

BACHELOR THESIS
Sebastian Berghoff

Computer-Vision-Techniken zur Analyse von physischen Brettspielen auf mobilen Geräten: eine Fallstudie am Beispiel von Go

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Sebastian Berghoff

Computer-Vision-Techniken zur Analyse von
physischen Brettspielen auf mobilen Geräten: eine
Fallstudie am Beispiel von Go

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Philipp Jenke
Zweitgutachter: Prof. Dr. Tim Tiedemann

Eingereicht am: 1. Februar 2024

Sebastian Berghoff

Thema der Arbeit

Computer-Vision-Techniken zur Analyse von physischen Brettspielen auf mobilen Geräten: eine Fallstudie am Beispiel von Go

Stichworte

Computer Vision, Mobile Computing, Spielanalyse, Go, Augmented Reality

Kurzzusammenfassung

Diese Arbeit untersucht die Entwicklung eines mobilen Anwendungssystems, welches durch den Einsatz von Computer-Vision-Techniken die Analyse physischer Brettspiele ermöglicht, dargestellt am Beispiel des Spiels Go. Das Hauptziel ist es, ein intuitives und effizientes System zu schaffen, das in Echtzeit Spielzustände erfasst und analysiert, ohne die Notwendigkeit einer manuellen Digitalisierung. Die Arbeit zeigt, wie diese Technologien das Analyseerlebnis verbessern und das Lernen im Spiel unterstützen. Durch die Einbindung innovativer Erkennungstechniken und der Verwendung von Augmented Reality trägt die Thesis dazu bei, neue Perspektiven für die Analyse physischer Brettspiele in einer digitalen Welt zu eröffnen.

Sebastian Berghoff

Title of Thesis

Computer Vision Techniques for Analyzing Physical Board Games on Mobile Devices: A Case Study of Go

Keywords

Computer Vision, Mobile Computing, Game Analysis, Go, Augmented Reality

Abstract

This thesis investigates the development of a mobile application system that enables the analysis of physical board games through the use of computer vision techniques, exemplified by the game Go. The main goal is to create an intuitive and efficient system that captures and analyzes game states in real time without the need for manual digitization. The work shows how these technologies improve the analysis experience and support in-game learning. By incorporating innovative recognition techniques and the use of augmented reality, the thesis contributes to opening new perspectives for the analysis of physical board games in a digital world.

Inhaltsverzeichnis

Abbildungsverzeichnis	ix
Tabellenverzeichnis	xii
1 Einleitung	1
2 Grundlagen	3
2.1 Einführung in Computer Vision	3
2.2 Einführung in Go	4
2.3 Neuronale Netzwerke zur Spielanalyse	5
2.4 Augmented Reality (AR) in mobilen Anwendungen	7
2.5 Grundlagen von Web-APIs	8
3 Verwandte Arbeiten	9
3.1 Augmented Reality for Board Games	9
3.2 Fast radial symmetry for detecting points of interest	10
3.3 Model Improves a Weak Classifier	10
3.4 Automatic extraction of game record from TV Baduk program	11
3.5 Extracting go game positions from photographs	11
3.6 Detection of Go-board contour in real image using genetic algorithm	12
3.7 Automatic extraction of go game positions from images: A multi-strategical approach to constrained multi-object recognition	13
3.8 A Survey of Monte Carlo Tree Search Methods	13
3.9 Mastering the game of Go with deep neural networks and tree search Mastering the Game of Go without human knowledge	14
3.10 Accelerating Self-Play Learning in Go	15
4 Konzeptentwicklung	16
4.1 Einführung in das Konzept	16
4.1.1 Projektübersicht und Unterscheidungsmerkmale	16

4.1.2	Anwendungsszenarien	17
4.1.3	Nutzen	17
4.2	Zielgruppenanalyse	18
4.3	Anforderungsanalyse	18
4.3.1	Spielbrett- und Spielstein-Erkennung	19
4.3.2	Spielanalyse	19
4.3.3	Benutzerinteraktion und Darstellung	20
4.3.4	Umgebungsanforderungen	20
4.3.5	Stabilität und Sicherheit	21
4.4	Beitrag zur Forschung und aktuellen Trends	21
4.5	Konzept der Benutzererfahrung	22
4.6	Systemarchitektur (Überblick)	22
4.6.1	Betriebssystem-API	22
4.6.2	Kamera-Interface	22
4.6.3	Benutzeroberfläche	23
4.6.4	Bildverarbeitung	23
4.6.5	Bildverarbeitungs-Bibliothek	24
4.6.6	Analyse-Manager	24
4.6.7	HTTP Client	24
4.6.8	Externe Analyse API	24
4.7	Auswahl der Technologien	25
4.7.1	Plattform: Android	25
4.7.2	Computer Vision: OpenCV	25
4.7.3	Programmiersprache: Java	25
4.7.4	Augmented Reality: Eigenentwicklung ohne ARCore	26
4.7.5	Spielanalyse: Externe API	26
4.8	Herausforderungen und Lösungsstrategien	26
4.9	Risikobewertung und Qualitätssicherung	28
4.9.1	Unsicherheiten bei eigenen Ansätzen	28
4.9.2	Detektionsfehler und die Benutzererfahrung	28
4.9.3	Technologische Abhängigkeiten	29
4.9.4	Qualitätssicherung	29
5	Implementierung und Tests	30
5.1	Technische Grundlagen und Entwicklungsumgebung	30

5.2	Implementierte Systemarchitektur	32
5.2.1	Datenfluss und Verarbeitungslogik	32
5.2.2	Asynchrones Thread Management	33
5.3	Implementierung des User Interface	34
5.4	Implementierung der Computer-Vision-Technologie	35
5.4.1	Vorverarbeitung	35
5.4.2	Erkennung der Ecken des Spielbrettes zur Entzerrung	41
5.4.3	Erkennung der schwarzen Spielsteine	49
5.4.4	Erkennung der weißen Steine	55
5.4.5	Abstraktion der Go-Stellung	59
5.5	Implementierung der Spiel-Analyse	60
5.6	Integration der augmentierten Darstellung	64
5.7	Qualitätssicherung und Tests	66
6	Evaluation und Diskussion der Ergebnisse	67
6.1	Verwendete Hardware	67
6.2	Verwendetes Testset	68
6.3	Evaluation der Spielbretterkennung	69
6.3.1	Bewertungskriterien (Spielbretterkennung)	69
6.3.2	Methodik und Ergebnisse (Spielbretterkennung)	69
6.3.3	Probleme (Spielbretterkennung)	70
6.3.4	Analyse und Interpretation der Ergebnisse (Spielbretterkennung)	71
6.3.5	Weitere Verbesserungsmöglichkeiten (Spielbretterkennung)	72
6.3.6	Vergleich mit bestehender Literatur (Spielbretterkennung)	72
6.4	Evaluation der Go-Stellungserkennung	73
6.4.1	Bewertungskriterien (Stellungserkennung)	73
6.4.2	Methodik (Stellungserkennung)	73
6.4.3	Ergebnisse (Stellungserkennung)	75
6.4.4	Probleme (Stellungserkennung)	75
6.4.5	Analyse und Interpretation der Ergebnisse (Stellungserkennung)	80
6.4.6	Weitere Verbesserungsmöglichkeiten in der Stellungserkennung	81
6.4.7	Vergleich mit bestehender Literatur (Stellungserkennung)	81
6.5	Evaluation der Spielanalyse	82
6.5.1	Bewertungskriterien (Spielanalyse)	82
6.5.2	Methodik und Ergebnisse (Spielanalyse)	82
6.5.3	Probleme (Spielanalyse)	82

6.5.4	Analyse und Interpretation der Ergebnisse (Spielanalyse)	82
6.6	Evaluation der Ergebnisdarstellung und des Userinterfaces	83
6.6.1	Bewertungskriterien (Ergebnisdarstellung und Userinterface)	83
6.6.2	Methodik und Ergebnisse (Ergebnisdarstellung und UI)	83
6.6.3	Probleme (Ergebnisdarstellung und Userinterface)	84
6.6.4	Analyse und Interpretation der Ergebnisse (Ergebnisdarst./UI)	84
6.7	Evaluation der Technischen Leistungsfähigkeit	85
6.7.1	Bewertungskriterien (Technischen Leistungsfähigkeit)	85
6.7.2	Methodik und Ergebnisse (Technischen Leistungsfähigkeit)	85
6.7.3	Probleme (Technischen Leistungsfähigkeit)	87
6.7.4	Analyse und Interpretation der Ergebnisse (Technischen Leistungs- fähigkeit)	88
6.7.5	Ausblick (Technischen Leistungsfähigkeit)	89
6.8	Evaluation der Benutzerfreundlichkeit und Kompatibilität	90
6.8.1	Bewertungskriterien (Benutzerfreundlichkeit und Kompatibilität)	90
6.8.2	Methodik und Ergebnisse (Benutzerfreundlichkeit und Kompatibi- lität)	90
6.8.3	Analyse und Interpretation der Ergebnisse (Benutzerfreundlichkeit und Kompatibilität)	90
6.9	Fazit der Evaluation	91
7	Fazit und Ausblick	92
	Literaturverzeichnis	94
A	Anhang	97
A.1	Verwendete Hilfsmittel	97
	Selbstständigkeitserklärung	98

Abbildungsverzeichnis

4.1	Überblick über die geplanten Komponenten	23
5.1	Die implementierte Systemarchitektur	31
5.2	Der Verarbeitungszyklus des Main- und der Analyse-Threads	33
5.3	Das Interface mit Debug Menü	34
5.4	Das Graubild vor und nach der Normalisierung	36
5.5	Das Graubild vor und nach einem Adaptiver Histogrammausgleich	37
5.6	Die SV-Ebene links und ihre definierte Farbintensität rechts	38
5.7	Das Farbbild des Go-Sets links und seine Farbintensität rechts	38
5.8	Links: Farbintensitätsbild, Mitte: Originales Graustufenbild Rechts: Ergebnis ihrer Addition	39
5.9	Ergebnis der Subtraktion der Farbintensität vom originalen Graustufenbild	40
5.10	Mögliche Binärbildumwandlungen: Links oben: Original, Rechts oben: Canny Edge Detection, Links unten: Adaptiver Schwellenwert, Rechts unten: Statischer Schwellenwert	41
5.11	Gespielte Steine stellen die Linienerkennung vor eine Herausforderung . .	43
5.12	Konturen des Spielbrettes: Links Oben: unbearbeitetes Binarbild, Rechts Oben: größte Kontur des unbearbeiteten Brettes, Links Unten: verbessertes Binarbild, Rechts Unten: größte Kontur des verbesserten Binarbilds	44
5.13	Kantenerkennung des Spielbrettes: Oben: Erkannte Kanten-Cluster, Links Unten: gemittelte Kanten, Rechts Unten: ermittelte Eckpunkte	46

5.14	Die von der Perspektive befreiten transformierten Basisbilder	48
5.15	Das Basisbild zur Erkennung von schwarzen Steinen vor und nach dem Filtern von dünnen Elementen	50
5.16	Ungenaue Platzierungen beeinflussen benachbarte Felder	51
5.17	Die Heatmap der Steine und ihr Binärbild nach Schwellenwertanwendung	52
5.18	Das Ergebnis der Erkennung der schwarzen Steine	53
5.19	Die Mittelpunkte der Steine (Rot) und die unter der Perspektive ange- passten Positionen (Gelb)	54
5.20	Arbeitsschritte zur Separierung des Hintergrundes: Links oben: Ursprüngliches transformiertes Basisbild, Rechts oben: Basisbild ohne Gitterlinien, Links unten: Binärmaske der zu füllenden Areale, Rechts unten: Der separierte Hintergrund	55
5.21	Reflexionsminderung durch separierten Hintergrund: Links: Das ursprüngliche transformierte Basisbild, Mitte: Der separierte Hintergrund, Rechts: Das Basisbild nach Abzug des Hintergrundes	57
5.22	Die erkannten Positionen vor (Türkis) und nach der Perspektivanpassung (Dunkelblau)	58
5.23	Eine Visualisierung der erkannten abstrahierten Go-Stellung	59
5.24	Eine vereinfachte Darstellung des Datenverlaufs zur Spiel-Analyse	61
5.25	Die Schritte zur augmentierten Darstellung der Ergebnisse	64
6.1	Eine repräsentative Auswahl des verwendeten Testsets	68
6.2	Ein gefülltes Brett separiert das Gitter in Segmente	70
6.3	Die Ergebnisse der Steinerkennung	74
6.4	Fehlende Erkennung weißer Steine bei starken Brettreflexionen: Links oben: Originalbild, Rechts oben: Transformiertes Basisbild, Links unten: Separierter Hintergrund, Rechts unten: Binärbild der weißen Steine nach Abzug des Hintergrundes	76
6.5	Missinterpretation von Schatten als schwarze Spielsteine: Links: Originalbild, Rechts: Erkannte Positionen schwarzer Spielsteine (gelb markiert)	77

6.6	Fehlende Erkennung schwarzer Steine bei Reflexionen: Links oben: Originalbild, Rechts oben: Transformiertes Basisbild mit Steinerkennung, Links unten: Heatmap der schwarzen Steine, Rechts unten: Binärbild der schwarzen Steine nach Anwendung des Grenzwertes	78
6.7	Missinterpretation von Schatten als schwarze Spielsteine: Links: Originalbild, Mitte: Erkannte schwarze Spielsteine, Rechts: Spielsituation nach der Schnittpunktzuordnung	79
6.8	Die Benutzeroberfläche mit Debug-Menü	83
6.9	Speicher- und CPU-Auslastung der App	85
6.10	Zeitliche Aufschlüsselung der Verarbeitungsschritte	86
6.11	Profilierung der Methodenaufrufe	87

Tabellenverzeichnis

A.1	Verwendete Hilfsmittel und Werkzeuge	97
-----	--	----

1 Einleitung

In einer zunehmend digitalisierten Welt verschwimmen die Grenzen zwischen virtueller und physischer Realität. Dies führt zu der Frage, wie physische Elemente unseres Alltags durch moderne Technologie bereichert und transformiert werden können. Vor diesem Hintergrund widmet sich diese Arbeit der Entwicklung eines mobilen Anwendungssystems, das darauf abzielt, eine Symbiose zwischen den sozialen Vorteilen des physischen Brettspiels und den analytischen Vorzügen digitaler Technologien zu schaffen. Die zentrale Forschungsfrage lautet: Wie kann ein mobiles Anwendungssystem entwickelt werden, das eine intuitive, schnelle und automatisierte Analyse von physischen Brettspielen, speziell anhand des Beispiels Go, ermöglicht und sich immersiv in spontane Spielsituationen einfügt, ohne dabei auf eine manuelle Digitalisierung angewiesen zu sein?

In diesem Kontext bietet die Integration von Computer-Vision-Techniken und Augmented Reality eine wirksame Lösung. Der Einsatz von Computer Vision ermöglicht die automatische und damit zeitsparende Erfassung des Spielgeschehens für eine Analyse. Gleichzeitig verbessert die immersive Darstellung der Spielanalysen direkt auf dem physischen Spielbrett die Lernerfahrungen, ohne dabei die Spielatmosphäre zu stören. Diese Techniken ermöglichen es Spielern, ihre Strategien und Taktiken direkt während des Spiels zu reflektieren und zu optimieren.

