen und mit einem Drehbereich von 180 Grad bewegbar sein. Anstelle von Hydraulik Zylindern soll für den Roboterarm auf Linearmotor zurückgegriffen werden. Wichtig ist, dass der Bewegungsbereichs des Roboterarms den Bereich eines Schachfeldes erreichen kann.

Neben dem Roboter Arms soll es eine Kamera geben, welche das Spielfeld erkennt und die einzelnen Figuren tracken kann. Die Figuren sollen dann von dem Roboterarm gegriffen/aufgenommen werden können und an eine andere Position gestellt werden können.

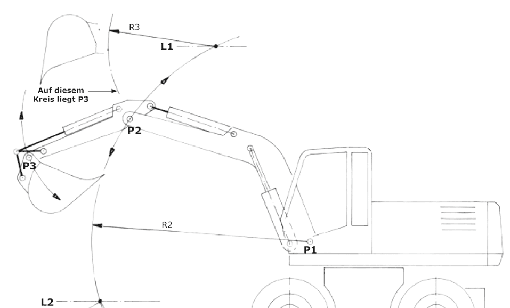


Abbildung 1: Abbildung Bagger

Elektronik

Der Roboter Arm soll später mithilfe eines Netzteils und einer handelsüblichen Steckdose betrieben werden können. Die Elektronik soll hierbei sicher gestaltet sein damit auch projektfremde Personen den schachspielenden Roboter gefahrlos benutzen können.

Software

Die Software soll in der Lage sein das Schachbrett zu erkennen. Dazu soll es Veränderungen von Positionen der Figuren auf dem Spielfeld erkennen können und damit entscheiden welche Figuren vom menschlichen Spieler bewegt wurden.

## Nicht Funktionale Anforderungen

Auch die nicht funktionalen Anforderungen können in die drei Bereiche unterteilt werden.

Hardware

Für den Roboter Arm ist es besonders wichtig, dass dieser präzise die einzelnen Positionen anfahren kann. Auch eine große Anzahl von Spielzügen soll nicht zu einer Verringerung der Genauigkeit führen.

Elektronik

Der Schaltungsaufbau der Elektronik soll sauber und nachvollziehbar gestalten werden, damit späteren Weiterentwicklungen oder Wartungen problemlos vollzogen werden können.

Software

Die Software soll, bei einer Partie von 38 Zügen, die Entscheidung über veränderte Figurenpositionen mit einer Quote von über 99% treffen, um möglichst viele fehlerlose Spielpartien zu ermöglichen.

## Out of Scope/Abgrenzung

Die Wirtschaftlichkeit und die Energieeffizienz des Schach Spielenden Roboters sollen kein Teil der Arbeit sein. Es handelt sich hierbei um ein Einzelstück, weshalb diese Punkte vernachlässigt werden können.

Auch soll der Roboter nicht universal an jedem Schachfeld spielen können, sondern wird explizit für **ein eigenes** Schachspiel entwickelt.

## Zeitplan

Für den Zeitplan und der Strukturierung des Projektes dient ein sogenanntes Gant Diagramm. Das Diagramm ist nachfolgend in Tabellenform grafisch dargestellt.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2023 KW | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 |
| Konzeptionierung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| IdEenfindung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Theoretische EInarbeitung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Konzeptionierung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Protoytbau |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CAD-Modellierung Roboter |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Moedllierung Spiel Erkennung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Zusammenbau des Prototyps |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Dokumentation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Struktur |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Einleitung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Technische Grundlagen |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Umsetzung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fazit |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2024 KW | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Programmierung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Puffer |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Figuren Erkennung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Schach Algorithmus |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Mechanische Umsetzung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| FUnktionstest |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Funktionstests |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fehler Korrektur |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Dokumentation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Struktur |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Einleitung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Technische Grundlagen |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Umsetzung |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Fazit |  |  |  |  |  |  |  |  |  |  |  |  |  |

Die Restliche Zeit bis zur Abgabe soll als Puffer und für mögliche Verzögerungen aufgrund von Lieferschwierigkeiten oder ähnliches verwendet werden.

# Theoretische Grundlagen

## Ähnliche Projekte/Umsetzungen

Als Inspiration für die Arbeit dienen zwei ähnliche Projekte (<https://www.raspberryturk.com/> und <https://www.mintgruen.tu-berlin.de/robotikWiki/doku.php?id=projektewise20:schachroboterpublic:start#die_bilderkennung>)

Der Unterschied ist, dass sowohl bei dem Projekt der Tu Berlin als auch [Raspberry Turk](https://www.raspberryturk.com/)

ein Art Schienen System zum Einsatz kommt und kein „Bagger Inspirierter“ Roboter Arm.

[…]

## 3D Druck Verfahren

3D ist ein Verfahren zur Herstellung von Werkstücken aus Pulver, geschmolzenen Materialen oder flüssigen Materialen. Die Materialen sind hierbei überwiegend verschiedene Kunststoffarten können allerdings auch Metalle, Keramik oder Beton sein. Je Material und Anwendungsgebiet gibt es hierfür verschiedene Verfahrensarten. Das am häufigsten verwendete Verfahren ist das Fused Deposition Modeling (FDM) Verfahren bei welchem das Material über eine beheizte Düse geschmolzen, und anschließend in dünnen Druckschichten auf ein Druckbett gepresst wird. Um die gewünschte Form zu bilden, verfährt die Druckdüse über die X und Y-Achse. Nach jeder Schicht wir die Druckdüse anschließend gehoben und somit ein 3D Dimensionales Objekt erstellt. Mithilfe dieses Verfahrens ist ein Maximaler Überhangwinkeln von 45 Grad möglich. Sollte ein größere Überhangbenötigt werden muss auf Stützstrukturen zurückgegriffen werden welche anschließend wieder entfernt werden müssen. Diese Faktoren sind beim Design von 3D Modellen zu berücksichtigen. (Krause, 2021)

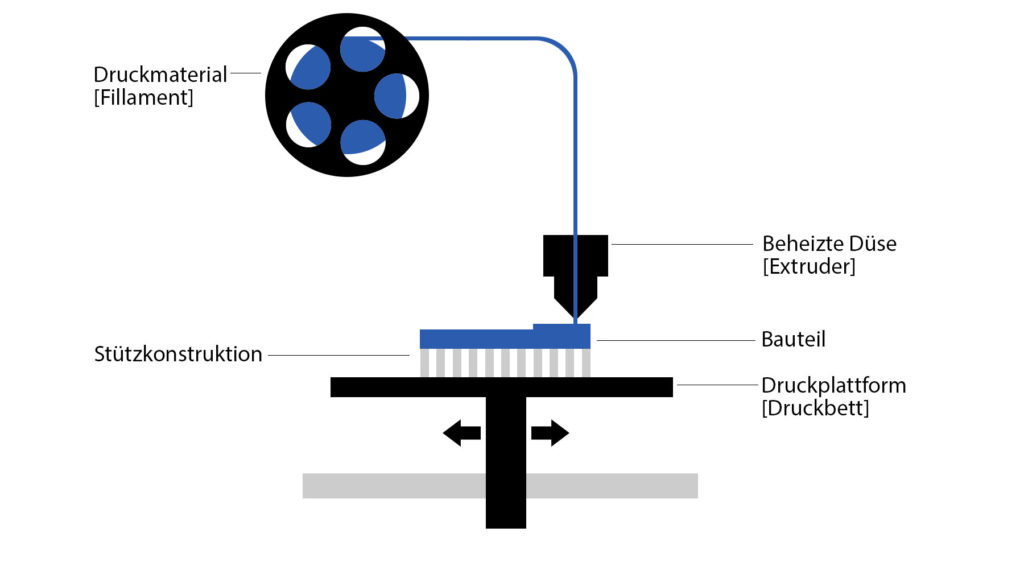


Abbildung 2: Funktionsweise des FDM-Druckverfahren

## 3D Druck Materialen

Mithilfe der FDM-Druckverfahren können verschiedene Materialen gedruckt werden. Aufgrund des vorhandenen Druckers und den bereits vorhanden Materialen werden nachfolgend nur ABS und PLA verglichen.

PLA ist das am häufigsten verwendetes Material im Privatgebrauch. Es ist kostengünstig und leicht zu drucken. Ein Nachteil ist die hohe Sprödigkeit und die damit verbundene geringe Resistenz gegen Schlageinwirkungen.

ABS ist neben PLA eins der am häufigsten verwendeten Materialen im Privatgebrauch. Es hat gute Materialeigenschaften im Bezug auf Stabilität und Halterbarkeit. Auch preislich gibt es nur geringe unterschiede zu PLA. Der Nachteil an ABS ist die starke Temperaturabhängige während dem Druck. So ist es nur mit sehr hohem Aufwand und einer stabilen Umgebungstemperatur ABS mit strengen Toleranzen zu drucken (O'Connell, 2021).

|  |  |  |
| --- | --- | --- |
| Eigenschaft | PLA | ABS |
| Druckaufwand | Gering | Hoch |
| Löslichkeit | Nicht löslich | In Aceton und Ester Ketonen |
| Festigkeit | Durchschnittlich | Hoch |
| Verzugsverhalten | Gering | Hoch |
| Druckbett Temperatur | Optional (20-60°C) | 80 -120°C |
| Druckdüse Temperatur | 190-210°C | 220-250°C |

Aufgrund der aufgelisteten Eigenschaften werden alle in der Arbeit verwendeten Teile aus PLA gedruckt. Sollte in Einzelfällen Teile aus ABS benötigt werden ist dies explizit gekennzeichnet.

# Aktoren

## Linearmotor

Linearmotoren sind Motoren, welchen mithilfe eines klassischen Motors und einer Gewindestage eine Drehbewegung in eine lineare Bewegung übersetzt. Hierfür wird die Bewegung mithilfe eines Getriebes zunächst übersetzt und anschließend auf die Gewindestange übertragen. Mithilfe von Sensoren an beiden enden des Linearmotors können ebenfalls die Endstellungen erkannt werden. Diese Funktion bieten allerdings nicht alle Linearmotoren.

Linearmotoren bieten den Vorteil, dass keine Umwandlung der rotarischen Bewegung in eine Linearbewegung außerhalb des Linearmotors stattfinden muss. Dies eliminiert die Notwendigkeit von einem extra Getriebe. Außerdem gibt es bei einen Linearmotoren kein Getriebespiel, welches ansonsten für Ungenauigkeiten sorgen könnte.

Nachteile eines Linearmotors sind hingegen das die Kraft der Motoren beschränkt ist, sollte größere Kräfte auftreten muss der Linearmotor extra dafür ausgelegt werden. Auch der Stromverbrauch kann im Vergleich zu klassischen Motoren höhere ausfallen. Das liegt daran, dass ein Linearmotor keine Selbsthemmung besitzt, das heißt wenn eine bestimmte Position angefahren wird und mit einer Last gehalten werden soll muss entweder eine Bremseinrichtung eingebaut werden oder durchgängig Strom am Motor angelegt sein (Wegener, 2008)

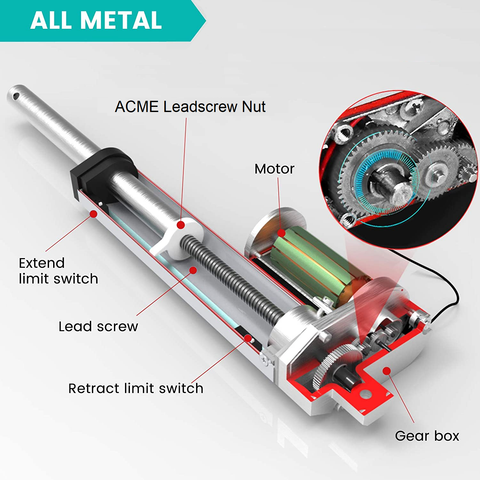


Abbildung 3: Aufbau eines Linearmotors (https://www.firgelliauto.com/en-de/blogs/actuators/linear-actuators-101)

Die Nachteile (geringer Kraft und höher Energieverbrauch) sind für die Entwicklung des Schachroboters vernachlässigbar bzw. bereits in den Anforderungen als „Out of Scope“ festgelegt worden. So ist es für das Heben von Schachfiguren nicht notwendig große Kräfte zu erzeugen, sondern vielmehr eine sehr genaue Ansteuerung der einzelnen Position zu ermöglichen.

## Schrittmotor

Schrittmotoren bestehen aus einem fixierten Stator und einen drehenden Rotor, welcher sich innerhalb des Stators befindet. Der Stator besteht aus mehreren Elektromagnetischen Spulen welches Kreisförmig um den Rotor platziert sind. Um nun eine Drehbewegung zu erzeugen, werden an die einzelnen Spulen (Siehe Abbildung: L1 bis L4) Spannung angelegt und somit ein Magnetfeld erzeugt. Dieses Magnetfeld erzeugt anschließende ein Drücken beziehungsweise ziehend des Stators wodurch dieser sich bewegt.

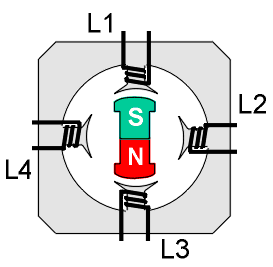


Abbildung 4: Aufbau eines Schrittmotors (https://www.xplore-dna.net/mod/page/view.php?id=678)

Je nach Anzahl der Spulen kann im Vollschrittbetrieb eine unterschiedliche Anzahl an Schritten angesteuert werden. In der Abbildung 4 können somit 4 Schritte angefahren werden.

Zur Ansteuerung eines Schrittmotors werden im Vergleich zu klassischen Motoren immer eine Motosteuerung benötigt. Diese Motorsteuerung kann den Schrittmotor sehr genau ansteuern und somit eine hohe Genauigkeit erreichen. Auch können sogenannte Mikroschritte gefahren werden, hierfür begrenzt die Motorsteuerung den Strom, welche an die einzelnen Spulen angelegt werden, dadurch sind Bewegungen von 1/5 bis 1/250 eines Vollschritts möglich.

Schrittmotoren bieten in ihrer Bewegung eine hohe Genauigkeit mit einer großen Kraft. Nachteile des Schrittmotors sind hingegen den hohen Energiebedarf und die relativ geringe Drehgeschwindigkeit (Konradin Industrie, 2024).

Für die Verwendung eines Schrittmotors innerhalb des Roboterarms sind diese Nachteile allerdings ebenfalls zu vernachlässigen, so ist der Energieverbrauch „Out of Scope“ und auch die Geschwindigkeit des Roboters ist beim Schach spielen zu vernachlässigen.

## Direkte Kinematik

# CAD-Modellierung

Die Entwicklung des Mechanischen Roboterarms wird in mehrere Teilabschnitte unterteilt. Zwei Linearmotoren sind für die groben Bewegungen des Roboterarms zuständig. Des weiterem wird ein Drehmechanismus verwendet, um den Kompletten Roboter auszurichten. Der letzte Teilabschnitt ist die Magnethalterung. Sie besteht aus zwei Aktoren, einem Linearmotor um welche Vertikal Bewegungen durchführt und ein Elektromagnet, welcher für das „Greifen“ der einzelnen Figuren zuständig ist.

## TinkerCad

Zur Modellierung der einzelnen Bauteile wird TinkerCad verwendet. TinkerCad ist eine auf Autodesk basierende Webanwendung zur CAD-Modellierung. TinkerCad wird häufig in der Ausbildung und für erste Schritte mit Autodesk verwendet. Hierfür bietet Autodesk die Anwendung kostenlos an.

Auch gibt es eine Reihe von Tutorials und eine ausführliche Dokumentation. Im Vergleich zu Autodesk bietet TinkerCad weniger Funktionen allerdings alle wichtigen für dieses Projekt.