Die vorgestellte mobile Anwendung analysiert Go-Partien mittels Kamera schnell und intuitiv. Um optimale Züge und Spielstände zu ermitteln, verwendet sie eine Bildverarbeitungs-pipeline zur Erkennung der Spielsituation und ruft eine externe Analyse-API zur Spielanalyse auf. Der Fokus auf das Spiel Go erfolgte, da es hier bereits eine umfangreiche Forschungsbasis zu diesen Bereichen gibt, welche als solide Grundlage für die Entwicklung dient.

Bisherige Forschungsarbeiten konzentrierten sich primär auf Einzelkomponenten wie die automatische Erfassung von Spielstellungen, die Analyse digitalisierter Brettstellungen

und die Anwendung von Augmented Reality zur erweiterten Wahrnehmung bei Brettspielen. Es wurde jedoch noch nicht untersucht, wie diese Komponenten erfolgreich in einer umfassenden Anwendung zusammengeführt und kooperativ eingesetzt werden können. Diese Arbeit widmet sich der Identifizierung und Erweiterung geeigneter Ansätze, um sie zu einem kohärenten All-in-One-System zu kombinieren, das den Anforderungen des realen Anwendungsfalls gerecht wird. Dabei liegt ein Schwerpunkt auf der Verbesserung der Brettstellungserkennung unter Verwendung innovativer Methoden in der Computer Vision, während die Analyse und Augmented Reality unterstützende Funktionen darstellen.

Die Motivation des Autors für diese Arbeit entspringt seiner Leidenschaft für das Spiel Go und seiner aktiven Rolle bei der European Go Federation. Mit der Entwicklung dieser App zielt er darauf ab, einen wertvollen Beitrag für die Go-Community zu leisten, indem er die Analyse und das Verständnis des Spiels vereinfacht. Diese Zielsetzung wird ergänzt durch das wissenschaftliche Interesse, die Anwendungsmöglichkeiten von Computer-Vision-Techniken in der Praxis zu erforschen und weiterzuentwickeln.

Der Aufbau der Arbeit ist strukturiert, um einen klaren und umfassenden Einblick in diesen Prozess zu geben. Zunächst werden in Kapitel 2 Grundlagen erläutert, darunter eine Einführung in Computer Vision, Go und neurale Netzwerke zur Spielanalyse. Kapitel 3 behandelt verwandte Arbeiten im Bereich der Brettspielanalyse und Augmented Reality. Die Konzeptentwicklung wird in Kapitel 4 ausführlich beschrieben, einschließlich der Zielgruppenanalyse, Anforderungsanalyse und Systemkomponenten. Kapitel 5 konzentriert sich auf die Implementierung und Tests des entwickelten Systems. Die Evaluation und Diskussion der Ergebnisse erfolgen in Kapitel 6, wobei verschiedene Aspekte wie Erkennungsraten und technische Leistungsfähigkeit bewertet werden. Schließlich fasst Kapitel 7 die Ergebnisse zusammen und bietet einen Ausblick auf zukünftige Entwicklungen und Anwendungsmöglichkeiten des entwickelten Systems.

2 Grundlagen

2.1 Einführung in Computer Vision

Computer Vision befasst sich mit der Verarbeitung und Analyse digitaler Bilder, um daraus abstrakte Informationen auf einer höheren Ebene zu gewinnen. Richard Szeliski beschreibt es in seinem Buch “Computer Vision: Algorithms and Applications” wie folgt: “In computer vision, we are trying to [...] describe the world that we see in one or more images and to reconstruct its properties, such as shape, illumination, and color distributions” (Szeliski, 2022). Während diese Aufgabe für Menschen, sogar für Kinder, trivial sein mag, stellt sie in der Informatik eine bedeutende Herausforderung dar, insbesondere an der Schnittstelle der bildlichen Datenverarbeitung zwischen Computern und der realen Welt. Ohne diese Fähigkeit könnten Maschinen keine Entscheidungen und Vorhersagen auf der Grundlage analytischer Daten treffen.

Zu den Hauptaufgaben der Computer Vision gehört die Erkennung von Objekten und Merkmalen. Beispielsweise werden aus der Erkennung von Kanten, Linien und Kreise identifiziert. Mithilfe dieser abstrahierten Informationen können dann ganze Objekte klassifiziert und analysiert werden. Ein Schwerpunkt der modernen Computer Vision liegt auf dem maschinellen Lernen. Dies umfasst Techniken wie Deep Learning, bei denen neuronale Netze eingesetzt werden, um Bilder zu klassifizieren.

Eine der Hauptherausforderungen in der Computer Vision sind schlechte Beleuchtungsverhältnisse. Dunkelheit verschlechtert die Qualität der Bilder und den Kontrast der Merkmale. Schatten können Kanten erzeugen, die zu keinen Objekten gehören, und Reflexionen können ein Bild ungleichmäßig aufhellen sowie primäre Objekte spiegeln. Auch die Perspektive der Bilder beeinflusst die Objekte und deren Varianz (Hartley und Zisserman, 2003). Objekte derselben Klasse können unterschiedliche Merkmale aufweisen und teilweise durch andere Objekte verdeckt sein.

In den letzten Jahren hat die Relevanz von Computer Vision-Anwendungen zugenommen. Wichtige Anwendungsbereiche wie “Self-driven vehicles: capable of driving point to point between cities” und “Medical Imaging: registering pre-operative and intra-operative imagery” zeugen von dem technologischen Fortschritt, der durch moderne Computer Vision erreicht wird (Szeliski, 2022). Die allgegenwärtige Verfügbarkeit von Smartphones hat auch mobile Anwendungen, wie Gesichtserkennung und Anwendungen für Augmented Reality, alltäglich gemacht.

In diesem Projekt wird Computer Vision verwendet, um die Stellung eines Go-Spiels korrekt zu erkennen, damit diese weiterverarbeitet werden kann. Dies ist entscheidend für die Analyse von Spielstrategien und die nächste Phase der Datenverarbeitung.

2.2 Einführung in Go

Go ist eines der ältesten abstrakten Brettspiele der Welt. Bereits vor über 2500 Jahren wurde Go in Asien gespielt und wurde zur Vermittlung von Logik, Strategie und Konfliktlösung verwendet (Shotwell, 2011). Noch heute hat es eine hohe kulturelle Bedeutung für die Bevölkerung dieser Region. Seine Faszination liegt in der Tiefe des Spiels und erfreut sich daher weiterhin einer großen Spielerschaft und einer professionellen Szene.

Beim Go gibt es nur drei Regeln. Zwei Spieler legen abwechselnd schwarze und weiße Steine auf die Schnittpunkte eines 19x19 Netzes. Der Spieler, der mit seiner Farbe die meisten freien Schnittpunkte umzingelt, ist der Sieger. Wird eine Gruppe von Steinen komplett vom Gegner umzingelt, ist sie gefangen und wird vom Brett genommen. Eine Stellungswiederholung ist nicht erlaubt. Genauere Regeln lassen sich der Webseite der “Japan Go Association” entnehmen (Association, 2023).

Trotz seiner einfachen Regeln besitzt Go eine hohe Komplexität, die es zu einem interessanten Forschungsgebiet für künstliche Intelligenz macht. Die Herausforderung für die KI liegt hierbei in der hohen Anzahl möglicher Züge und der Intuition, die erfahrene Spieler verwenden, um diese einzugrenzen. Trotz hoher Aufwendungen war es bis vor Kurzem nicht möglich, eine KI zu entwickeln, die erfahrene Spieler besiegen konnte (Müller, 2002). Durch einen Durchbruch bei der Verwendung neuronaler Netze gelang dies letztendlich DeepMind mit AlphaGo im Jahr 2016. Auch in der Computer Vision wurde es wiederholt aufgegriffen, da es sich gut als Anwendungsfall für Mustererkennung eignet. Die einfache Anordnung von runden Steinen auf einem so großen Gitternetz

verdeutlicht den Bedarf einer hohen Genauigkeit, da bereits kleine Abweichungen große Unterschiede in der Evaluation der Stellung machen können. Damit verbindet Go eine uralte Tradition mit moderner Technologie.

2.3 Neuronale Netzwerke zur Spielanalyse

Neuronale Netze gewinnen zunehmend an Relevanz, da sie ein abstraktes Konzept bieten, das sich auf eine Vielzahl komplexer Probleme anwenden lässt. Sie ähneln in ihrer Struktur dem Aufbau des menschlichen Gehirns, was sie besonders anpassungsfähig für neue Aufgabenstellungen macht. Einzelne Knoten, ähnlich den Neuronen, sind unterschiedlich stark miteinander verbunden, um Muster in Daten zu erkennen. Im Kontext dieser Arbeit ist es wichtig, neuronale Netze zu untersuchen und zu verstehen, da sie den Grundstein zur Spielanalyse darstellen.

Es gibt verschiedene Arten von Netzwerken; jedoch sind für die Bildverarbeitung und die Analyse von Strategiespielen die Convolutional Neural Networks (CNNs) am relevantesten. “[...] CNNs are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include [...] image data, which can be thought of as a 2-D grid of pixels” (Heaton, 2018). Die räumlichen Mustererkennungsfähigkeiten von CNNs sind also besonders nützlich bei der Analyse der räumlichen Strukturen, die in Strategiespielen wie Go vorhanden sind.

Neuronale Netze müssen für eine Aufgabe trainiert werden, um sie zu beherrschen. Dieser Lernvorgang lässt sich grundsätzlich in drei Arten unterteilen:

Beim Supervised Learning wird das Netzwerk mit Daten trainiert, die Eingabe/Ausgabe Paare enthalten. Die Trainingsdaten müssen eine repräsentative Stichprobe der zu analysierenden Daten abdecken. Durch den Vergleich der Ausgaben des Netzwerks mit den gewünschten Ausgaben wird das Modell angepasst, bis das Modell ausreichend präzise ist. Diese Methode wird eingesetzt, um aus historischen Daten Vorhersagen oder Klassifizierungen zu treffen.

Beim Reinforcement Learning stehen weniger Informationen zur Verfügung. Das Netzwerk erhält lediglich Rückmeldungen in Form von Belohnungen oder Strafen, basierend auf den getroffenen Entscheidungen. Das Netzwerk wird dann angepasst, sodass die Belohnungen maximiert werden. Diese Methode eignet sich zum Beispiel für Strategiespiele, bei denen die KI durch Spielen gegen sich selbst lernt.

Beim Unsupervised Learning gibt es keine konkreten Rückmeldungen über die korrekte Ausgabe. Das Netzwerk versucht autonom, Eingabedaten in ähnliche Klassen zu gruppieren oder Muster in den Daten ohne vorherige Kenntnis der Ergebnisse zu erkennen. Es wird häufig verwendet, um verborgene Cluster in nicht etikettierten Daten zu entdecken.

Das Training von neuronalen Netzen ist typischerweise ein sehr langsamer und rechenintensiver Prozess. Übermäßiges Training kann auch zu einer Überspezialisierung des Netzwerks führen, sodass es möglicherweise auf neuartige Situationen nicht mehr effektiv reagieren kann.

In der Praxis werden zur Analyse meist bereits trainierte Netze verwendet. Diese bieten den Vorteil, dass der Nutzer Rechenleistung primär für die Anwendung und nicht für das Training aufwenden muss. Allerdings können sie sich dadurch auch nicht mehr an neue Gegebenheiten anpassen und haben möglicherweise Schwierigkeiten, mit unerwarteten Situationen umzugehen.

Ein Meilenstein in der Entwicklung mit neuronalen Netzen wurde von DeepMind mit AlphaGo erreicht. Es war die erste KI, die einen menschlichen Go-Profi besiegen konnte und meisterte damit eine Aufgabe, die zuvor durch traditionelle maschinelle Verfahren nicht annähernd gelöst werden konnte. Während die ursprüngliche Version dieser Netze noch durch Supervised Learning trainiert wurde, wurde diese später durch eine noch stärkere Reinforcement Learning Variante ersetzt (Silver u. a., 2016) (Silver u. a., 2017). Dieser Durchbruch veränderte die Bedeutung von maschinellem Lernen nachhaltig und läutete das moderne Zeitalter der Dominanz von neuronalen Netzen ein, was sich in Anwendungen wie Chat GPT widerspiegelt.

In Zukunft wird diese Technologie noch zu deutlich fortschrittlicheren Anwendungen führen. Die ethische Verantwortung, die mit der Verwendung von immer leistungsfähigeren Netzwerken einhergeht, darf daher nicht übersehen werden. “AI and robotics have raised fundamental questions about what we should do with these systems, what the systems themselves should do, and what risks they have in the long term” (Müller, 2020). Mit zunehmender Autonomie müssen potenzielle gesellschaftliche Auswirkungen bedacht werden, und Grenzen müssen gezogen werden, die negative Auswirkungen begrenzen.

2.4 Augmented Reality (AR) in mobilen Anwendungen

Durch die ständige Verfügbarkeit von Smartphones und die dadurch ermöglichte Nutzung von Augmented Reality hat sich diese Technologie in den letzten Jahren signifikant weiterentwickelt. In seinem Artikel “A Survey of Augmented Reality” definiert Azuma AR durch die folgenden drei Merkmale (Azuma, 1997):

- Kombiniert Realität und virtuelle Elemente (“Combines real and virtual“),
- Interaktivität in Echtzeit (“Interactive in real time“),
- Registriert in 3-D (“Registered in 3-D”).

Diese Merkmale unterstreichen, dass AR die physische Welt mit virtuellen Inhalten bereichert, die auf die Sinne einwirken. Die Wahrnehmung wird also so erweitert, dass sie als natürliche Elemente der Umgebung wahrgenommen werden. AR reagiert unmittelbar auf die Aktionen der Nutzer und unterscheidet sich damit von statischen Visualisierungen, indem sie interaktiv sind. Außerdem ist die korrekte Platzierung der Objekte für eine immersive Erfahrung unerlässlich. Für Brettspiele wäre dies zum Beispiel eine glaubwürdige Positionierung von Spielsteinen.

Eine Herausforderung besteht also darin, die Perspektive korrekt zu erfassen und reale Objekte richtig zu erkennen, damit virtuelle Elemente glaubwürdig in die reale Umgebung eingebettet werden können. Hierbei ist eine effiziente Integration von Computer Vision-Techniken entscheidend, um eine korrekte Erkennung und Positionierung von Objekten zu ermöglichen.

Es ist wichtig, AR von Virtual Reality (VR) abzugrenzen. In einer VR-Umgebung wird die Wahrnehmung der Realität nicht nur erweitert, sondern durch eine vollständig virtuelle Umgebung ersetzt.

Im mobilen Bereich hat AR große Fortschritte gemacht. Smartphones besitzen bereits die notwendigen Kameras und Sensoren, die für AR benötigt werden. Die Nutzer können somit durch die Bildschirme hindurch eine veränderte Welt erleben. Die Anwendungen reichen von einfachen Spielen bis zu medizinischen Anwendungen im Gesundheitswesen. Sie können auch ermöglichen, den Nutzern zusätzliche Informationen zu Objekten in ihrer Umgebung einzublenden oder ihr Wohnzimmer virtuell neu einzurichten, um einen Kauf zu evaluieren.