## Erster Linearmotor

## Zweiter Linearmotor

## Drehmechanismus

## Magnethalterung

## Überblick CAD Modellierung

# Elektronik/Schaltungsaufbau/Hardware

# Figuren Erkennung

## Theoretische Grundlagen

### Schachbrett

Ein Schachbrett besteht aus 64 Feldern, die in einem 8x8-Raster angeordnet sind. Die Felder werden abwechselnd in hellen und dunklen Farben dargestellt. Zur einfacheren Identifikation der Felder wird das Brett von den Buchstaben A bis H horizontal und von den Zahlen 1 bis 8 vertikal gekennzeichnet. Die Linien werden von links nach rechts mit den Buchstaben A bis H bezeichnet und die Reihen von unten nach oben mit den Zahlen 1 bis 8. So wird beispielsweise das Feld in der untersten linken Ecke als A1 bezeichnet, das Feld rechts daneben als B1 und so weiter bis H1. Das oberste rechte Feld wäre H8. [2]

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung 5 Schachbrett mit Feldbeschriftungen

Ausführliche Notation

In der ausführlichen Notation wird das Anfangs- und das Zielfeld des Zuges aufgezeichnet, zusammen mit einer Abkürzung für die Figur. Die Abkürzungen sind: König (K), Dame (D), Turm (T), Läufer (L), Springer (S) und Bauern, die nicht gekennzeichnet werden. Ein Zug wird wie folgt notiert: Ein Bauer, der von e2 nach e4 zieht, wird als "e2-e4" notiert. Ein Springer, der von b1 nach c3 zieht, wird als "Sb1-c3" aufgezeichnet. Wenn eine Figur eine andere schlägt, wird der Bindestrich durch ein 'x' ersetzt, zum Beispiel "Ta1xa3". Für die computergestützte Analyse werden oft die Anfangsbuchstaben der englischen Namen verwendet: König (K), Dame (Q von Queen), Turm (R von Rook), Läufer (B von Bishop), Springer (N von Knight). [2]

Besondere Züge

Schachgebot wird durch ein "+" nach dem Zug angezeigt, ein Schachmatt durch ein "#".

Bauernumwandlung wird mit "=" notiert, zum Beispiel "d7-d8=D".

Die Rochade wird als "0-0" (kurze Rochade) oder "0-0-0" (lange Rochade) aufgezeichnet.

Schläge „en passant“ werden mit "ep" gekennzeichnet, zum Beispiel "d5xe6 ep".

Das Verständnis dieser Notation ist grundlegend für die Entwicklung von Algorithmen zur Erkennung von Schachzügen, da sie die Grundlage bildet, auf der die Spielzüge digital erfasst und verarbeitet werden können. [2]

### Digitale Bilder und RGB-Werte

Digitale Bildverarbeitung ist eine wesentliche Technik in vielen Forschungs- und Industrieanwendungen. Ein digitales Bild besteht aus einer rechteckigen Anordnung von Pixeln, wobei jeder Pixel eine Farbe repräsentiert. Diese Farben werden durch das RGB-Farbmodell definiert, bei dem jeder Pixel durch drei numerische Werte dargestellt wird: Rot (R), Grün (G) und Blau (B) [3].

Ein Pixel in einem digitalen Bild kann verschiedene Intensitäten der drei Farben Rot, Grün und Blau annehmen. Die Kombination dieser drei Farbwerte bestimmt die endgültige Farbe des Pixels. Zum Beispiel ergibt die Kombination (255, 0, 0) einen reinen roten Pixel, während (0, 255, 0) einen reinen grünen Pixel und (0, 0, 255) einen reinen blauen Pixel ergibt. Wenn alle drei Werte gleich sind, entsteht eine Graustufe, und wenn alle drei Werte maximal (255) sind, entsteht die Farbe Weiß [3].

Das RGB-Farbmodell ist ein additives Farbmodell, was bedeutet, dass die Farben durch das Hinzufügen von Licht in den Grundfarben Rot, Grün und Blau erzeugt werden. Dies steht im Gegensatz zu subtraktiven Modellen wie CMYK, die in Druckverfahren verwendet werden. Die Intensitäten in RGB reichen typischerweise von 0 bis 255, was 256 mögliche Werte pro Farbkanal ergibt. Daraus resultieren insgesamt 16.777.216 mögliche Farbvariationen für jeden Pixel in einem 24-Bit-Farbbild [3].

Digitale Bilder werden als zweidimensionale Arrays dargestellt, in denen jeder Eintrag (Pixel) die entsprechenden RGB-Werte enthält. Diese Struktur ermöglicht die effiziente Speicherung, Bearbeitung und Anzeige von Bildern auf digitalen Geräten. Das Verständnis dieser Grundstruktur ist essentiell für die Entwicklung und Implementierung von Algorithmen zur Bildverarbeitung und Analyse, wie sie beispielsweise bei der Erkennung von Schachzügen in diesem Projekt benötigt werden [3].

## Konzept

Das primäre Ziel des Schachroboterprojekts ist die zuverlässige Erkennung von Schachzügen durch Analyse von Veränderungen auf dem Schachbrett. Hierfür ist es essenziell, zu jedem Zeitpunkt genau zu wissen, welche Figuren sich auf welchen Feldern befinden. Dies ermöglicht es dem System, jede Veränderung im Spielkontext richtig zu interpretieren und darauf basierend Entscheidungen zu treffen.

Um dieses Ziel zu erreichen, gibt es grundsätzlich zwei Ansätze: den Einsatz eines trainierten Modells der künstlichen Intelligenz (KI), das jede Figur auf jedem Feld identifizieren kann, und eine Methode, die sich auf die Erkennung von Änderungen auf dem Brett konzentriert, ohne direkt die Art der Figur oder deren Farbe zu bestimmen. Während der erste Ansatz eine umfassende Analyse jeder einzelnen Figur erfordert, fokussiert sich der zweite Ansatz darauf, nur die Veränderungen der Felder zu erfassen.

### Methodik zur Spielzugerkennung

In diesem Projekt wird eine Methode implementiert, die ohne den Einsatz von KI auskommt. Dieser Ansatz basiert darauf, das Schachfeld zu erfassen und die 64 Schachfelder zu separieren. Zu Beginn des Spiels und vor jedem Zug sollen die Durchschnitte der RGB-Werte jedes Schachfeldes erfasst werden und als Referenzwerte gespeichert werden. Bei jeder nachfolgenden Bildaufnahme werden erneut die RGB-Werte jedes Feldes erfasst und mit den Referenzwerten verglichen und dabei die Differenz berechnet. Die Veränderungen der durchschnittlichen RGB-Werte, die zwischen den Aufnahmen festgestellt werden, zeigen an, welche Felder von einem Spielzug betroffen sind, da sich ihre durchschnittlichen RGB-Werte verändert haben. Wenn beispielsweise festgestellt wird, dass das Feld A8 und A4 einen signifikanten Unterschied in den durchschnittlichen RGB-Werten haben, leitet das System daraus ab, dass die Figur, die zu Beginn auf A8 war, nach A4 gezogen wurde. Diese Information wird im System aktualisiert, sodass zu jeder Zeit eine genaue Abbildung der Spielsituation vorliegt. Bei dieser Methodik ist zu beachten, dass nicht nur Spielzüge die RGB-Werte eines Feldes ändern, sondern auch sich ändernde Lichtverhältnisse, Schatten usw. diese Werte beeinflussen können. Die Herausforderung ist also solche Veränderungen zu erkennen oder rauszufiltern, um nur die gewollten Änderungen zu interpretieren.

Um das Konzept anhand eines Beispiels zu erklären, sollen hier zwei Schachfelder abgebildet sein. Abbildung x1 zeigt ein weißes Viereck, das ein weißes Schachfeld repräsentieren soll, das sich innerhalb einer schwarzen Umrandung befindet. Abbildung x2 zeigt dasselbe weiße Schachfeld mit einem grauen Viereck darin. Das graue Viereck repräsentiert hier eine Schachfigur. So könnte eine ideale Aufnahme von oben eines weißen Schachfelds mit grauer Figur aussehen. Die Maße der Vierecke sind so gewählt, dass der graue Flächeninhalt genau so groß ist wie der weiße Flächeninhalt.

Ein Bild, das Rechteck, Screenshot, Quadrat, Design enthält.

Automatisch generierte BeschreibungEin Bild, das Rechteck, weiß, Rahmen enthält.