Im Kontext von physischen Brettspielen eignet sich AR, um den Spielstand, die beste Aktion oder sogar ganze Spielvarianten aufzuzeigen.

Angesichts der Herausforderungen der modernen Zeit, wie der Pandemie und dem Fortschreiten dieser Technologie, ist es klar, dass AR in unserer Zukunft eine große Rolle spielen wird, insbesondere als Bildungs- und Lernwerkzeug.

2.5 Grundlagen von Web-APIs

Web-APIs (Application Programming Interfaces) spielen eine wichtige Rolle bei der Kommunikation im Web. Sie dienen als Schnittstelle zwischen verschiedenen Applikationen und ermöglichen den Austausch von Daten mittels Internetprotokollen, meist HTTPS.

Diese APIs erlauben den Zugriff auf Dienste im Internet, wodurch Entwickler Services, die auf anderen Plattformen oder Geräten bereitgestellt werden, in ihre Anwendungen integrieren können. Clients senden eine HTTP-Anfrage und erhalten in der Regel eine Antwort im JSON- oder XML-Format, die aufgrund ihrer Lesbarkeit in der Programmierung bevorzugt werden. Der Trend geht vor allem zu REST-Diensten, da sie wegen ihrer Einfachheit und der Nutzung von Standard-HTTP-Methoden wie GET, POST, PUT und DELETE bei Entwicklern beliebt sind (Fielding, 2000).

Web-APIs erfordern einen verantwortungsbewussten Umgang mit dem Datenschutz. Häufig beinhalten sie Zugriffe auf sensible, benutzerbezogene Daten. Deshalb müssen Datenschutzrichtlinien strikt befolgt und umgesetzt werden, um die Privatsphäre und Sicherheit der Nutzer zu gewährleisten.

3 Verwandte Arbeiten

3.1 Augmented Reality for Board Games

Im Konferenzbericht “Augmented Reality for Board Games” befasst der Autor sich mit der Ergänzung von traditionellen Brettspielen mit AR-Elementen anhand des Spiels Monopoly (Molla und Lepetit, 2010). Durch das Ersetzen der Spielfiguren und die Einführung von weiteren visuellen Elementen auf dem Brett liegt der Fokus darauf, mehr Immersion für den Spieler zu schaffen. Zur Brett-Erkennung wird ein BRIEF-Deskriptor als Methode genutzt, um Merkmale zwischen einem Referenzbild und dem von der Webcam aufgenommenen Bild abzugleichen. Um die Spielfiguren zu detektieren, wird ein Viola-Jones-Objektdetektor genutzt. Er wurde ursprünglich für die Gesichtserkennung entwickelt und funktioniert am besten mit Objekten, die markante Merkmale haben, nachdem ein Trainingsverfahren durchgeführt wurde. Dieses Training erfordert eine umfangreiche Sammlung von Beispielen, die genau die Art von Objekt darstellen, die erkannt werden soll. Auf die Darstellung der AR-Objekte wird nicht detailliert eingegangen.

Die Nutzung von BRIEF-Detektoren könnte bei einer Anwendung auf ein Go-Brett auf Schwierigkeiten stoßen. Go-Bretter haben ein gleichmäßiges Gittermuster, das wenige einzigartige Merkmale hat. Des Weiteren werden diese Merkmale im Laufe des Spiels durch immer mehr Spielsteine verdeckt. Die Verwendung des Viola-Jones-Algorithmus könnte zu Problemen führen, da Go-Steine wenige einzigartige Merkmale haben, vor allem, wenn sie sich zu Clustern verbinden und sich gegenseitig verdecken.

3.2 Fast radial symmetry for detecting points of interest

Im Artikel “Fast radial symmetry for detecting points of interest” stellen die Autoren eine Methode zur Erkennung von Merkmalen basierend auf radialer Symmetrie vor (Loy und Zelinsky, 2003). Sie hebt also Areale eines Bildes, die runde Objekte wie Augen oder Räder haben, hervor. Sie ist robust gegenüber Bildrauschen und Verdeckungen und eignet sich für den Einsatz in Echtzeit. Obwohl dieser Algorithmus ideal für die Erkennung von Go-Steinen in unverzerrten Bildern erscheint, ist er nicht in vielen Standardbibliotheken enthalten, was wahrscheinlich eine eigene Implementierung erfordern würde. Auch bei der Bildung von größeren Gruppen von Steinen eines Spielers und damit einer Flächenbildung einer Farbe könnten weitere Symmetrien innerhalb der Gruppe erkannt werden, die keine Steine repräsentieren.

3.3 Model Improves a Weak Classifier

Obwohl die Primärquelle des Reports von Scher leider nicht mehr verfügbar ist, stellt sie einen einfachen Ansatz dar, der durchaus erwähnenswert ist (Seewald, 2010a). Das Ziel dieser Arbeit ist es, mithilfe von Videoaufnahmen die Züge einer gesamten Go-Partie zu erkennen. Die Kamera ist statisch, und die Eckpunkte des Go-Brettes werden manuell eingegeben. Damit wird das Bild entzerrt und die Schnittpunkte werden eindeutig festgelegt. Es kommt eine Kantenerkennung zum Einsatz, deren Ergebnis mit einer Hough-Transformation für Kreise verarbeitet wird, um die Steine zu identifizieren. Diese werden dann den dichtesten Schnittpunkten zugeordnet.

Ein Problem dieser Methode stellt die Zuverlässigkeit der erkannten Kanten dar, da kein Pre-Processing erfolgt. Dies bereitet insbesondere beim Auftreten von Reflexionen Probleme. Zudem ist die Voraussetzung von manuell festgelegten Ecken heutzutage veraltet. In dieser Arbeit soll die Ermittlung der Eckpunkte automatisch erfolgen.

3.4 Automatic extraction of game record from TV Baduk program

Der Konferenzbericht “Automatic extraction of game record from TV Baduk program” zielt darauf ab, Go-Partien aus dem TV mitzuschreiben (Kang u. a., 2005). Dafür wird ein unverzerrtes, quadratisches Brett vorausgesetzt. Das Gitter des Go-Brettes wird durch die Erstellung eines vertikalen und horizontalen Histogramms detektiert, indem regelmäßige Peaks in jeder Dimension identifiziert werden. Frames, in denen kein quadratisches Brett erkannt wird, weil Teile des Brettes durch Arme oder Einblendungen verdeckt sind, werden verworfen. Danach erfolgt die Erstellung eines Histogramms über das erkannte Brett. Basierend auf diesen Daten wird ein Threshold für die weißen und die schwarzen Steine angewandt, um diese zu erkennen. Werden nach dem Threshold in einem Areal neue Pixel erkannt, wird dies als der neue Zug wahrgenommen.

Für die Stellungserkennung ist diese Methode nur dann geeignet, wenn bereits ein unverzerrtes Brett vorliegt. Die Linien können nur dann durch ein vertikales bzw. horizontales Histogramm erkannt werden, wenn das Brett noch recht leer ist und somit die Peaks für die Linien eindeutig sind. Probleme wie Reflexionen oder ungleichmäßige Beleuchtung werden aufgrund eines statischen Thresholds bei der Steinerkennung ignoriert. Was dazu führt, dass einige Steine falsch erkannt werden. Insgesamt ist es daher nicht möglich, diese Methode ohne weitere Prozessierung zur Bildbeleuchtung und einer besseren Methodik zur Bretterkennung zuverlässig einzusetzen.

3.5 Extracting go game positions from photographs

Der Artikel “Extracting go game positions from photographs” beschreibt eine Methode zur Erkennung der Go-Brettstellung in Einzelbildern (Hirsimäki, 2005). Für die Separierung des Gitters kommt ein adaptiver Schwellenwert zum Einsatz. Anschließend wird eine Hough-Transformation für Linien verwendet. Mithilfe der Heatmap der Hough-Linien wird ein kleineres Raster identifiziert, das sich auf das am deutlichsten hervortretende regelmäßige Untergitter konzentriert. Unter Zuhilfenahme einer Minimierungsfunktion erfolgt die schrittweise Erweiterung des Gitters, bis es ein 19x19 großes Gebiet umfasst. Steine werden anhand eines einfachen Schwellenwerts für die Fläche an den jeweiligen Schnittpunkten identifiziert.

Der adaptive Schwellenwert scheint eine praktikable Methode zum Filtern der Gitterlinien zu sein. Die Verwendung der Hough-Transformation ermöglicht einen guten Einblick in die mögliche Ausrichtung der Gitterlinien. Wenn das Brett jedoch stärker mit Steinen besetzt ist als mit Gitterlinien, könnte es vorkommen, dass die identifizierten Linien zwischen den Kanten der Steine positioniert sind, welche das ursprüngliche Gitter verdecken. Daraus resultierend kann das detektierte Gitter potenziell um ein halbes Feld in jede Richtung abweichen. Die Methode zur Erkennung von Steinen ist sehr rudimental und zeigt Schwächen, wenn die Beleuchtung auf dem Brett ungleichmäßig ist. Insbesondere Steine, die nicht mittig auf ihren Schnittpunkten platziert sind – was durchaus häufig vorkommt –, decken diese nicht vollständig ab. Diese Problematik wird besonders bei Betrachtung aus einem flachen Winkel von der Seite deutlich, da in solchen Fällen die Steine eine starke elliptische Form annehmen können. Des Weiteren besteht die Möglichkeit, dass Schatten von weißen Steinen fälschlicherweise als schwarze Steine interpretiert werden, während Reflexionen auf dem Brett oder sogar auf schwarzen Steinen irrtümlich als weiße Steine klassifiziert werden könnten.

3.6 Detection of Go-board contour in real image using genetic algorithm

Das Paper “Detection of Go-board contour in real image using genetic algorithm” beschreibt einen Prozess zur Konturerkennung eines Go-Brettes (Shiba und Mori, 2004). Zunächst wird das Bild in ein Binärbild umgewandelt, indem ein Sobel-Filter zur Kantenerkennung genutzt wird. Anschließend wird ein genetischer Algorithmus verwendet, um ein viereckiges Polygon an die Konturen des Brettes zu approximieren. Die Approximation wird bewertet und wiederholt, bis die präziseste Anpassung an das Bild erreicht ist.

Die Idee, die Konturen des Go-Brettes zu benutzen, um es zu lokalisieren, ist sehr interessant. Damit kann die Perspektive festgestellt und das Brett entzerrt werden. Jedoch setzt die Methode voraus, dass sich das Brett gut vom Hintergrund absetzt, um die Kanten zu erkennen. Da oft mit einem Holzbrett auf Holztischen gespielt wird, kann dies problematisch sein.

3.7 Automatic extraction of go game positions from images: A multi-strategical approach to constrained multi-object recognition

Die Arbeit von Seewald ergänzt traditionelle Bildverarbeitungstechniken mit maschinellem Lernen, um Go-Stellungen zu erkennen (Seewald, 2010b). Zuerst identifiziert der SIFT-Algorithmus wichtige Schlüsselpunkte im Kamerabild und erzeugt zu diesen Feature-Deskriptoren. Danach werden diese mithilfe von Machine Learning klassifiziert, um wichtige Elemente wie Schnittpunkte, T-Kreuzungen, Ecken und Go-Steine zu identifizieren. Das Modell lernte hierzu von den Trainingsteams, die unter denselben Bedingungen aufgenommen wurden. Aus diesen Daten werden die Eckpunkte des Brettes ermittelt. Durch Abgleichen der Perspektive mit einem 3D-Modell des Gitters wird die Position dann validiert und verfeinert. Die Steinerkennung wird zusätzlich zur Schlüsselpunktidentifikation mit einer Canny-Edge-Detection mit Threshold ergänzt. Die Farbe der Steine wird mithilfe ihrer durchschnittlichen Helligkeit ermittelt.

Das System erreicht eine hohe Genauigkeit unter unterschiedlichen Beleuchtungsverhältnissen und profitiert von dem Ansatz, Informationen aus verschiedenen Ansätzen zusammenzutragen und zu validieren. Allerdings wird in dieser Arbeit nur mit einem 8x8 Brett gearbeitet, was eine deutlich niedrigere Komplexität als die Detektion auf einem originalen 19x19 Brett erfordert und mit einer besseren Bildqualität pro Schnittpunkt arbeiten kann. Zur Klassifizierung müssen die Netze auch trainiert werden. Es müssen also größere Trainingssets mit unterschiedlichen Bedingungen zur Verfügung stehen. Die Netzwerke könnten zudem mit einer Klassifizierung unter neuartigen Bedingungen überfordert sein. Außerdem werden die Steine nicht zuverlässig durch SIFT klassifiziert, was einen klassischen Ansatz empfehlenswerter macht.

3.8 A Survey of Monte Carlo Tree Search Methods

Der Artikel “A Survey of Monte Carlo Tree Search Methods” untersucht die Monte Carlo Tree Search (MCTS)-Methoden, die besonders in der Analyse von Brettspielen eingesetzt werden (Browne u. a., 2012). Ausgehend von der Wurzel wählt der Algorithmus Kindknoten basierend auf Kriterien wie Erforschung neuer Knoten und Exploration guter Knoten

aus, bis er ein Blatt des Baumes findet. Ein Spiel wird dann zufällig bis zum Ende gespielt, um einen Gewinner zu ermitteln. Die Ergebnisse der Simulation werden verwendet, um die besuchten Knoten zu aktualisieren. Die Stärke ist, dass MCTS in Echtzeit arbeitet und keine spielspezifischen Vorkenntnisse benötigt, was den Algorithmus für eine Vielzahl von Brettspielen, wie Schach und Go, anwendbar macht. Der Artikel beleuchtet auch fortgeschrittene Anpassungen und Optimierungen des MCTS, die zur Verbesserung seiner Effektivität beitragen. In Umgebungen mit hoher Komplexität in der Bewertung und vielen möglichen Zügen kann MCTS jedoch sehr viel Rechenzeit benötigen.

Im Kontext von Go bildete die MCTS die dominante Analyseverfahren vor 2016, welche jedoch noch nicht annähernd an das Spielverständnis von Menschen herankommt. Nun bilden sie weiterhin eine Basis, die durch den Einsatz neuronaler Netze noch effektiver wird.

3.9 Mastering the game of Go with deep neural networks and tree search | Mastering the Game of Go without human knowledge

In der Studie “Mastering the game of Go with deep neural networks and tree search” revolutionieren die Autoren die Analysefähigkeiten durch KI in Brettspielen, indem sie deep neural networks mit der Monte Carlo Tree Search (MCTS) kombinieren (Silver u. a., 2016). Dieser Ansatz verbessert die traditionelle MCTS, indem sie nicht mehr zufällige Simulationen für die Zugauswahl verwendet, sondern stattdessen mit der Intuition von neuronalen Netzwerken informierte Entscheidungen trifft. Diese Netzwerke bewerten Spielzustände und bestimmen Züge, wodurch die Effizienz und Genauigkeit der Suche im Entscheidungsbaum verbessert wird. Das Netzwerk wird dabei zunächst durch überwachtes Lernen von Profipartien trainiert, und trainiert dann mit selbstverstärkendem Lernen durch Partien gegen sich selbst weiter. Das Ergebnis ist eine KI, die wesentlich effektiver in Go ist, das für ihre enorme Komplexität bekannt ist. Ergänzend hierzu entwickelt DeepMind in “Mastering the Game of Go without human knowledge” mit AlphaGo Zero, ein System, das ausschließlich durch Spielen gegen sich selbst lernt, ohne menschliche Spiele als Lerngrundlage (Silver u. a., 2017). Es nutzt ein verfeinertes neuronales Netzwerk, das gleichzeitig Positionsbewertung und Zugvoraussage durchführt. Dies führt zu einer schnelleren und effizienteren Lernstrategie. Diese Forschung hebt das Potenzial der MCTS hervor, wenn es mit maschinellem Lernen kombiniert wird, nicht nur in Go,

sondern auch als Analysemethode für eine Vielzahl anderer Brettspiele und Entscheidungsprozesse. Eine solche selbstlernende KI ist in der Lage, Wissen und Strategien zu generieren, die weit über die menschlichen Fähigkeiten hinausgehen.

Eines der Probleme dieser Technik ist der enorme Bedarf an Rechenressourcen, der erforderlich ist, um ein solches Netzwerk zu trainieren. Für viele Analyseanwendungen bleibt dieses Konzept daher aus finanzieller Sicht unrentabel. Im Kontext von Go treten jedoch weitere kleinere Probleme auf. AlphaGo kennt nur Gewinnwahrscheinlichkeiten und versteht das Konzept von Punkten nicht. In verlorenen Situationen ist AlphaGo daher nicht mehr in der Lage, den besten Zug zu finden, da alle Analysen zu einer Niederlage führen, was die Möglichkeit auf ein Comeback nach dem Fehler des Gegners nimmt. Ähnlich verhält es sich bei gewonnenen Stellungen, bei denen AlphaGo punktemäßig schlechte Züge macht, da ohnehin alle Züge zum Sieg führen. Dies ist insbesondere bei Handicap-Partien problematisch. AlphaGo ist in solchen Situationen nicht für Analysen geeignet.

3.10 Accelerating Self-Play Learning in Go

Das Paper “Accelerating Self-Play Learning in Go” stellt eine signifikante Weiterentwicklung dar, die auf den Erfolgen von AlphaGo aufbaut und in KataGo sichtbar wird (Wu, 2019). Es verbessert den Lernprozess durch den Einsatz optimierter Algorithmen für Partien gegen sich selbst. Der Fokus liegt dabei auf einem effizienteren Bewertungsnetzwerk, das präzise Spielstandsschätzungen vornehmen kann. So ist das Netzwerk nicht nur in der Lage, optimale Züge zu lernen, sondern auch suboptimale zu identifizieren. Diese Fähigkeit verbessert die Anpassungsfähigkeit der KI an verschiedene Spielstile und erhöht ihre Leistungsfähigkeit in Handicap-Partien, indem sie sowohl aus Siegen als auch aus Niederlagen lernen kann. Dadurch wird das Training insgesamt erheblich effizienter.

Im Unterschied zu AlphaGo zeichnet sich KataGo also durch die zusätzliche Berücksichtigung von Punkteunterschieden neben dem primären Fokus auf Gewinnwahrscheinlichkeit aus, was es derzeit zum führenden Analysewerkzeug macht.

4 Konzeptentwicklung

4.1 Einführung in das Konzept

In der Tradition des Go-Spiels sind Begegnungen über das Brett von großer sozialer und kultureller Bedeutung. Noch heute gibt es in Japan Go-Salons, die ein fester Bestandteil der kulturellen Landschaft sind. In ihnen versammeln sich Go-Spielende, um sich in Partien zu messen und Gemeinschaft zu erleben. In Deutschland hat mit dem Aufkommen des Internet-Go jedoch die Präsenz des physischen Spielens nachgelassen, da es den Leuten ermöglicht, bequem von zu Hause aus zu spielen. Die COVID-19-Pandemie hat diesen Trend noch verstärkt, wodurch viele Spieleabende nicht mehr stattfinden. Dabei geht eine wesentliche soziale Komponente verloren, die dieses Projekt wieder aufleben lassen möchte.

4.1.1 Projektübersicht und Unterscheidungsmerkmale

Das Konzept dieser Arbeit stellt eine All-in-One-Lösung für die Analyse physischer Go-Partien vor. Bisherige Analyse-Tools sind nur auf digitalen Plattformen verfügbar und benötigen daher eine vorherige manuelle Digitalisierung physischer Spiele. Die Ergebnisse werden auf einem abstrakten, digitalen Brett dargestellt. Projekte, die sich mit einer automatischen Digitalisierung von Partien beschäftigt haben, nutzten hierfür eine fest installierte Kamera, die damit einhergehende Vorbereitungen benötigt.

Die hier entwickelte App hingegen vereinfacht diesen Prozess durch die Verwendung der integrierten Kamera eines Smartphones. Spielstellungen können Ad-hoc aus der Perspektive eines sitzenden Spielers erkannt und anschließend analysiert werden. Das ermöglicht auch die hier verwendete Nutzung von Augmented Reality, um die Ergebnisse direkt auf dem physischen Brett darzustellen.

4.1.2 Anwendungsszenarien

Die Anwendung findet in verschiedenen Szenarien ihren Einsatz. Bei informellen Go-Treffen, wie in Cafés, Bars oder Go-Salons, bietet die App die Möglichkeit, nicht nur nach, sondern auch während des Spiels Analysen durchzuführen, sofern beide Spieler einverstanden sind. Dies ermöglicht es, aufkommende Diskussionen sofort zu evaluieren. In Turniersituationen ermöglicht die App Spielern, ihre Partien direkt nach dem Spiel zu analysieren. Zuschauer können mit der App unkompliziert und diskret den aktuellen Spielstand einer Partie ermitteln, ohne die Spieler zu stören oder den Spielverlauf zu beeinflussen. Go-Lehrer können die App nutzen, um schneller optimale Züge zu identifizieren und sich auf die Vermittlung der dahinterliegenden Strategien zu konzentrieren.

4.1.3 Nutzen

Die automatische Stellungserkennung ermöglicht Echtzeit-Analysen, was den Feedback-Loop verkürzt und eine effektivere Lernerfahrung ermöglicht. Dies motiviert Spieler, die sich aufgrund der Verfügbarkeit von Computeranalysen bisher auf Online-Go fokussierten, wieder zum traditionellen Brettspiel zurückzukehren.

Die mobile Anwendung bietet den Vorteil, dass keine zusätzliche Hardware benötigt wird, da sie mit Smartphones funktioniert, die Menschen stets dabei haben. Die App ist dadurch für ein breites Spektrum an Spielern zugänglich, unabhängig von deren technologischem Hintergrund.

Die augmentierte Darstellung bietet ein ästhetisch ansprechendes Spielerlebnis und reduziert die kognitive Belastung im Vergleich zu abstrakten Interfaces, da sie Informationen in das natürliche Spielumfeld integriert. Dies fördert die Konzentration auf das Spiel. Die Immersion verstärkt den Lernprozess, da Spieler sich stärker in das Spiel einbezogen fühlen. Die traditionelle Spielkultur des Go-Spiels wird erhalten, indem sie moderne Technologie in die physische Präsenz des Spiels integriert.

Anstatt Spieler in eine digitale Schnittstelle zu zwingen, fördert die Technologie somit eine Analyse direkt auf dem Spielbrett. Dies erleichtert Face-to-Face-Interaktionen und reduziert Ablenkungen, die durch den Fokus auf technische Geräte entstehen. Direkte Interaktionen am Spielbrett wirken sich motivierend auf umstehende Zuschauer aus, sich am Review-Prozess zu beteiligen. Dadurch trägt die App zur Stärkung der sozialen Komponente des Go-Spiels bei.

Die Applikation bietet zudem alle gängigen Vorzüge einer Computeranalyse. Dazu zählen die kontinuierliche Verfügbarkeit sowie eine objektive Fehlererkennung und -korrektur, die entscheidend zu einem vertieften Verständnis von Strategien beitragen. Diese Aspekte werden im Folgenden jedoch nicht weiter vertieft.

4.2 Zielgruppenanalyse

Die Hauptzielgruppe dieser App sind Go-Spieler, die eine Vorliebe für das physische Spiel haben. Sie sind überwiegend männlich und im Alter von 16 Jahren und älter, beinhalten aber auch ganze Familien. Vor allem diejenigen, die regelmäßig spielen und eine mittlere bis hohe Spielstärke aufweisen, suchen nach Möglichkeiten, sich zu verbessern. Sie sind bei Turnieren anzutreffen und spielen mindestens einmal im Monat. Das Gemeinschaftserlebnis ist für sie ein wesentlicher Bestandteil des Go-Spiels. Sie nutzen moderne Technologien wie Computer-Go-Software für die Spielanalyse. Die App richtet sich insbesondere an Spieler, die keine stärkeren Spieler in der Nähe haben, die sie um Rat fragen können. Sie bietet eine Lösung für diejenigen, die sich außerhalb der digitalen Medien verbessern wollen.

4.3 Anforderungsanalyse

Für die Entwicklung einer Go-Spielanalyse-App ist die Anforderungsanalyse unerlässlich, um ein klares Verständnis dafür zu erhalten, was Spieler und Nutzer der App benötigen und erwarten. Die Spieler erwarten nicht nur eine App, die zuverlässig Spielanalysen liefert, sondern auch in einer Vielzahl von Umgebungen funktioniert und eine intuitive Benutzererfahrung bietet. Darüber hinaus müssen wir Aspekte wie Datenschutz, Reaktionszeiten und Kompatibilität mit verschiedenen Geräten berücksichtigen, um eine breite Nutzung zu gewährleisten. Die Anforderungsanalyse definiert die zu erreichenden Ziele und beschreibt den gewünschten Endzustand der Software.

4.3.1 Spielbrett- und Spielstein-Erkennung

Die Anwendung muss in der Lage sein, das Go-Spielbrett auch bei variierenden Kamerawinkeln zu erkennen. Der Nutzer der App sitzt meist direkt vor dem Go-Brett und hat keinen orthogonalen Blick auf das Brett. Bei der gemeinschaftlichen Analyse kann es auch sein, dass ein Nutzer steht und einen diagonalen Blick auf das Brett hat.

Die Brettstellung muss korrekt erkannt werden. Die Positionserkennung von schwarzen und weißen Steinen auf dem Spielbrett und die Farbe des zu ziehenden Spielers definieren die Brettstellung. Der Nutzer der App erwartet, dass dieselbe Stellung analysiert wird, die auf dem Brett liegt.

Toleranz gegenüber verschiedenen Hintergrund- und Tischfarben ist vonnöten. Go wird an den verschiedensten Orten gespielt. Auf hellen oder dunklen Tischen, im Freien im Gras oder auf dem Teppich. Die Analyse sollte unabhängig vom Ort funktionieren.

Nicht relevante Objekte im Kamerabild müssen ignoriert werden. Oft stehen unrelatierte Objekte wie Trinkgläser und Snacks im Bild. Außerdem befinden sich die Bowls neben dem Brett, die die Spielsteine beinhalten und die gefangenen Steine. Sie dürfen die Analyse nicht beeinflussen.

Verschiedene Lichtverhältnisse müssen kompensiert werden. In der Wohnung sind die Lichtverhältnisse anders als in der Bar oder im Park. Die App muss in der Lage sein, die verschiedenen Lichtverhältnisse zumindest teilweise zu kompensieren. Allerdings ist es wichtig zu verstehen, dass die Anpassung an sämtliche Lichtsituationen eine komplexe Aufgabe darstellt und eine vollständige Abdeckung aller Szenarien unrealistisch ist. Es sollten aber zumindest einige Konzepte implementiert werden, die dies unterstützen.

4.3.2 Spielanalyse

Es muss ein Analysetool zur Bewertung des aktuellen Spielstands und zur Findung des optimalen nächsten Zugs verwendet werden. Dies ist die primäre Information, an der der Nutzer interessiert ist. Sie wird aus der Brettstellung extrahiert, bevor sie dargestellt wird. Damit sie die beschränkten Ressourcen des Smartphones nicht auslastet, wird diese extern ausgeführt.

Welcher Spieler am Zug ist, muss eine einstellbare Option sein. Aus der Stellung ist nicht ersichtlich, ob Schwarz oder Weiß am Zug ist, wenn mit Handicap gespielt wird. Der Nutzer muss den Spieler auswählen können, der am Zug ist.

Die Überlastung der Analyse-API muss verhindert werden. Die Analyse-API wird mit Go-Bots geteilt, die auf Servern spielen. Sie darf nicht durch unnötige, zu häufige oder duplizierte Anfragen überlastet werden, auch wenn der Betreiber für ihre Skalierung verantwortlich ist.

4.3.3 Benutzerinteraktion und Darstellung

Die Benutzeroberfläche sollte einfach und intuitiv sein. Um das schnelle Verstehen der Analyseergebnisse zu fördern, sollte die UI minimalistisch und auf das Wesentliche reduziert sein. Die Sicht auf das Go-Brett sollte nicht durch UI-Elemente behindert werden.

Der aktuelle Punktstand und der beste nächste Zug müssen dargestellt werden. Die berechneten Analyseergebnisse müssen dem Nutzer dargestellt werden.

Der beste Zug sollte augmentiert dargestellt werden. Dies ist, wie zuvor beschrieben, wichtig für eine immersive Analyseerfahrung, ohne die Notwendigkeit, das Spiel physisch zu verlassen.

4.3.4 Umgebungsanforderungen

Die App sollte auf Geräten mit Android API 26 oder höher funktionieren. Damit wird sie auf über 92% der Android-Geräte lauffähig sein.

Es muss eine funktionierende Kamera mit Zugriffsrechten zur Verfügung stehen. Ohne funktionierende Kamera auf dem Smartphone des Benutzers oder Zugriffsrechten für die App kann kein Brett visuell analysiert werden.

Eine Internetverbindung ist erforderlich. Dies ist notwendig zur Kommunikation mit der externen Analyse-API.

Die App sollte auf Geräten mit begrenzten technischen Spezifikationen lauffähig sein. Um die App einer breiten Masse zugänglich zu machen, sollte sie auch auf schwächeren Geräten laufen.

Die Sicht auf das Go-Brett muss frei von Objekten sein. Die Stellung kann nur erkannt werden, wenn die Sicht darauf nicht eingeschränkt ist. Es muss sich um übliche Go-Sets handeln. Es müssen nur schwarze und weiße Steine und ein holzfarbenedes Brett unterstützt werden. Andere Spielsets sind sehr ungewöhnlich.

4.3.5 Stabilität und Sicherheit

Die Reaktionszeit der App sollte nicht mehr als 5 Sekunden betragen. Die Reaktionszeit der App sollte für eine nahtlose Benutzererfahrung optimiert werden, wobei die Analyse nicht länger als 5 Sekunden dauern darf. Bei Überschreitung dieser Zeit geht der Nutzer davon aus, dass die App nicht richtig funktioniert.

Die App sollte stabil laufen und frei von Abstürzen sein. Eine abstürzende App kann ihre Funktion nicht erfüllen.

Präventive Maßnahmen gegen Speicherlecks sind erforderlich. Speicherlecks können die Anwendung verlangsamen oder zum Absturz bringen.