Automatisch generierte BeschreibungZu Beginn des Spiels werden die Referenzwerte des leeren Schachfelds berechnet und gespeichert. In diesem idealen Fall haben alle Pixel des weißen Schachfelds die RGB-Werte (255, 255, 255), wodurch sich ein Durchschnitt für das ganze Feld von (255, 255, 255) ergibt. Abbildung x2 soll nun das gleiche Schachfeld bei der zweiten Aufnahme repräsentieren. Hier werden die aktuellen RGB-Werte des Schachfelds zusammengerechnet. Jeder Pixel der grauen Fläche hat die RGB-Werte (100, 100, 100), und da die Fläche gleich groß wie die weiße Fläche ist, ergibt sich für das Feld ein Durchschnittswert von (100+255)/2 = 177,5 für jeden Farbkanal. Die Differenz für jeden Kanal wäre dann 255 - 177,5 = 77,5. Diese Information über die Farbänderung eines

Feldes soll dann genutzt werden, um den gespielten Zug zu erkennen. Um die Zuverlässigkeit der Bilderkennung unter verschiedenen Lichtverhältnissen und anderen variablen Umgebungsbedingungen zu gewährleisten, wird zusätzlich die Standardabweichung der RGB-Werte berechnet.

### Bestimmung des gespielten Zuges

Die zwei Felder mit den größten kombinierten Werten aus RGB-Differenzen werden als die Felder identifiziert, auf denen eine Veränderung stattgefunden hat. Dies deutet typischerweise darauf hin, dass eine Schachfigur bewegt wurde: Ein Feld zeigt, wo die Figur zuvor stand, und das andere Feld zeigt die neue Position der Figur.

Sinnhaftigkeit und Grenzen des Verfahrens

Jedoch hat dieser Ansatz auch Schwächen. Eine Herausforderung ist die Empfindlichkeit gegenüber Veränderungen in der Umgebungsbeleuchtung, die trotz der Korrekturen durch Standardabweichung die RGB-Werte beeinflussen können. Ebenso können komplexe Züge, bei denen mehr als eine Figur innerhalb eines Zuges bewegt wird (wie bei einer Rochade), schwieriger zu identifizieren sein, da mehrere Felder betroffen sind und die Methode primär die zwei größten Veränderungen erfasst. Zusätzliche Algorithmen oder manuelle Überprüfungen könnten erforderlich sein, um solche speziellen Spielzüge korrekt zu interpretieren.

### Methodik zum Bilderfassungszeitpunkt

Eine Herausforderung in der Bilderkennung für diesen Schachroboter ist die Bestimmung des optimalen Zeitpunkts für die Aufnahme eines neuen Bildes, um die Veränderungen auf dem Schachbrett zu erfassen. Die Genauigkeit der Zugerkennung hängt maßgeblich davon ab, dass die Bilder genau dann aufgenommen werden, wenn ein Zug abgeschlossen ist. Es gibt mehrere Ansätze, um diesen kritischen Moment zu bestimmen:

**Analyse von Bildveränderungen:** Eine Methode besteht darin, die Bilder kontinuierlich zu analysieren und nach einer Phase ohne signifikante Änderungen im Bild zu suchen. Das Ende einer solchen Phase kann darauf hinweisen, dass ein Zug beendet wurde und das Brett stabil ist, was ein idealer Zeitpunkt für die Aufnahme eines neuen Bildes ist.

**Einsatz von maschinellem Lernen:** Durch maschinelles Lernen kann ein Modell trainiert werden, das spezifische Ereignisse wie das Erscheinen oder Verschwinden einer Hand über dem Brett erkennt. Dies kann genutzt werden, um das Ende eines Zuges zu identifizieren, wenn die Hand das Schachbrett verlässt.

**Manuelle Bestätigung:** Die einfachste, jedoch benutzerabhängige Methode ist die manuelle Bestätigung eines Zuges. Der Benutzer kann zum Beispiel durch Drücken der Enter-Taste auf einem Laptop signalisieren, dass ein Zug abgeschlossen ist und ein neues Bild aufgenommen werden soll.

Jeder dieser Ansätze hat seine eigenen Vor- und Nachteile in Bezug auf Genauigkeit, Benutzerfreundlichkeit und technische Umsetzbarkeit. Die Wahl der geeigneten Methode hängt von den spezifischen Anforderungen des Projekts und den verfügbaren Ressourcen ab.

## Umsetzung

### Hardware

Im Rahmen der Entwicklung des Schachroboters spielt die Bilderkennung eine entscheidende Rolle. Hierfür wird die Swissonic Webcam 2 Full HD AF eingesetzt, die mit einer Auflösung von 1920x1080 Pixeln und einer Übertragungsrate von 30 Bildern pro Sekunde für die präzise Erfassung der Schachfiguren und deren Positionen geeignet ist. Die Kamera verfügt über ein integriertes Stereomikrofon, das klare Tonübertragungen während Webkonferenzen ermöglicht, jedoch in diesem Projekt primär zur Erfassung von Umgebungsgeräuschen dient.

Die Verbindung der Kamera zum Laptop erfolgt über einen USB 2.0 Anschluss, der eine schnelle und zuverlässige Übertragung der Bilddaten gewährleistet. Der Laptop fungiert als Verarbeitungseinheit und ist mit einer spezialisierten Bilderkennungssoftware ausgestattet, die die Daten in Echtzeit verarbeitet. Dies ermöglicht eine effiziente Analyse der Spielsituation und eine schnelle Reaktion des Roboters. Die technische Konfiguration ist speziell auf die Anforderungen des Schachroboters abgestimmt, um optimale Ergebnisse bei der Bilderkennung zu erzielen.

### Software-Implementierung

Die Software-Implementierung wurde mit der Programmiersprache Python durchgeführt. Python ist gut für die Bildverarbeitung geeignet, da es eine breite Unterstützung durch Bibliotheken bietet und es ermöglicht, schnell Prototypen zu erstellen. Diese Eigenschaft ist besonders nützlich, da das entwickelte Konzept umfangreich getestet werden muss [4]. Für die Bildverarbeitung wurde die OpenCV-Bibliothek (cv2) verwendet.

Es gibt eine zentrale Komponente, die Main.py-Datei. Diese Datei führt die Hauptlogik des Programms aus und ruft dabei Funktionen aus den Dateien camera.py und chess.py auf. Diese Dateien stellen spezifische Funktionen bereit, die einzelne Aufgaben erledigen, wie zum Beispiel das Aufnehmen von Bildern oder das Erkennen des Schachbretts.

* Main.py: Diese Datei enthält die Hauptlogik des Programms und koordiniert die Abläufe. Sie ruft Funktionen aus den anderen Modulen auf und verarbeitet die Daten, um die gewünschten Ergebnisse zu erzielen.
* camera.py: Diese Datei enthält Funktionen für die Interaktion mit der Kamera, einschließlich der Aufnahme von Bildern.
* chess.py: Diese Datei enthält Funktionen zur Erkennung und Verarbeitung des Schachbretts und der Schachfiguren.

Durch die modulare Struktur der Software wird eine klare Trennung der Aufgaben ermöglicht, was die Wartung und Erweiterung des Programms erleichtert.

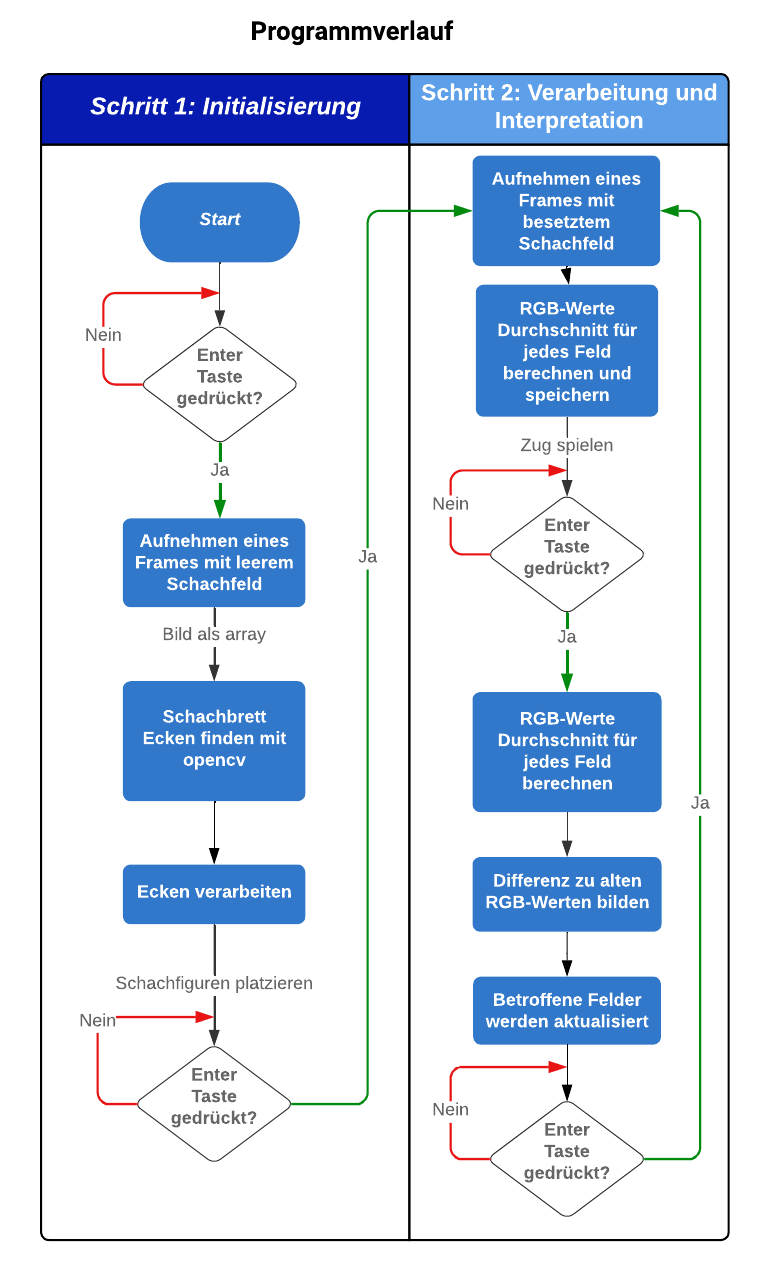


Abbildung x zeigt den Ablauf des Programms. Beim Start des Programms befindet es sich in einem Wartezustand und wartet in einer Schleife auf die Eingabe des Benutzers durch das Drücken der Enter-Taste. Bei Betätigung der Enter-Taste macht die Kamera eine Bildaufnahme des Schachbretts (auch Frame genannt). Diese Aufnahme erfolgt zunächst ohne Schachfiguren, da OpenCV das Schachbrett nicht erkennen kann, wenn Figuren die Kanten und Ecken der Felder verdecken. Das aufgenommene Bild wird dann an die OpenCV-Bibliothek übergeben, die eine Funktion zur Erkennung eines 8x8-Schachbretts aufruft. Diese Funktion gibt bei Erfolg die inneren 49 Ecken des Schachbretts zurück, wie in Abbildung x gezeigt, und liefert ein Array mit den Koordinaten dieser Ecken.

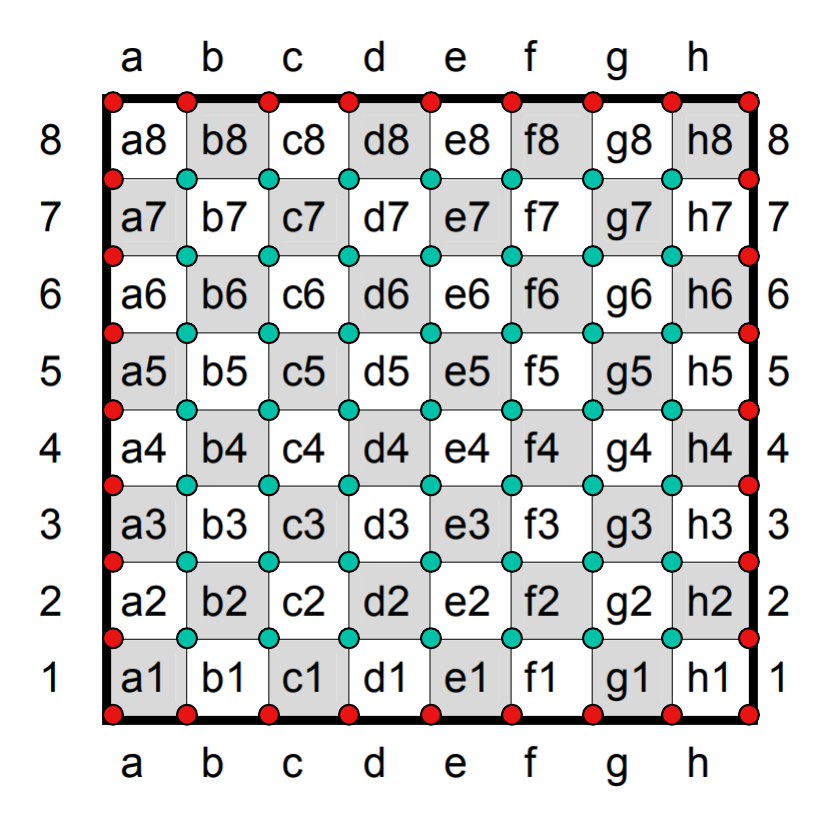
Im nächsten Schritt werden die Koordinaten der äußeren Ecken durch Extrapolation berechnet. Durch die Bestimmung der horizontalen und vertikalen Abstände zwischen den Linien können die äußeren Punkte abgeschätzt und dem Array der Ecken hinzugefügt werden. Die durch OpenCV gefundenen Punkte und die durch Extrapolation geschätzten Punkten sind in einem Beispiel hier visualisiert.

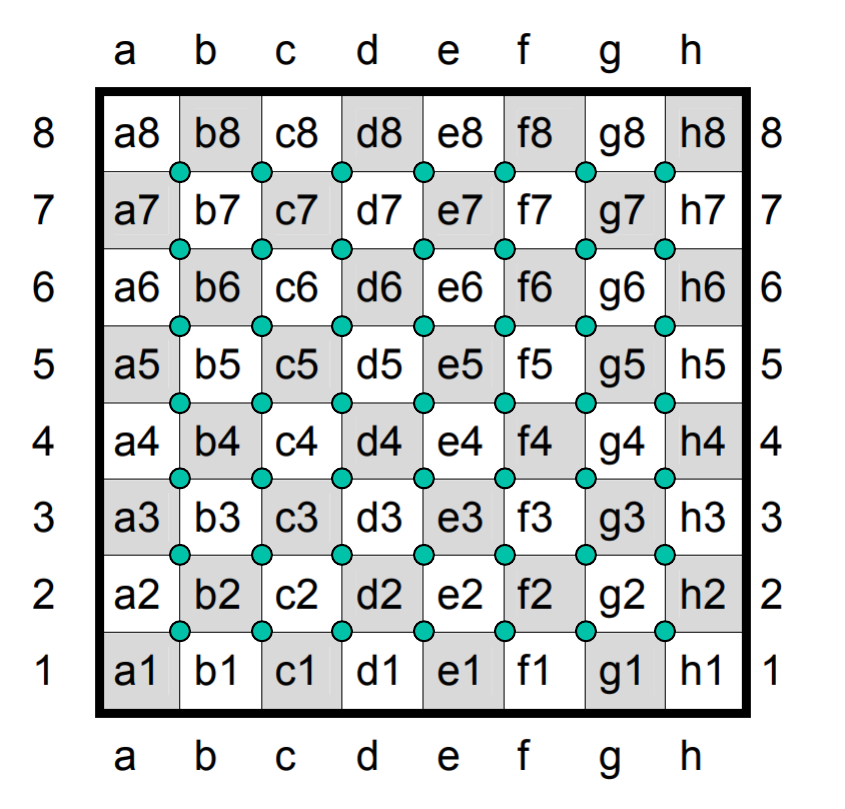
Ein Bild, das Rechteck, Quadrat, Muster, Kunst enthält.