Der Datenschutz sollte angemessen sein. Die App sollte keine personenbezogenen Daten verarbeiten und keine Daten speichern. Die externe Analyse erfolgt ausschließlich auf abstrakten Go-Spielstellungen.

4.4 Beitrag zur Forschung und aktuellen Trends

Ein Forschungsbeitrag dieses Projekts liegt in der Einführung der Nutzung von Farbinintensitätsanalysen zur Reduzierung von Reflexionen auf dem Go-Brett. Durch die Messung der Farbinintensität können Bereiche, die durch Lichtreflexionen weißer erscheinen, effektiv identifiziert und angepasst werden. Das verbessert die Genauigkeit der Stein- und Brettlinienerkennung. Diese Methode, die bei wechselnden Lichtverhältnissen hilfreich ist, stellt eine bedeutende Abweichung von bisherigen Ansätzen dar.

Eine weitere Innovation liegt in der Positionserkennung des Brettes. Durch die Verwendung eines Filters für dünne Linien werden die Gitterlinien separiert. Nach dieser Hervorhebung wird die größte Kontur dieser dünnen Linien selektiert, um die Position des Brettes zu identifizieren.

Zusätzlich trägt das Projekt zur Forschung bei, indem es als erste Anwendung Augmented Reality für die unmittelbare Darstellung analysierter Go-Züge nutzt.

Diese Methodiken helfen zukünftigen Forschungen und Anwendungen, nicht nur in Go, sondern auch in weiteren Bereichen, in denen Computer Vision eingesetzt wird.

4.5 Konzept der Benutzererfahrung

Die App ist auf intuitive und minimalistische Interaktionen ausgerichtet. Für die Analyse muss der Nutzer nur die Kamera auf das Brett halten. Das System extrahiert dann automatisch die zur Analyse benötigten Informationen aus dem Bild und stellt dem Nutzer die Analyseergebnisse augmentiert dar. Dazu wird ein Indikator an der Stelle eingeblendet, an der der beste Zug zu finden ist. Zusätzlich wird der aktuelle Spielstand auf der Tischebene neben dem Brett als 3D-Text eingeblendet. Wichtig für die Analyse ist es, den Spieler, der am Zug ist, festzulegen. Dazu kann der Nutzer einen kleinen Kreis in der Farbe des zu ziehenden Spielers berühren, um diesen zu ändern.

4.6 Systemarchitektur (Überblick)

Das System ist modular aufgebaut und besteht aus mehreren Hauptkomponenten, die miteinander interagieren. Im Folgenden wird ein Überblick über diese Komponenten in Abbildung 4.1 gegeben, ohne dabei zu tief ins Detail zu gehen.

4.6.1 Betriebssystem-API

Diese Ebene dient als Schnittstelle zwischen der Betriebssystemumgebung und der Anwendung. Sie bietet eine Abstraktion der Systemfunktionen, wie der Kamera, sodass die Anwendung unabhängig von spezifischer Hardware funktioniert.

4.6.2 Kamera-Interface

Das Kamera-Interface stellt das Bindeglied zwischen der Betriebssystem-API und der Bildverarbeitung dar. Neue Kamerabilder werden hier erfasst.

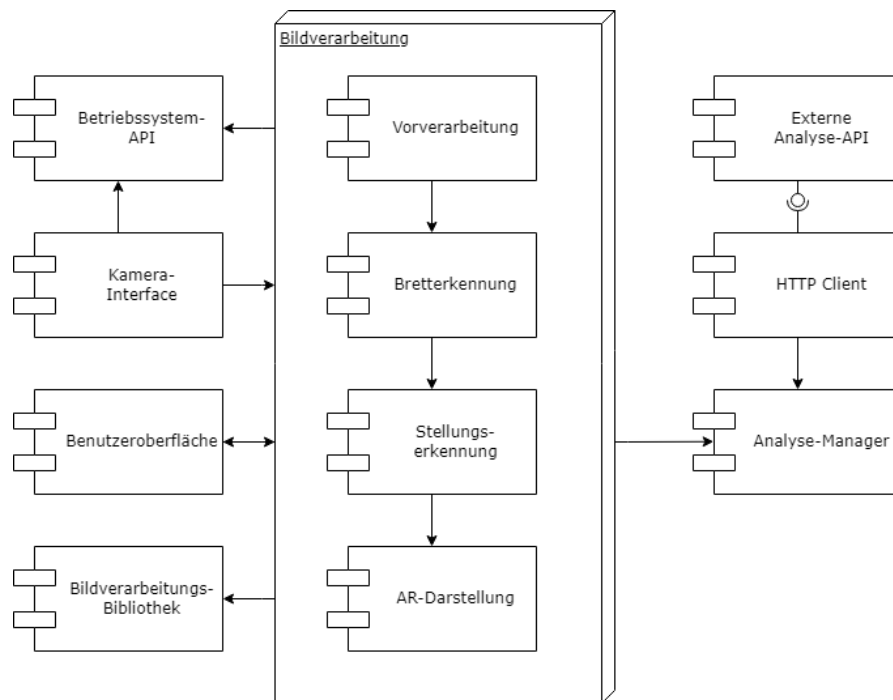


Abbildung 4.1: Überblick über die geplanten Komponenten

4.6.3 Benutzeroberfläche

Die Benutzeroberfläche dient als Schnittstelle zum Nutzer. Hier kann er die augmentierte Darstellung der Ergebnisse sehen, aber auch den Spieler wählen, der am Zug ist.

4.6.4 Bildverarbeitung

Vorverarbeitung: Hier wird das Bild für die weitere Verarbeitung vorbereitet. Das Ziel ist es, einen Output zu generieren, der von den Umgebungsverhältnissen, wie z.B. der Beleuchtung, unabhängig ist.

Bretterkennung: Die Aufgabe dieses Moduls ist es, das Brett zu erkennen und zu entzerren, sodass die nächsten Schritte der Pipeline unabhängig von der Perspektive auf das Brett ausgeführt werden können.

Stellungserkennung: Hier wird die Go-Stellung des entzerzten Brettes erkannt, indem die Steinpositionen extrahiert werden. Hierzu ist es relevant zu wissen, wer am Zug ist. Die gefundene Stellung wird dann zur Analyse weitergereicht.

AR-Darstellung: Falls die erkannte Stellung bereits analysiert wurde, werden die AR-Elemente hier in das Originalbild eingearbeitet und das Endergebnis bereitgestellt.

4.6.5 Bildverarbeitungs-Bibliothek

Hier werden grundlegende Bildverarbeitungsfunktionen zur Verfügung gestellt, die von der Bildverarbeitung genutzt werden.

4.6.6 Analyse-Manager

Hier wird der Analyseprozess verwaltet. Andere Komponenten können hier neue Go-Stellungen zur Analyse ablegen, ihren Analysestatus abfragen oder die vorliegenden Ergebnisse erhalten. Die Analysekomponenten können sich hier Go-Stellungen abholen, die sie dann bearbeiten und letztendlich ihre Ergebnisse mitteilen.

4.6.7 HTTP Client

Dieser Client sendet zu analysierende Go-Stellungen an die externe Analyse API und teilt dann die Ergebnisse nach Erhalt mit.

4.6.8 Externe Analyse API

Diese API kümmert sich um die Analyse der Go-Stellungen. Sie nutzt zur Analyse neuronale Netze.

4.7 Auswahl der Technologien

Die Auswahl der Technologien ist für die Entwicklung neuer Anwendungen ein entscheidender Prozess. Sie legt den Grundstein für die Funktionalität des Endprodukts.

4.7.1 Plattform: Android

Android ist die Plattform mit der größten Nutzerbasis auf dem mobilen Markt. Es eignet sie ideal für eine möglichst große Verbreitung der App. Die Erfahrung im Umgang des Autors mit Android-Geräten erleichtert den Entwicklungsprozess. Android zeichnet sich auch durch eine große Entwicklerfreundlichkeit aus, insbesondere für Neueinsteiger in der mobilen Anwendungsentwicklung, da eine große Community von Entwicklern ihr Wissen teilt. Für den Autor ist es auch relevant, dass er bereits über ein Android-Gerät verfügt und damit keine zusätzlichen Kosten durch die Anschaffung eines Gerätes mit einer anderen Plattform anfallen.

4.7.2 Computer Vision: OpenCV

OpenCV ist eine weit verbreitete Bibliothek mit einer großen Bandbreite an Computer-Vision-Funktionen. Der Autor hat bereits Erfahrungen mit dieser Bibliothek gesammelt, was den Entwicklungsprozess erleichtert. Auf die Entwicklung in Echtzeit ist es spezialisiert und bietet daher viele optimierte Funktionen. Es verfügt über eine große Community und profitiert von einer umfangreichen Dokumentation mit vielen Tutorials und Beispielen. Die Verwendung ist kostenlos und der Code ist Open-Source, wodurch fehlende Funktionalitäten sogar selbstständig hinzugefügt werden können.

4.7.3 Programmiersprache: Java

Java ist die bevorzugte Sprache für die Entwicklung auf Android-Geräten. Der Autor ist mit dieser Sprache bereits gut vertraut, was die Bewältigung komplexer Aspekte erleichtert und die Entwicklungszeit verkürzt. Zudem ist es möglich, eine einheitliche Sprache für das gesamte Projekt zu nutzen, da sie auch für OpenCV verwendet werden kann. Die Performance von Java-Code ist für Echtzeitanwendungen nicht optimal, jedoch werden die rechenintensiven Operationen im in C geschriebenen OpenCV-Code ausgelagert, was

weiterhin Effizienz gewährleistet. Die Community-Unterstützung ist groß, wodurch eine Menge an Beispielcode und Hilfestellungen für Entwickler verfügbar sind.

4.7.4 Augmented Reality: Eigenentwicklung ohne ARCore

Durch die Vermeidung des Industriestandards ARCore wird die Anwendungsentwicklung vereinfacht und entschlackt. Zusätzlicher Overhead wird vermieden und die Anwendung wird kleiner. Die Analyse von statischen Bildern erfordert keine Erkennung im 3D-Raum. Lediglich für die visuelle Darstellung der Analyse wäre ARCore hilfreich. Durch eine einfache Darstellung durch Layering-Methoden kann jedoch auch mit OpenCV eine ansprechende visuelle Darstellung mit einfachen Mitteln erreicht werden.

4.7.5 Spielanalyse: Externe API

Wie bereits im Kapitel 3.10 besprochen wurde, ergeben sich durch die Verwendung eines neuronalen Netzes zur Spielanalyse, das auch den Score evaluieren kann, erhebliche Vorteile für das Lernverständnis der Nutzer. Die Nutzung eines neuronalen Netzes ist für ein mobiles Endgerät jedoch sehr rechenaufwendig. Eine externen API zur Spielanalyse ist also sinnvoll, um Ressourcen zur Bildverarbeitung zu sparen. Auch Probleme der Skalierung und der Zuverlässigkeit werden auf eine Third-Party ausgelagert und müssen nicht weiter beachtet werden. Da die später verwendete externe API bereits Verwendung für Go-Bots zur Analyse auf Go-Servern hat, verlässt man sich hier auch auf ein bereits erprobtes System.

4.8 Herausforderungen und Lösungsstrategien

Die Bildanalyse beim Go-Spiel stellt aufgrund der Komplexität des Spielfeldes und der Spielsteine mehrere Herausforderungen dar. Eine Herausforderung besteht darin, das Bild von perspektivischen Verzerrungen zu befreien, die durch Aufnahmen aus verschiedenen Winkeln entstehen. Zur Beseitigung wird ein Algorithmus zur Perspektivkorrektur angewendet, um das Bild in eine Art Vogelperspektive umzuwandeln. Um diese Korrektur durchzuführen, können die Eckpunkte des Brettes auf ein Quadrat transformiert werden.

Diese Eckpunkte des Brettes sollen automatisch erkannt werden. Dazu kann die äußere Kontur der Gitterlinien verwendet werden (3.6). Die Gitterlinien können vom Bild separiert werden, indem ein adaptiver Threshold auf das Originalbild angewendet wird und dann nur feine Konturen herausgefiltert werden (3.5). Diese Methode muss auch bei verschiedenen Lichtverhältnissen funktionieren. Durch Reflexionen können Bereiche eines Brettes aufgehellt werden. Um die Gitterlinien darüber hinweg besser zu separieren, werden Farbinformationen zur Hilfe genommen. Das Go-Brett ist normalerweise holzfarben, während Steine und Gitterlinien schwarz und weiß sind.

Ein weiteres Herausforderung besteht in der korrekten Identifikation und Separierung von Spielsteinen auf dem Brett. Bei der Unterscheidung von schwarzen und weißen Steinen werden Farb- und Formerkennung-Techniken sowie verbesserte Schwellenwertverfahren verwendet, die insbesondere bei ungleichmäßiger Beleuchtung einen Mehrwert bieten. Das Helligkeitsspektrum von schwarzen und weißen Steinen kann sich überlappen. Durch das Abziehen der Hintergrundhelligkeit werden die Helligkeitswerte von schwarzen und weißen Steinen durch die Umgebung angepasst. Es kann auch zur Bildung eines großen Steinclusters kommen, der schwer zu separieren ist. Hierbei wird die Größe einzelner Steine einbezogen, um diese aufzulösen.

Objekte neben dem Brett dürfen die Detektion nicht verschlechtern, besonders Steine in den Bowls der Spieler. Eine Strategie hierzu ist es, den Analysebereich auf die größte zentral zu findende Kontur zu beschränken, was üblicherweise dem Gitternetz des Brettes entspricht.

Eine weitere Herausforderung ist die Notwendigkeit einer schnellen Bildverarbeitung in Echtzeit. Die Anwendung kann durch die Optimierung von Code verbessert werden. Doch selbst mit optimierten Algorithmen kann die Verarbeitung der Analyse Zeit in Anspruch nehmen. Um dies zu adressieren, ist die Implementierung einer asynchronen Analyse unerlässlich. Durch die Einrichtung separater Threads zur Kommunikation mit der API bleibt die Anwendung reaktionsfähig, während die Stellung analysiert wird. Die Anwendung kann weiterlaufen und die Ergebnisse dann darstellen, wenn sie verfügbar sind.

Auch das Darstellen der Analyseergebnisse führt zu Herausforderungen. Um eine immersive Darstellung in AR zu ermöglichen, müssen die Ergebnisse unter Berücksichtigung der Original-Perspektive dargestellt werden. Dies wird erreicht, indem Markierungen im unverzerrten Raum eingefügt und dann in den Original-Raum transformiert werden. Durch eine Verschiebung auf verschiedenen Ebenen in vertikaler Richtung wird die Illusion eines 3D-Effekts erzeugt.

Die enormen Rechenressourcen und Kosten, die mit dem Training eines Analyse-Netzwerks einhergehen, werden durch die Nutzung eines vortrainierten Netzwerks überwunden. Netzwerke wie AlphaGo neigen jedoch dazu, suboptimale Züge in speziellen Spielsituationen zu machen (3.9), insbesondere in Handicap-Partien, da diese auf Grundlage von Gewinnwahrscheinlichkeiten operieren. Durch die Nutzung des KataGo-Netzwerks kann diese Schwäche jedoch beseitigt werden, da dieses zusätzlich über ein Bewertungsnetzwerk verfügt. Dieses Netzwerk trägt dazu bei, suboptimale Züge zu vermeiden, da es auf einer Kombination von Punktebewertung und Gewinnwahrscheinlichkeit basiert (3.10).