Automatisch generierte Beschreibung

Anschließend wird der Benutzer aufgefordert, die Schachfiguren in der klassischen Ausgangsstellung auf dem Brett zu platzieren. Nach der Platzierung der Figuren drückt der Benutzer erneut die Enter-Taste, um fortzufahren. Ein weiteres Bild des Schachbretts wird aufgenommen. Basierend auf den bekannten Positionen der Ecken wird jedes Schachfeld einzeln betrachtet, und die durchschnittlichen RGB-Werte für jedes Feld werden berechnet und als Referenzwerte gespeichert.

Der Benutzer führt nun einen Schachzug durch und drückt erneut die Enter-Taste. Ein neues Bild des Schachbretts wird aufgenommen, und die aktuellen RGB-Werte der Felder werden berechnet. Diese Werte werden mit den Referenzwerten der vorherigen Position verglichen, um die Differenzen zu ermitteln. Für jedes Feld wird die Änderung der RGB-Werte berechnet, und die zwei Felder mit den größten Veränderungen werden identifiziert. Diese Felder werden als die vom Zug betroffenen Felder interpretiert. Das Schachbrett-Objekt, das alle Schachfelder speichert, wird nun aktualisiert. Das Feld, auf dem zuvor eine Figur stand, wird als leer markiert, da die Figur von diesem Feld wegbewegt wurde. Das Ziel-Feld, auf das die Figur gezogen wurde, wird nun als besetzt markiert und erhält die Figur, die vorher auf dem Ausgangsfeld stand.





## Tests und Probleme

Bei den Tests zur Erkennung von Schachzügen wurden zunächst weiße und schwarze Figuren verwendet, um zu testen, ob die richtigen Feldern, die von einem Zug betroffen sind, erkannt werden. Es wurden zufällige Züge (aus Plausibilitätsgrunden auch nicht legale Züge) gemacht. Es stellte sich heraus, dass fälschlicherweise andere Züge erkannt wurden, da die Veränderungen auf anderen Feldern größer waren als auf den Feldern, auf denen tatsächlich Figuren gezogen wurden.

Dies lag daran, dass weiße Figuren, die auf weiße Felder ziehen, die Farbwerte des Feldes kaum verändern, da das Feld weiterhin hauptsächlich weiß bleibt. Dadurch wurden Veränderungen durch Schatten oder Lichtveränderungen auf anderen Feldern als signifikant größer erkannt.

Ein weiteres Problem, das sich bei den Tests herausstellte, betraf die Erkennung von Zügen, die innerhalb der Linien b und g sowie der Reihen 2 und 7 ausgeführt wurden. In diesen Fällen wurden die Züge nicht korrekt erkannt. Stattdessen wurden Felder als verändert wahrgenommen, die eine Linie dahinter lagen. Beispielsweise wurde bei einem Zug auf der Linie 2 fälschlicherweise eine Veränderung auf der Linie 1 registriert, und bei Zügen auf der Linie b wurde eine Veränderung auf der Linie a erkannt.

Dieses Problem trat auf, weil die Figuren aufgrund ihrer Höhe und dem Winkel der Kamera einen Teil des benachbarten Feldes auf der anderen Linie überdeckten und eventuell einen Schatten auf dieses warfen. Diese Überdeckung beeinflusste die RGB-Werte des benachbarten Feldes, was dazu führte, dass das System fälschlicherweise annahm, dass auf diesem Feld eine Veränderung stattgefunden hatte. Besonders problematisch war dies, wenn das gezogene Feld ein schwarzes Feld war, da hinter dem schwarzen Feld immer ein weißes Feld lag. In solchen Fällen führte die Überdeckung und der Schatten dazu, dass die RGB-Werte des weißen Feldes stärker verändert wurden als die des schwarzen Feldes. Zusätzlich war die obere Fläche der Figur meist heller als die anderen Seiten. Aufgrund des Kamerawinkels befand sich diese helle obere Fläche teilweise auf dem benachbarten Feld, was die RGB-Werte dieses Feldes ebenfalls beeinflusste und zu weiteren Fehlinterpretationen führte.

Hier ein Beispiel: Eine Figur wird von b5 nach b6 gezogen, wie im zweiten Bild zu sehen ist. Dabei fällt auf, dass die Figur im zweiten Bild auf das benachbarte weiße Feld ragt und dieses teilweise überdeckt. Zudem wird ein Schatten auf dieses Feld geworfen, was die Farbwerte zusätzlich verfälscht. Diese Veränderungen sind auch in den RGB-Werten erkennbar. Bei diesem Zug wurden die Farbwertänderungen wie folgt gemessen: Auf dem Startfeld b5 eine Differenz von 126, auf dem Zielfeld b6 eine Differenz von 46 und auf dem benachbarten Feld a6 eine Differenz von 52. Aufgrund dieser Messwerte wurde der Zug fälschlicherweise als Bewegung von b5 nach a6 interpretiert.Ein Bild, das Rechteck, Quadrat, Kunst, Muster enthält.

Automatisch generierte Beschreibung Ein Bild, das Rechteck, Quadrat, Kunst, Muster enthält.

Automatisch generierte Beschreibung

B5: 126, a6: 52 b6: 46

Diese Fehlinterpretationen zeigen, dass die Position und der Winkel der Kamera sowie die Größe der Figuren eine entscheidende Rolle bei der Genauigkeit der Bilderkennung spielen. Um dieses Problem zu beheben, müssen möglicherweise Anpassungen an der Kameraposition oder die Implementierung zusätzlicher Algorithmen zur besseren Unterscheidung von tatsächlichen Bewegungen und Überdeckungen vorgenommen werden.

## Verbesserungen

Die erste und schnellste Verbesserung bestand darin, Figuren in anderen Farben zu verwenden, die sich deutlich von den RGB-Werten der schwarzen und weißen Felder unterscheiden. Zur Auswahl standen blaue, rote und grüne Figuren.

Bei Tests stellte sich heraus, dass die blauen Figuren zu dunkel waren und sich nicht signifikant genug von den schwarzen Feldern abhoben, was zu häufigen Fehlinterpretationen führte.

Ein Bild, das Rechteck, Quadrat, Kunst, Muster enthält.

Automatisch generierte BeschreibungEin Bild, das Rechteck, Quadrat, Kunst, Muster enthält.

Automatisch generierte Beschreibung

Im ersten Bild ist eine blaue Figur auf dem Feld h5 zu sehen, die im zweiten Bild auf das Feld c7 gezogen wird. Während im ersten Bild die Figur auf h5 gut erkennbar ist, da sie sich deutlich vom weißen Feld abhebt, zeigt das zweite Bild, dass die Figur auf c7 kaum vom schwarzen Feld zu unterscheiden ist. Dies wird durch die Farbwertdifferenzen bestätigt: auf h5 wurde eine Differenz von 137 gemessen, auf c7 jedoch nur eine Differenz von 21. Auf dem benachbarten Feld h6 wurde eine Differenz von 24 festgestellt, was zu einer Fehlinterpretation führte. Im Gegensatz dazu boten die roten und grünen Figuren einen ausreichenden Farbkontrast sowohl zu den weißen als auch zu den schwarzen Feldern. Aufgrund ihrer besseren Unterscheidbarkeit wurden daher rote und grüne Figuren für das Projekt ausgewählt.

Um das Problem mit der Zugerkennung auf den Linien b und g sowie den Reihen 2 und 7 zu beheben, wurden mehrere Maßnahmen ergriffen:

Zunächst wurden die Maße der Figuren verkleinert. Dies führte dazu, dass die Figuren weniger Platz auf den Feldern einnahmen und je nach menschlicher Platzierung weiter von den benachbarten Feldern entfernt waren. Die geringere Höhe der Figuren reduzierte auch den verdeckten Bereich aufgrund des Kamerawinkels und

Ein Bild, das Rechteck, Quadrat, Farbigkeit, Im Haus enthält.

Automatisch generierte Beschreibung

geometrischer Faktoren. Diese Änderung verringerte den verdeckten Bereich und den Schatten, den die Figuren werfen, was die Beeinflussung benachbarter Felder minimierte.