Die Herausforderungen dieses Projektes lassen sich also mit einer Kombination aus fortschrittlichen Bildbearbeitungstechniken und moderner KI lösen, um eine reibungslose Analyse des physischen Go-Spiels zu ermöglichen.

4.9 Risikobewertung und Qualitätssicherung

Bei einem so komplexen Projekt, in dem neue Ansätze erprobt werden, ist es wichtig, das Risiko zu bewerten und die Qualitätssicherung zu planen. In diesem Kapitel werden potenzielle Risiken untersucht, die mit dem Konzept verbunden sind, und es werden Strategien aufgezeigt, die die Zuverlässigkeit der Anwendung gewährleisten.

4.9.1 Unsicherheiten bei eigenen Ansätzen

In diesem Projekt werden einige neue Ansätze verfolgt, wie die Berücksichtigung von Farbinformationen oder der Bretterkennung anhand der Gitterlinienkontur. Auch wenn dies Möglichkeiten zur Innovation bietet, birgt es auch Unsicherheiten. Es steht nicht fest, wie zuverlässig diese neue Methode funktioniert. Um das Risiko zu minimieren, wird der Entwicklungsprozess geplant und dokumentiert. Die Funktion wird regelmäßig überprüft und bewertet. Es besteht auch weiterhin die Möglichkeit, auf Lösungen bestehender Arbeiten zurückzufallen, welche jedoch ihre eigenen Probleme mit sich bringen.

4.9.2 Detektionsfehler und die Benutzererfahrung

Durch den Ansatz, die Stellung automatisch zu erkennen und das Analyseergebnis augmentiert darzustellen, ergibt sich das Problem, dass der Nutzer nicht verifizieren kann,

ob die Stellung richtig erkannt wurde und ob der vorgeschlagene Zug daher Sinn macht. Dies ist ein Kompromiss, der bewusst gewählt wurde, um die Immersion für den Nutzer maximal zu halten. Ein Debug-Modus zum Anzeigen der erkannten Stellung könnte zur Verifikation helfen.

4.9.3 Technologische Abhängigkeiten

Die Entwicklung auf der Android-Plattform bedeutet, dass Änderungen in der API das Projekt behindern können. Eine flexible Architektur mit modularen Komponenten hilft, dieses Risiko zu beherrschen. Auch die Abhängigkeit von einer Third-Party-Analyse-API birgt ein Risiko, da die Anwendung nicht mehr funktioniert, sobald die API offline geht. Es ist daher wichtig, diesen Programmteil modular zu gestalten, sodass diese Schnittstelle jederzeit durch eine andere ersetzt werden kann.

4.9.4 Qualitätssicherung

Es gibt einen speziellen Debug-View, der alle Schritte der Bearbeitungspipeline darstellen kann. So können Bretter unter verschiedenen Blickwinkeln und Lichtbedingungen getestet und die Probleme, die sich dadurch ergeben, detektiert werden. Obwohl diese Arbeitsweise arbeitsintensiv ist, ermöglicht sie eine tiefere Analyse und Anpassung an unvorhergesehene Probleme. Ergänzend wurde ein automatischer Test implementiert, der ein Set von fünf Bildern beinhaltet. Dieser Test kann manuell gestartet werden, wobei die Ergebnisse mit einer festgelegten Baseline verglichen werden. Diese Bilder entsprechen nicht dem Testset, das zum Zwecke der Evaluation verwendet wurde.

5 Implementierung und Tests

5.1 Technische Grundlagen und Entwicklungsumgebung

Entwicklungsumgebung: Android Studio

Die App wurde in Android Studio entwickelt. Es ist die offizielle Entwicklungsumgebung für Android. Android Studio bietet eine einsteigerfreundliche, intuitive Codeumgebung mit guten Debugging-Tools. Es gibt guten Support und eine große Nutzerbasis, sodass eine effiziente Entwicklung der App mit einer integrierten Android SDK möglich ist.

Ziel-Android API: API 26+

Die App zielt auf API-Level 26 und höher ab. Hardware, die älter ist, ist üblicherweise nicht leistungsfähig genug, um diese App auszuführen. Gleichzeitig ermöglicht es die Nutzung einiger fortgeschrittener Funktionen und Berechtigungen, die die Entwicklung einfacher machen.

Bildverarbeitung: OpenCV 4.8.0

OpenCV ist die führende Bibliothek im Bereich der Computer Vision und wird für das Kamera-Interface und die Bildverarbeitungsfunktionen genutzt. OpenCV 4.8.0 ist die zu Entwicklungsbeginn neueste zur Verfügung stehende Version.

Build-System: Gradle 7.4.2

Gradle wird für die Automatisierung des Build-Prozesses verwendet. Version 7.4.2 ist die neueste, die von OpenCV 4.8.0 unterstützt wird.

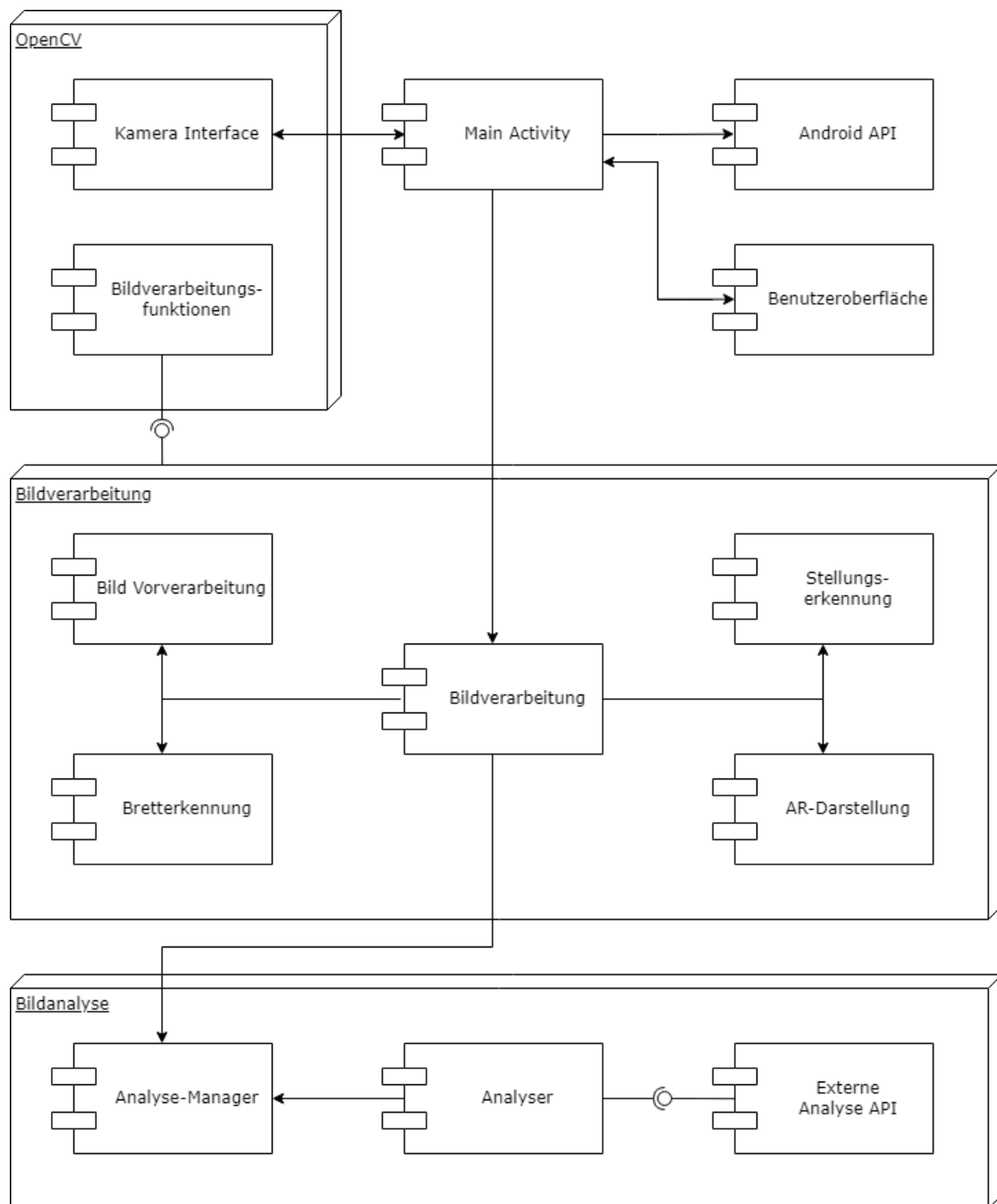


Abbildung 5.1: Die implementierte Systemarchitektur

5.2 Implementierte Systemarchitektur

5.2.1 Datenfluss und Verarbeitungslogik

Abbildung 5.1 gibt einen Überblick über die Architektur des Systems. Der Einstiegspunkt der App und ihre zentrale Steuerungseinheit ist die Main Activity. Sie kommuniziert mit der Android API und fragt dort unter anderem Berechtigungen an, wie jene zur Verwendung der Kamera. Die GUI-Komponenten werden von hier eingerichtet und verwaltet. Sie initiiert und verwaltet auch das Kamera-Interface von OpenCV und liefert eine Callback-Funktion `onCameraFrame`, welche die Bildverarbeitungs pipeline startet.

Als OpenCV Kamera-Interface wird eine `CameraBridgeViewBase` genutzt. Sie ruft die Callback-Methode der Main Activity auf, wenn ein neuer Frame zur Verfügung steht und die Bearbeitung des vorherigen bereits abgeschlossen ist. Frames, deren Bearbeitung abgeschlossen wurde, werden über die Main Activity im Userinterface dargestellt.

Zu analysierende Frames werden an die Bildverarbeitung weitergeleitet. Die Bildverarbeitung nutzt die Funktionen von OpenCV und kümmert sich in einer Pipeline um die Vorverarbeitung der Kameraframes, die Erkennung des Spielbrettes und der Go-Stellungen. Sie stellt dem Analyse Manager Go-Stellungen zur Analyse bereit und stellt die Ergebnisse dar.

Der Analysemanager verwaltet zu analysierende Go-Stellungen und ihre Analyseergebnisse und wird zu Programmbeginn von der Main Activity erstellt. Die Analyser leiten die Go-Stellungen vom Manager an eine externe Analyse API weiter.

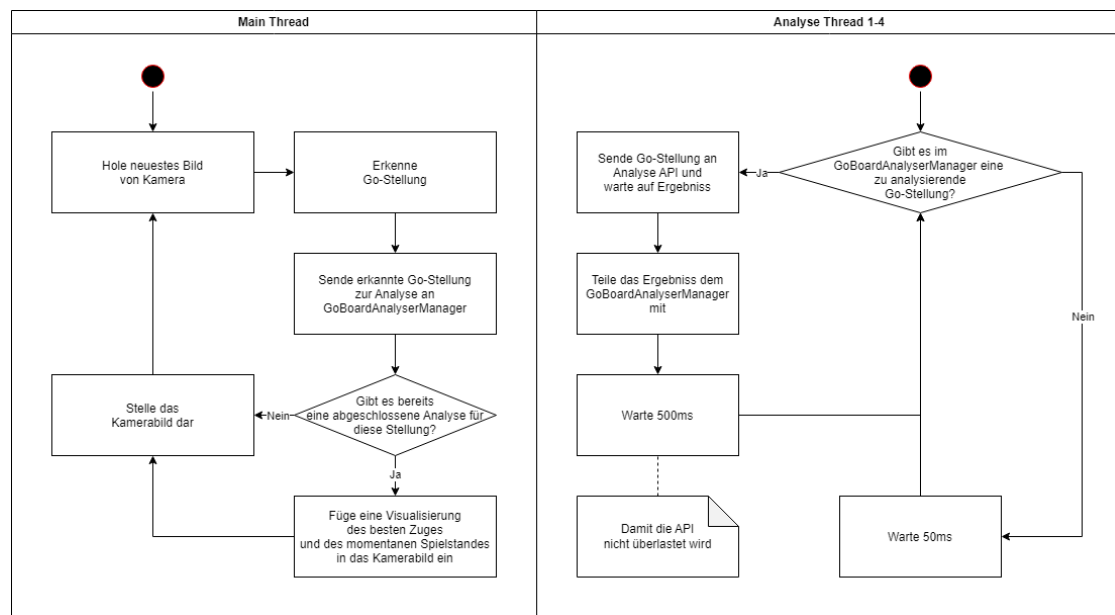


Abbildung 5.2: Der Verarbeitungszyklus des Main- und der Analyse-Threads

5.2.2 Asynchrones Thread Management

Da die Analyse der Stellungen über die externe API lange dauert, findet die Analyse asynchron und parallel statt. Innerhalb der Main Activity wird ein Executor-Service mit vier Threads erzeugt. Jeder dieser Threads führt einen Analyser aus, der die Spielanalyse vornimmt. In Abbildung 5.2 sind die Verarbeitungszyklen der Threads dargestellt. Der Main-Thread ist dafür zuständig, das Kamerabild zu verarbeiten. Bei jedem neuen Kameraframe wird die Go-Stellung erkannt und zur Analyse an den Analysemanager weitergegeben. Danach werden die Analyseergebnisse, falls vorhanden, in das Bild eingearbeitet und das Ergebnis dargestellt. Die Analyse-Threads pollen zu analysierende Go-Stellungen vom Manager. Wenn Stellungen vorhanden sind, werden sie zur Analyse an die externe API weitergeleitet und die Ergebnisse anschließend dem Manager mitgeteilt.

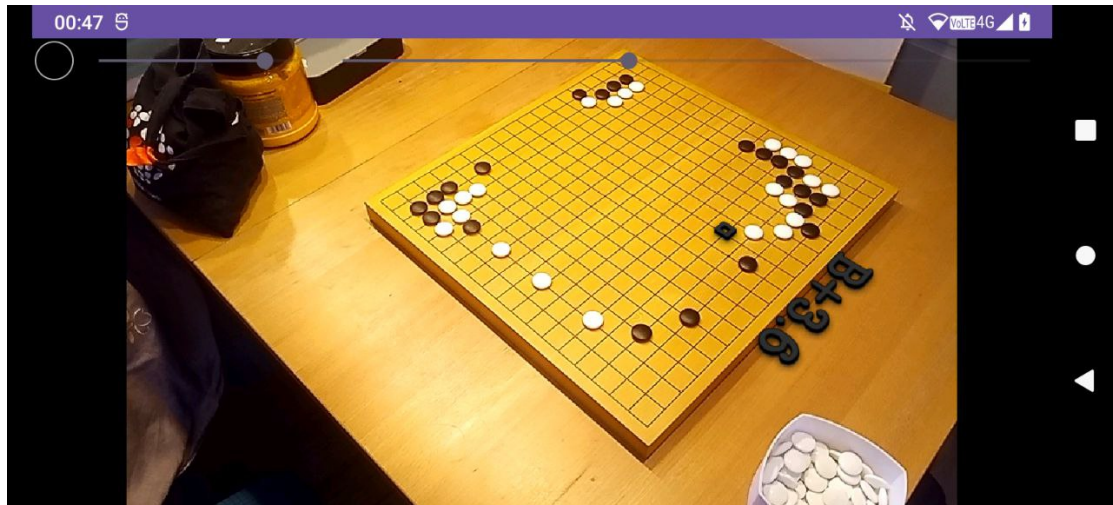


Abbildung 5.3: Das Interface mit Debug Menü

5.3 Implementierung des User Interface

Das Userinterface ist sehr einfach gehalten und in Abbildung 5.3 zu sehen. Der Status aller Elemente wird in einer Klasse mit statischen Feldern festgehalten, auf welche von überall zugegriffen werden kann. Es gibt einen Kameraview, der die bearbeiteten Kameraframes darstellt und über die Main Activity vom CameraBridgeViewBase Kamera Interface betreut wird. Als zweites Element gibt es einen Button, mit der Farbe des zu ziehenden Spielers. Dies ist ein selbst erstelltes UI-Element, das über einen ImageView verfügt und in einer Klasse ChooseColorView verwaltet wird. Die Main Activity verfügt über einen OnClickListener für das Element und vertauscht bei Aktivierung die Farben. Im Weiteren gibt es noch zwei Seekbars, um einen Debug-Modus zu unterstützen, der alle Zwischenschritte der Verarbeitung anzeigen kann. Diese sind aber nicht Teil des finalen Produkts.