Eine weitere Maßnahme war die Anpassung des Algorithmus. Anstatt nur die durchschnittlichen RGB-Werte zu messen und zu berechnen, wurde zusätzlich die Standardabweichung der Farbwerte für jeden Kanal berechnet. Die Idee dahinter ist, dass ein weißes oder schwarzes Feld eine einheitliche Farbe hat und daher idealerweise jeder Pixel eines weißen Feldes die Werte (255, 255, 255) aufweist. Berechnet man hier die Standardabweichung für den roten Kanal, erhält man durch die Formel mit beispielsweise 100 Pixeln:

und

daraus folgt:

und somit, .

Wenn also alle Werte gleich sind, entspricht der Mittelwert jedem einzelnen Wert. Da es keine Abweichungen vom Mittelwert gibt, ist die Standardabweichung 0.

Im folgenden Beispiel besteht ein Feld aus 100 Pixeln, wobei eine Figur die Hälfte der Feldfläche einnimmt. Somit haben 50 Pixel durch die Figur den Rotkanalwert von 100 und 50 Pixel durch das weiße Feld einen Wert von 255. Ist eine Figur auf dem Feld, beträgt die Standardabweichung der roten Werte nicht 0, sondern:

Daraus folgt:

und somit, .

Wenn also eine Figur auf dem weißen Feld ist, hat dieses Feld eine Standardabweichung von 77.5 statt 0. Diese Information, beziehungsweise die Differenz der Standardabweichungen vor und nach einem Zug, kann zusätzlich verwendet werden, um zu erkennen, welches Feld durch einen Zug betroffen ist.

In der Praxis wird jedoch nicht jeder Pixel die Werte 255 haben, und somit wird die Standardabweichung ungleich 0 sein, wobei sich diese auch mit verschiedenen Lichtverhältnissen ändert. Dadurch wird auch auf Feldern, die nicht von einem Zug betroffen sind, eine Differenz in der Standardabweichung zu erkennen sein. Die Idee hierbei ist, dass zwischen zwei Aufnahmen die RGB-Werte eines Feldes sich durch zum Beispiel ändernde Lichtintensität gleichmäßig verändern, wodurch sich der Durchschnitt der RGB-Werte ändert, die Standardabweichung jedoch weitgehend gleich bleibt, da die Werte sich um den veränderten Mittelwert streuen. Haben zum Beispiel die Pixel in einem weißen Feld mit 100 Pixeln die Werte für den Rotkanal von 50 Mal 220 und 50 Mal 240, ergibt sich eine Standardabweichung von 10. Ändern sich alle Werte um -10, also 50 Mal 210 und 50 Mal 230, ändert sich der RGB-Durchschnitt um 10, jedoch bleibt die Standardabweichung bei 10 und somit gleich wie zuvor. Dies trägt zur richtigen Erkennung des richtigen Feldes bei.

Diese Verbesserung im Algorithmus kann jedoch durch die vorher genannte Anpassung eingeschränkt werden, bei der nur der mittlere Teil eines Feldes betrachtet wird. Wird der betrachtete Bereich zu klein gewählt, kann es dazu kommen, dass nur noch der Bereich betrachtet wird, wo die Figur zu sehen ist und kein Teil mehr vom weißen Feld. Betrachtet man die Standardabweichung von einem zuvor weißen unbesetzten Feld und einem danach besetzten Feld oder umgekehrt, ist die Standardabweichung in beiden Fällen 0, da jeder Wert gleich ist. Die Differenz ist somit auch 0 und trägt nicht zur Erkennung der betroffenen Felder bei. Es ist also wichtig, den Bereich möglichst klein zu wählen, jedoch nicht so klein, dass der Bereich eine ganze Figur abdecken kann.Ein Bild, das Rechteck, Quadrat, Reihe, Bilderrahmen enthält.

Automatisch generierte BeschreibungEin Bild, das Rechteck, Screenshot, Quadrat, Design enthält.

Automatisch generierte Beschreibung

Ein Bild, das Rechteck, Screenshot, Quadrat, Design enthält.

Automatisch generierte BeschreibungEin Bild, das Rechteck, weiß, Rahmen enthält.

Automatisch generierte Beschreibung

Die Standardabweichung ist ein Maß für die Streuung oder Verteilung der Werte eine Zufallsvariable um ihren Erwartungswert. Sie wird häufig in der Statistik verwendet, um die Genauigkeit und Zuverlässigkeit von Daten zu beurteilen. Die Berechnung der Standardabweichung basiert auf der Varianz, welche die durchschnittliche quadrierte Abweichung vom Erwartungswert darstellt. [5]

Die Varianz eine Zufallsvariable ist definiert als der Erwartungswert der quadrierten Abweichung vom Erwartungswert und kann berechnet werden durch:

[5]

Die Standardabweichung ist die Quadratwurzel der Varianz:

[5]

Eine weitere sehr effektive Maßnahme war die Anpassung des Algorithmus. Es wurde festgestellt, dass benachbarte Felder häufig durch die Überdeckung oder den Schatten einer Figur beeinflusst wurden, insbesondere an den Rändern eines Schachfeldes. Daher wurde der Bereich eines Schachfeldes, in dem die RGB-Werte berechnet werden, verkleinert, indem nur die inneren 80% eines Feldes berücksichtigt wurden. Das Innere eines Feldes wird immer durch das Ziehen einer Figur auf dieses Feld beeinflusst, während Schatten oder Überdeckungen das Innere eines Feldes viel unwahrscheinlicher beeinflussen oder gar nicht beeinflussen.

Ein Bild, das Entwurf, Reihe, Design, Diagramm enthält.

Automatisch generierte BeschreibungDiese Anpassung erhöhte die Genauigkeit der Erkennung erheblich und eliminierte die ständigen Fehlinterpretationen auf den Linien 2 und 7 sowie den Reihen b und g, wo die meisten Überdeckungen und Schatten aufgetreten waren.Ein Bild, das Reihe, Design enthält.

Automatisch generierte Beschreibung

## Ergebnis

Das Ergebnis ist eine funktionierende Bilderkennung von Zügen mit einer Genauigkeit von etwa 95%. Zur Erkennung des Schachbretts wird OpenCV verwendet. Dabei muss das Schachbrett zunächst leer sein, damit die Ecken erkannt werden können. Dies stellt einen zusätzlichen Schritt dar. Alternativ könnten die Eckpunktkoordinaten fest eingebaut werden, da das Schachbrett eine feste Position hat. Allerdings könnte sich die Kamera leicht verschieben, weshalb es sinnvoll ist, die Ecken jedes Mal neu zu erkennen.

Die inneren 49 Eckpunkte werden von OpenCV mit hoher Genauigkeit erkannt, während die äußeren Eckpunkte extrapoliert werden und daher nicht so präzise sind. So liegen die oberen Eckpunkte leicht unter den tatsächlichen Kanten, wie hier zu sehen.

Ein Bild, das Rechteck, Quadrat, Kunst, Wand enthält.

Automatisch generierte Beschreibung

Diese Ungenauigkeit hat jedoch keinen Einfluss auf die Funktionsweise, da sie nur eine geringe Verschiebung darstellt und die RGB-Werte nur aus einem Bereich in der Mitte erfasst werden.

Die Interaktion durch den Benutzer erfolgt durch das Drücken der Entertaste. Zum Beispiel wird die Entertaste gedrückt, um das Programm darüber zu informieren, dass der nächste Zug gemacht wurde.

# Zuggenerierung und -ausführung

## Konzept

Um auf die Züge des menschlichen Gegners effektiv reagieren zu können, muss der Schachroboter in der Lage sein, eigenständig Züge zu generieren. Die Entwicklung eines eigenen Schachalgorithmus wäre jedoch eine komplexe und ressourcenintensive Aufgabe. Daher greifen wir auf bestehende Schachengines zurück, die bereits hochentwickelte Algorithmen zur Zugbestimmung implementieren.