5.4 Implementierung der Computer-Vision-Technologie

In diesem Unterkapitel wird erklärt, welche Computer-Vision-Techniken angewendet werden, um die auf dem Bild abgebildete Go-Stellung zu abstrahieren. Grundsätzlich handelt es sich hierbei um eine umfassende Bildverarbeitungs pipeline, die im folgenden von Anfang bis Ende dokumentiert und erklärt wird. Hauptsächlich wird dazu der Java-Wrapper der OpenCV-Bibliothek genutzt, um diese zu verarbeiten.

Im Folgenden wird ein grundlegendes Verständnis der Fachbegriffe und Konzepte der Computer Vision vorausgesetzt. Für eine detaillierte Einführung in diese Themen empfiehlt sich das Buch “Image Processing, Analysis and Machine Vision” von Sonka, Hlavac und Boyle, das eine umfassende Grundlage bietet (Sonka u. a., 2013).

5.4.1 Vorverarbeitung

Die Vorverarbeitung ist ein grundlegender Schritt der Bildverarbeitung. Die Hauptaufgabe ist es, eine Basis zu schaffen, die in den folgenden Teilen verwendet werden kann, indem die Rohdaten verbessert werden. Die Ergebnisse sollten die Brett- bzw. Gittererkennung, die Erkennung der weißen und der schwarzen Steine vereinfachen. Grundlegend ist zu beachten, dass in den meisten Smartphones bei der Nutzung der Kamera bereits eine Vorverarbeitung der Bilder stattfindet. Eine grundlegende Rauschunterdrückung, Schärfung, Kontrastverbesserung und Linsenkorrektur der Bilder ist also meistens schon vorgenommen worden, ohne dass man diese beeinflussen kann. In diesem Teil wird der Fokus auf Techniken gelegt, die speziell auf die hier zu bearbeitende Problemstellung angepasst sind.



Abbildung 5.4: Das Graubild vor und nach der Normalisierung

Grundlegende Verarbeitungsschritte

Zuallererst wird das Bild in ein Graustufenbild umgewandelt. Auf nur einem Kanal zu arbeiten, reduziert die Datenkomplexität und ist dadurch deutlich effizienter. Auch viele der im Folgenden verwendeten Algorithmen sind nur auf Graustufenbildern möglich. Danach wird das Bild normalisiert, sodass es das gesamte Helligkeitsspektrum einnimmt. Dieser Prozess ist in Abbildung 5.4 dargestellt.

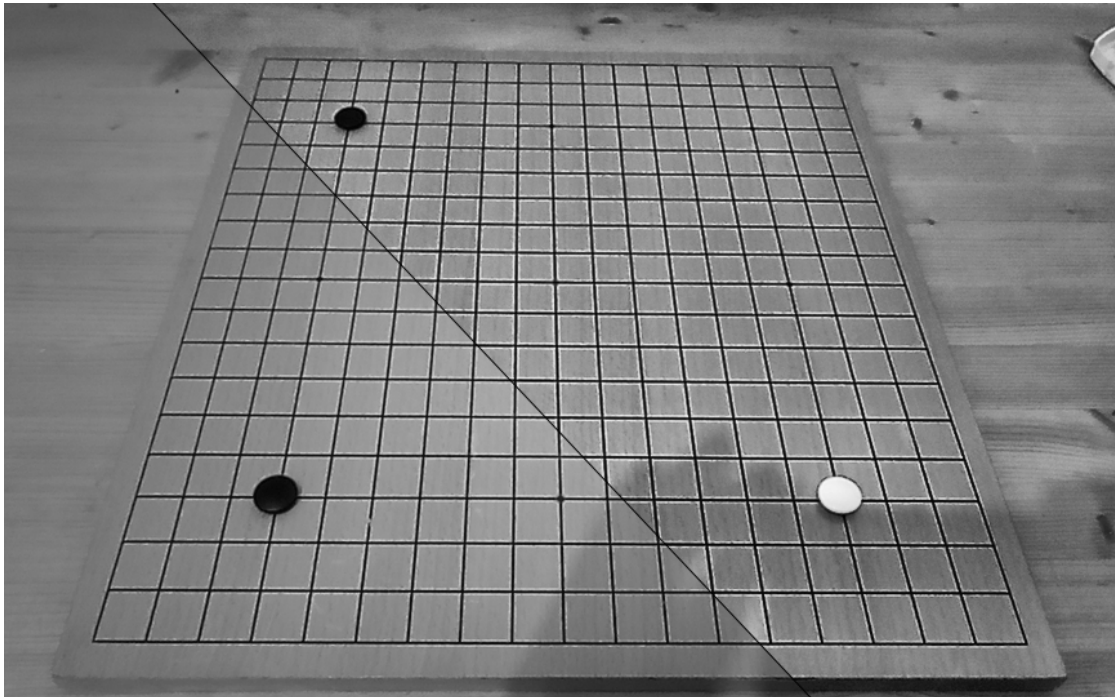


Abbildung 5.5: Das Graubild vor und nach einem Adaptiver Histogrammausgleich

Adaptiver Histogrammausgleich

Die Anwendung eines adaptiven Histogrammausgleichs ist ein sehr gängiger Arbeitsschritt an dieser Stelle der Pipeline. Hierbei werden Die Intensitätswerte des Originalbildes so angepasst, dass sie stattdessen gleichmäßigen Verteilt sind. Dadurch erhofft man sich eine Verbesserung des Bildkontrasts und eine größere Unabhängigkeit von den Beleuchtungsverhältnissen. Das Ergebnis der Anwendung ist in Abbildung 5.5 zu erkennen. Unerwarteterweise verschlechtert diese Technik das Bild für unsere Anwendung beträchtlich. Das Problem besteht darin, dass Go-Bretter bereits eine sehr einheitliche Farbe aufweisen und der adaptive Histogrammausgleich hauptsächlich bei Bildern angewendet wird, die starke Kontrastunterschiede aufzeigen. Infolgedessen werden Schatten verstärkt, die Maserung des Holzes betont und Helligkeitsunterschiede hervorgehoben. Somit bewirkt der Histogrammausgleich das Gegenteil von dem, was in der Vorverarbeitung für die nachfolgenden Schritte erzielt werden soll und wird nicht verwendet.

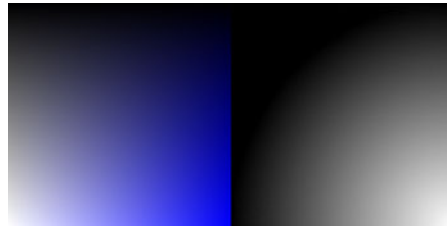


Abbildung 5.6: Die SV-Ebene links und ihre definierte Farbintensität rechts

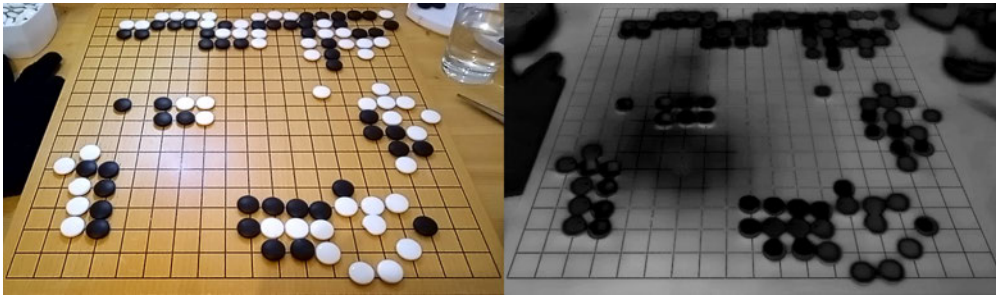


Abbildung 5.7: Das Farbbild des Go-Sets links und seine Farbintensität rechts

Nutzung der Farbinformationen

Um die Steine und die Gitterlinien vom Brett zu separieren, müssen Kontraste verbessert und Reflexionen reduziert werden. Hierzu wird ein einzigartiger Ansatz implementiert, der auf Farbinformationen basiert. Das Go-Brett unterscheidet sich von den Steinen hauptsächlich durch seine Farbigkeit, während die Gitterlinien und Steine schwarz und weiß sind. Sie weisen nur leichte Verfärbungen aufgrund der Umgebungsbeleuchtung auf. Was für Menschen als farbig gilt, kann in der Informatik jedoch ganz anders definiert werden. Es reicht nicht aus, nur die Sättigung zu betrachten, da die Farbe Schwarz ohne sichtbare Änderungen sowohl eine minimale als auch eine maximale Sättigung aufweisen kann. Zur Bestimmung der Farbigkeit wird in dieser Arbeit eine Kombination aus Sättigung und Helligkeit im HSV-Farbraum verwendet. Je kleiner der euklidische Abstand eines Farbpunktes in der SV-Ebene zum Punkt (S_{max}, V_{max}) ist, desto farbiger wird ein Pixel betrachtet. Die Auswirkungen dieser Definition sind in Abbildung 5.6 sichtbar. Je farbiger ein Bereich für einen Menschen erscheint, desto heller wird er in der Transformation dargestellt.

In Abbildung 5.7 wird die Methode auf das Farbbild des Go-Sets angewendet. Wir erhalten Informationen darüber, welche Teile zum Brett gehören und keine Steine sind, und erkennen, wo sich Reflexionen befinden.

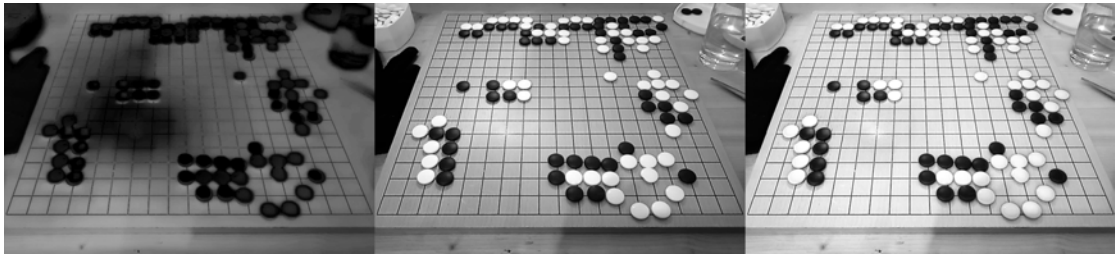


Abbildung 5.8: Links: Farbintensitätsbild,
Mitte: Originales Graustufenbild
Rechts: Ergebnis ihrer Addition

Erzeugung des Basisbildes für die Gittererkennung und der Erkennung schwarzer Steine

Ziel ist es, eine bestmögliche Basis für die Gittererkennung und die Erkennung schwarzer Steine zu schaffen. Hierbei muss der Kontrast zwischen ihnen und dem Bretthintergrund maximiert werden. Außerdem müssen Reflexionen auf dem Brett beseitigt werden. Die Informationen aus der Farbanalyse werden dafür genutzt. Zunächst wird das Farbintensitätsbild als Maske verwendet, um die Brettteile zu identifizieren, die nicht von Steinen oder Gitterlinien verdeckt sind. Dadurch kann die mittlere Helligkeit des reflektionsfreien Brettes ermittelt werden. Das ursprüngliche Graustufenbild wird nun mit dem gewichteten Farbintensitätsbild addiert. Der Einfluss der Farbintensität wird dabei durch die ermittelte mittlere Bretthelligkeit so gewichtet, dass der Bretthintergrund nahezu weiß wird.

Das resultierende Ergebnis ist nahezu frei von Reflexionen und eignet sich hervorragend als Basis für die Bretterkennung. Es ist in Abbildung 5.8 dargestellt. Der Grund, warum dies so gut funktioniert, ist, dass das Bild der Farbintensität nahezu ein Negativbild des Bretthintergrunds erzeugt. Aufgehellte Bereiche sind weniger farbig. Die Spielsteine und Gitterlinien bleiben von der Korrektur unberührt. Für abgedunkelte Bereiche des Spielbrettes funktioniert die Kompensation weniger gut, da sie ebenfalls weniger farbig sind. Dies ist im rechten Bereich der unteren Spielbrettkante zu sehen. Doch auch für diese Bereiche wird der Kontrast zu den Gitterlinien und Spielsteinen deutlich erhöht.

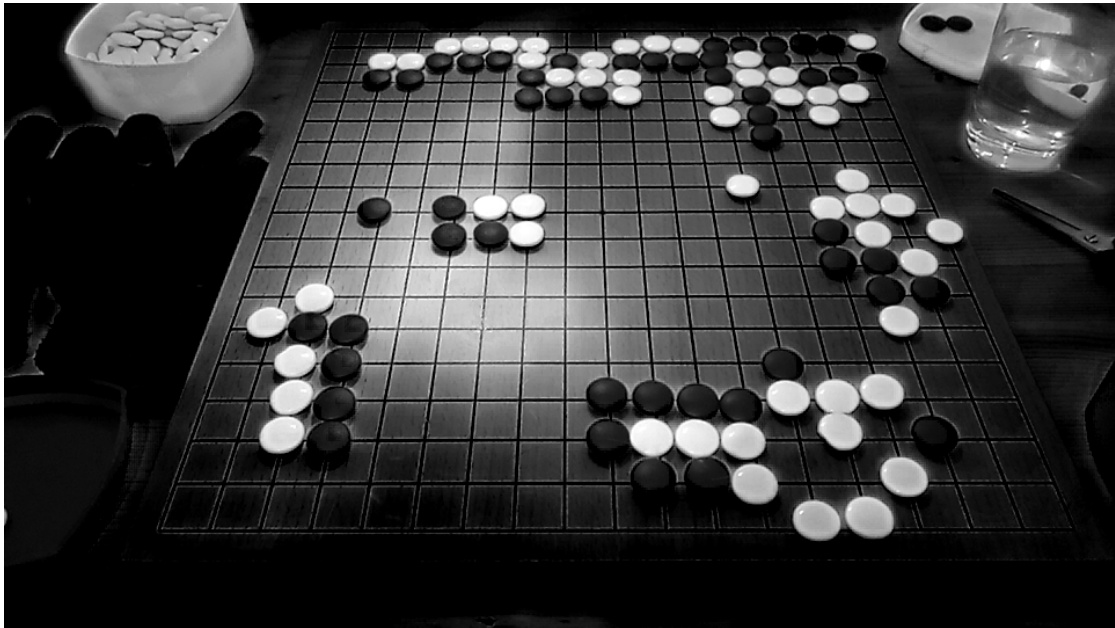


Abbildung 5.9: Ergebnis der Subtraktion der Farbintensität vom originalen Graustufenbild

Erzeugung des Basisbildes für die Erkennung weißer Steine

Analog verhält es sich mit der Erstellung des Basisbildes für die Erkennung der weißen Steine. Auch hier soll der Kontrast zwischen den Steinen und dem Brett erhöht werden. Anstatt das Farbintensitätsbild zum Graustufenbild zu addieren, wird es diesmal jedoch davon abgezogen. Dadurch wird der Bretthintergrund abgedunkelt.