Unter den vielen verfügbaren Schachengines haben wir Stockfish ausgewählt, eine der stärksten und am weitesten verbreiteten Open-Source-Schachengines. Stockfish ist bekannt für seine hohe Spielstärke und Effizienz, was es zu einer idealen Wahl für unser Projekt macht. Eine vergleichende Analyse verschiedener Schachengines hat gezeigt, dass Stockfish in verschiedenen Testsituationen durchgehend hoch bewertet wird, insbesondere im Hinblick auf CPU- und Speichereffizienz sowie Gewinnraten gegenüber anderen Top-Engines wie Rybka und Komodo (Sojka, 2022).

In diesem Projekt wird die Stockfish-Schach-Engine verwendet, um basierend auf einem vom Benutzer gewählten Schwierigkeitsgrad einen Zug zu generieren. Der Schwierigkeitsgrad bestimmt dabei die Parameter "depth" (Suchtiefe) und "thinking time" (Denkzeit) der Engine. Sobald Stockfish den besten Zug berechnet hat, wird dieser an einen Arduino weitergegeben. Der Arduino steuert die Motoren des Schachroboters, um den von Stockfish berechneten Zug auf dem Schachbrett auszuführen. Dabei gibt es drei relevante Szenarien:

1. **Zug zu einem unbesetzten Feld:** Der Schachroboter fährt zum Startfeld, nimmt die Figur auf und legt sie auf dem Zielfeld ab.
2. **Zug zu einem besetzten Feld:** Der Schachroboter nimmt zuerst die Figur auf dem Zielfeld auf und entfernt sie vom Schachbrett. Anschließend bewegt er die Figur vom Startfeld zum Zielfeld.
3. **Rochade:** Bei der Rochade müssen zwei Figuren (der König und der Turm) bewegt werden. Der Roboter fährt zuerst das Startfeld des Königs an, nimmt ihn auf und legt ihn auf dem entsprechenden Zielfeld ab. Anschließend wird der Turm vom Start- zum Zielfeld bewegt.

**Die Steuerung der Motorbewegungen des Schachroboters erfolgt durch das Anlegen einer Spannung an die Motoren. Diese Spannung bewirkt, dass die Motoren mit einer konstanten Geschwindigkeit arbeiten. Die Länge der Zeit, in der die Spannung angelegt wird, bestimmt die zurückgelegte Strecke der Linearmotoren. Um also den Schachroboter zu einem bestimmten Feld auf dem Schachbrett zu bewegen, muss die genaue Dauer ermittelt werden, für die die Motoren aktiviert bleiben. Es wird eine Ausgangsposition festgelegt, die als Referenzposition dienen soll. Durch experimentelles Bestimmen dieser Zeitdauer kann sichergestellt werden, dass der Roboter präzise die gewünschten Felder anfährt.**

Sobald der Schachroboter alle Bewegungen abgeschlossen und den Zug ausgeführt hat, kann das Python-Programm weiterlaufen.

## Umsetzung

Die Integration von Stockfish in das System erfolgt in mehreren Schritten:

Installation und Einrichtung: Stockfish wird auf demselben Computer installiert, der auch die Bilderkennungssoftware ausführt. Dies ermöglicht eine nahtlose Kommunikation zwischen den beiden Systemkomponenten.

Kommunikation über UCI (Universal Chess Interface): Stockfish unterstützt das UCI-Protokoll, das eine standardisierte Methode bietet, um Züge zu senden und Empfehlungen von der Engine zu erhalten. Unser System verwendet UCI, um die aktuelle Stellung nach jedem erkannten menschlichen Zug zu übermitteln und den optimalen Zug von Stockfish zu empfangen.

Ein Bild, das Text, Screenshot, Diagramm, parallel enthält.

Automatisch generierte Beschreibung

Der Programmablauf wurde erweitert, um die Initialisierung von Stockfish und die Zuggenerierung sowie -ausführung zu berücksichtigen.

Zu Beginn des Programms wird die Stockfish-Schach-Engine initialisiert, wie in Schritt 1 ergänzt. Die Engine wird mit den entsprechenden Parametern wie Suchtiefe und Denkzeit konfiguriert. Nachdem der Mensch einen Zug gemacht hat, werden die betroffenen Felder identifiziert. Der Zug wird dann an Stockfish übermittelt, um die interne Schachstellung zu aktualisieren. Stockfish wird nun nach einem Zugvorschlag gefragt. Dieser Zug wird vom Programm abgerufen und für die Ausführung vorbereitet.

Das Arduino-Programm wird ausgeführt, um den von Stockfish vorgeschlagenen Zug umzusetzen. Der Zug und die betroffene Figur werden an das Arduino-Board übergeben. Das Arduino-Board hält eine Tabelle mit den Schachfeldern und den jeweiligen Motor-Parametern, um von der Ausgangsposition (Nullpunkt) das Zielfeld präzise zu erreichen.

Abhängig vom Typ des Zugs erfolgt die Bewegungslogik: Bei einem normalen Zug fährt der Schachroboter zum Startfeld, aktiviert den Elektromagneten, um die Figur aufzunehmen, kehrt zur Ausgangsposition zurück, fährt dann zum Zielfeld, deaktiviert den Elektromagneten, um die Figur abzulegen, und kehrt schließlich wieder zur Ausgangsposition zurück. Bei einem Schlagzug fährt der Schachroboter zuerst zum Zielfeld, aktiviert den Elektromagneten, um die gegnerische Figur aufzunehmen, kehrt zur Ausgangsposition zurück, lässt die gegnerische Figur dort fallen und führt dann die Bewegungen wie bei einem normalen Zug aus. Bei der Rochade führt der Schachroboter die Bewegungen des normalen Zugs zuerst mit dem König aus und anschließend mit dem Turm. Beide Figuren werden nacheinander bewegt, wie im normalen Zug beschrieben.

Nach der Ausführung des Zugs kehrt der Schachroboter zur Ausgangsposition zurück, um der Kamera nicht im Weg zu stehen und bereit für den nächsten Schritt zu sein. Der Programmablauf beginnt erneut, mit der Berechnung und Speicherung neuer RGB-Referenzwerte.

Ansteuerung des Roboters

# Herausforderungen

## Heben aus engen Stellen heraus

## Figurenerkennung

# Fazit

## Kritische Reflektion

## Ausblick

# Literaturverzeichnis

Konradin Industrie. (5 2024). *Was sind Schrittmotoren, welche Typen gibt es und wie funktionieren sie?* Von https://kem.industrie.de/elektromotoren/was-sind-schrittmotoren-welche-typen-gibt-es-und-wie-funktionieren-sie/#1 abgerufen

Krause, M. (2021). *Baubetriebliche Optimierung des vollwandigen Beton-3D-Drucks.* Springer Vieweg Wiesbaden.

O'Connell, J. (6. August 2021). ABS vs PLA (3D-Drucker-Filament): Die Unterschiede. *ALL3DP*.

Wegener, R. (16. 10 2008). Dissertation. *Zylindrischer Linearmotor mit konzentrierten Wicklungen für hohe Kräfte*. Dortmund.

[1] Kyrychok, T.Y., Kyrychok, P.A. & Parkhomenko, A.V. Image evaluation procedure based on the average color deviation. *Radioelectron.Commun.Syst.* **57**, 175–179 (2014). <https://doi.org/10.3103/S0735272714040049>

[2]  [Hartmut Riedel (2005), "Das Aufschreiben von Schachpartien" notation.pdf (kantschule-falkensee.de)](https://www.kantschule-falkensee.de/kant2003/ags/schach/training/allg/notation.pdf)

[3] Alala, B., Mwangi, W., & Okeyo, G. (2014). Image Representation using RGB Color Space. International Journal of Innovative Research and Development

[4] [Which language is more advantageous to use in image processing? | by Emrecan Parlar | Medium](https://medium.com/@eparlar/which-language-is-more-advantageous-to-use-in-image-processing-45e29577b6b4#:~:text=Image%20processing%20is%20a%20widely,languages%20used%20for%20this%20purpose.)

[5] Barot, M., Hromkovič, J. (2020). Standardabweichung. In: Stochastik 2. Grundstudium Mathematik. Birkhäuser, Cham. https://doi.org/10.1007/978-3-030-45553-8\_2

Kapitel:

Ergebniss/diskussion/kritik von figuren erkennung und zuggenerierung und ausführung

Theoretische grundlagen ausbauen