Das Ergebnis ist in Abbildung 5.9 zu sehen. Für den Großteil des Bildes heben sich die weißen Steine nun stark vom Hintergrund ab. Allerdings werden die Reflexionen nicht kompensiert, sondern sogar verstärkt. Der Kontrast zu den weißen Steinen wird aber auch an diesen Stellen erhöht. Dieses Bild benötigt noch weitere Bearbeitungsschritte, um die Reflexionen zu beseitigen. Es bildet aber eine gute Basis.

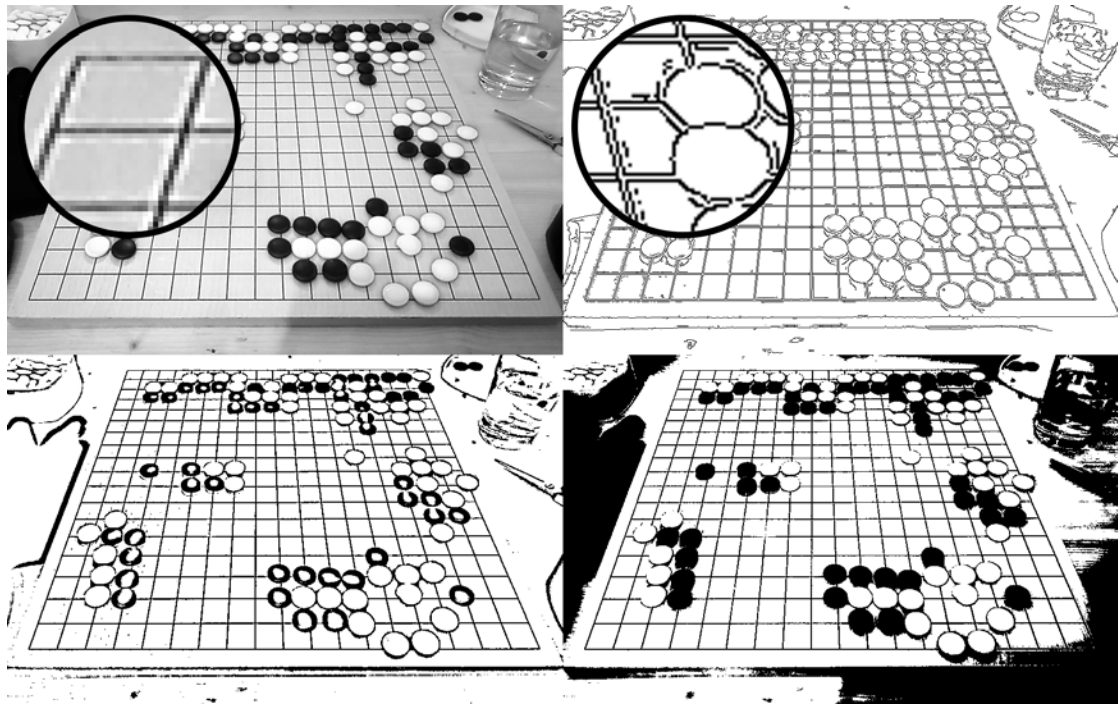


Abbildung 5.10: Mögliche Binärbildumwandlungen:
Links oben: Original,
Rechts oben: Canny Edge Detection,
Links unten: Adaptiver Schwellenwert,
Rechts unten: Statischer Schwellenwert

5.4.2 Erkennung der Ecken des Spielbrettes zur Entzerrung

Der Zweck der Spielbretterkennung ist es, die Perspektive zu entzerren und zu einer Art Vogelperspektive zu transformieren, um sie für die weitere Stellungserkennung vorzubereiten. Da das Brett ein Quadrat bildet, kann hierfür eine 4-Punkt-Transformation benutzt werden, sofern die Ecken des Spielbrettes bekannt sind. Das Ziel dieses Kapitels ist es daher, die 4 Ecken des Spielbrettes, bzw. hier die 4 Ecken des Gitters, zu erkennen. Die Hauptidee ist es, zunächst die Gitterlinien von dem Rest des Brettes zu separieren, dann die Außenkanten des Gitters zu erkennen und dadurch auf die Ecken rückzuschließen. Eines der Hauptprobleme besteht dabei darin, resistent gegen die Füllung des Spielbrettes mit Spielsteinen zu sein.

Überführung in ein Binärbild

Die Basis für diese Verarbeitungspipeline bildet das in der Vorverarbeitung erzeugte Basisbild für die Gittererkennung. Der erste Schritt besteht darin, dieses Graustufenbild in ein Binärbild umzuwandeln, in dem die Gitterlinien enthalten sind. Problematisch ist dabei, dass die Gitterlinien, die weit von der Kamera entfernt sind, sehr dünn sind. Durch die geringe Bildauflösung sind die Schwarz-Werte dort nicht mehr so intensiv. Gleichzeitig existieren noch Restreflexionen und Schattenwürfe im Bild, die die Schwarz-Werte des Gitters weiter beeinflussen.

Ein primitiver Ansatz mit einem statischen Schwellenwert ist daher nicht ausreichend, wie in Abbildung 5.10 zu erkennen ist. Die Gitterlinien sind sehr dünn oder zeigen kleinere Löcher. Teile des Bildes, die beschattet sind, überschreiten den Grenzwert, sodass die Information der Gitterlinien verloren geht. Die Nutzung einer Canny Edge Detection birgt ebenfalls Probleme. Die Gitterlinien, bestehen aus 2 Kanten. Dadurch ergibt sich eine doppelbändige Kontur der Linien, die nicht gefüllt ist und nicht trivial gefüllt werden kann, da die Kontur Löcher enthält. Die Nutzung eines adaptiven Schwellenwerts, wie er in 3.5 verwendet wird, sieht am vielversprechendsten aus. Hierbei wird der Schwellenwert nicht global für das gesamte Bild festgelegt, sondern im Verhältnis zu der lokalen Umgebung. Dadurch ist das Resultat unabhängig von den Lichtverhältnissen in diesen Bereichen. Es eignet sich vor allem, um dünne Elemente wie die Linien herauszufiltern. Das Ergebnis ist der Erhalt der vollständigen Dicke der Linien. Außerdem werden großflächige schwarze Flächen herausgefiltert, da sie sich nicht von der Umgebung absetzen. Für dieses Projekt wird daher diese Methode genutzt.

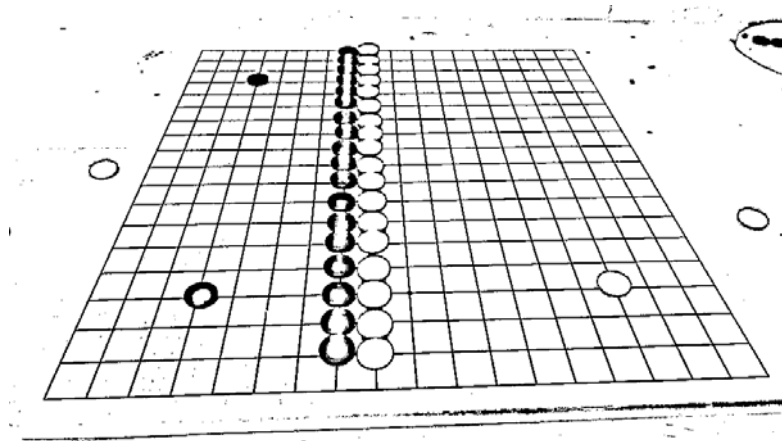


Abbildung 5.11: Gespielte Steine stellen die Linienerkennung vor eine Herausforderung

Hough-Transformation zur Erkennung von Linien

Die Hough-Transformation ist eine übliche Methode, um Linien in einem Bild zu erkennen. Vereinfacht dargestellt werden Geraden in der Polarkoordinatenform durch jeden Punkt gelegt und der Winkel durchiteriert. Die Anzahl der Bildpunkte, die pro Gerade dabei getroffen wird, wird akkumuliert. Umso mehr Punkte getroffen werden, desto besser repräsentiert die Gerade eine im Bild vorhandene Linie.

Wird dieser Ansatz direkt, wie in 3.5 auf das Brett angewandt, ist er jedoch nicht sehr erfolgreich bei der Detektion der Brettlinien, sobald das Brett mit Steinen gefüllt ist. Die Steine verdecken die Gitterlinien und bilden durch ihre Kanten neue Punkte, die nicht zu den Gitterlinien gehören. Abbildung 5.11 zeigt eine Situation, bei der die Konturen der Go-Steine eine starke Linie bilden, die keiner Gitterlinie entspricht. Das gezeigte Beispiel ist zum Verständnis überspitzt dargestellt, allerdings ist das Bilden von Mauern während des Spiels sehr üblich und das besetzen ganzer Spalten häufig.

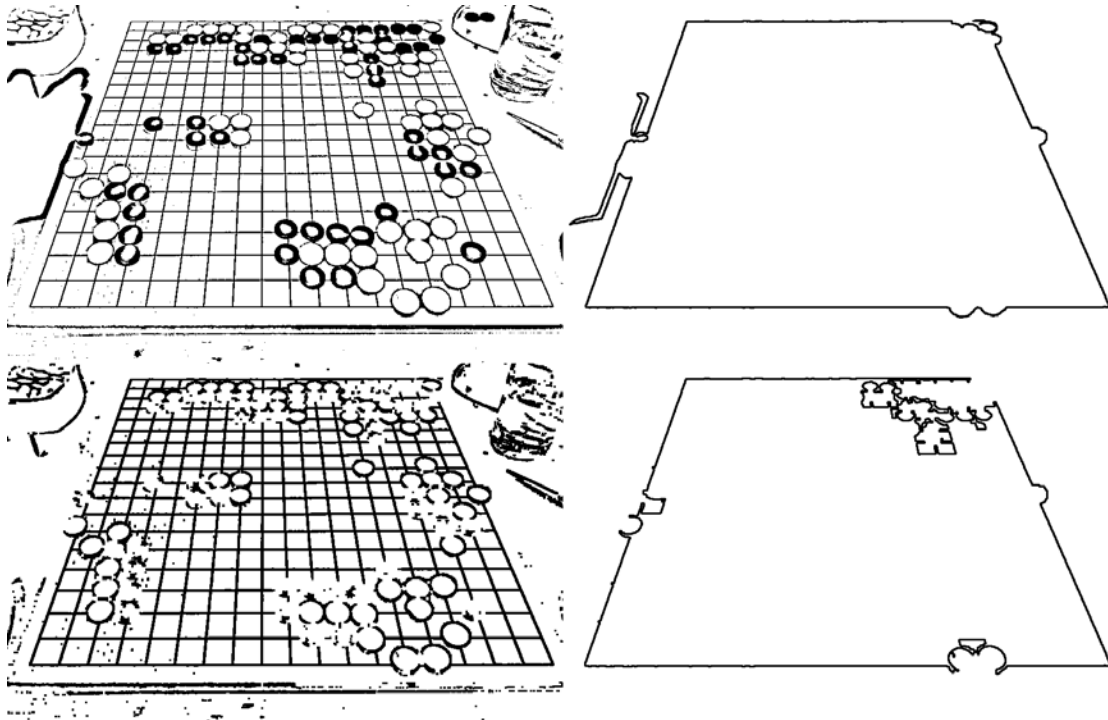


Abbildung 5.12: Konturen des Spielbrettes:
Links Oben: unbearbeitetes Binärbild,
Rechts Oben: größte Kontur des unbearbeiteten Brettes,
Links Unten: verbessertes Binärbild,
Rechts Unten: größte Kontur des verbesserten Binärbilds

Erkennung der Brettkonturen

Diese Arbeit nutzt eine alternative Strategie zur Erkennung des Go-Brettes, die von 3.6 inspiriert ist. Es ist nicht notwendig, alle Gitterlinien zu erkennen. Werden die äußeren Gitterkanten des Brettes erkannt, so können dadurch die Eckpunkte ermittelt werden. Dies reicht aus, um jeden anderen Schnittpunkt zu interpolieren und das gesamte Gitter zu rekonstruieren. Dadurch, dass das Spielbrett über einen leeren Rand verfügt, bildet das Gitter eine geschlossene Kontur, auch wenn es mit Steinen bespielt wird. Da die erste Reihe des Brettes nur sehr selten bespielt wird, kann diese Kontur dann benutzt werden, um mit der Hough-Transformation zur Erkennung von Linien auf die äußeren Gitterlinien zurückzuschließen.

Der Nutzer richtet die Kamera üblicherweise sehr zentral auf das Brett und bietet dem Brett den meisten Raum im Bild. Aus allen Konturen, die das Binärbild bietet, wird daher die größte Kontur gewählt, die das Zentrum umschließt. In Abbildung 5.12 ist diese größte zentrale Kontur dieses unbearbeiteten Binärbildes zu sehen. Sie bildet die Position des Go-Brettes bzw. des Gitters bereits gut ab. Sie soll als Basis dienen, um mit der Hough-Transformation die Kanten des Brettes zu erkennen. Zwei Steine auf der ersten Reihe auf der linken Seite des Brettes verbinden jedoch die Gitterkontur mit einer weiteren, in diesem Fall der Kontur der Handschuhe, die nicht zum Brett gehört. Diese bildet eine Gerade, die als Gitterkante missverstanden werden kann. Dies ist vor allem ein Problem, wenn das Go-Brett einen Schatten wirft, wie er zum Beispiel an der unteren Kante des Brettes zu sehen ist.

Um diese Verbindung der Konturen durch Steine zu verhindern, wird das Binarbild zunächst weiter bearbeitet. Mit einem Kernel werden Areale des Binarbildes identifiziert, die dicker als die Gitterlinien sind. Diese Areale werden entfernt. Um kleine Lücken in den Konturen zu schließen, wird anschließend eine Dilatation auf die verbleibenden Pixel angewendet. Das Ergebnis dieser Bearbeitung ist ein für die Gitteridentifikation verbessertes Binärbild, wie es in 5.12 zu sehen ist. Sucht man nun die Kontur des Brettes, so ist diese nicht mehr mit anderen Konturen verbunden. Der Nachteil dieser neuen Kontur ist, dass es Durchbrüche in die inneren Areale des Brettes geben kann. Diese sind jedoch üblicherweise nicht sehr gerade und können daher nicht als eine äußere Kante des Brettes missverstanden werden. Im Folgenden wird mit dieser Kontur weitergearbeitet.

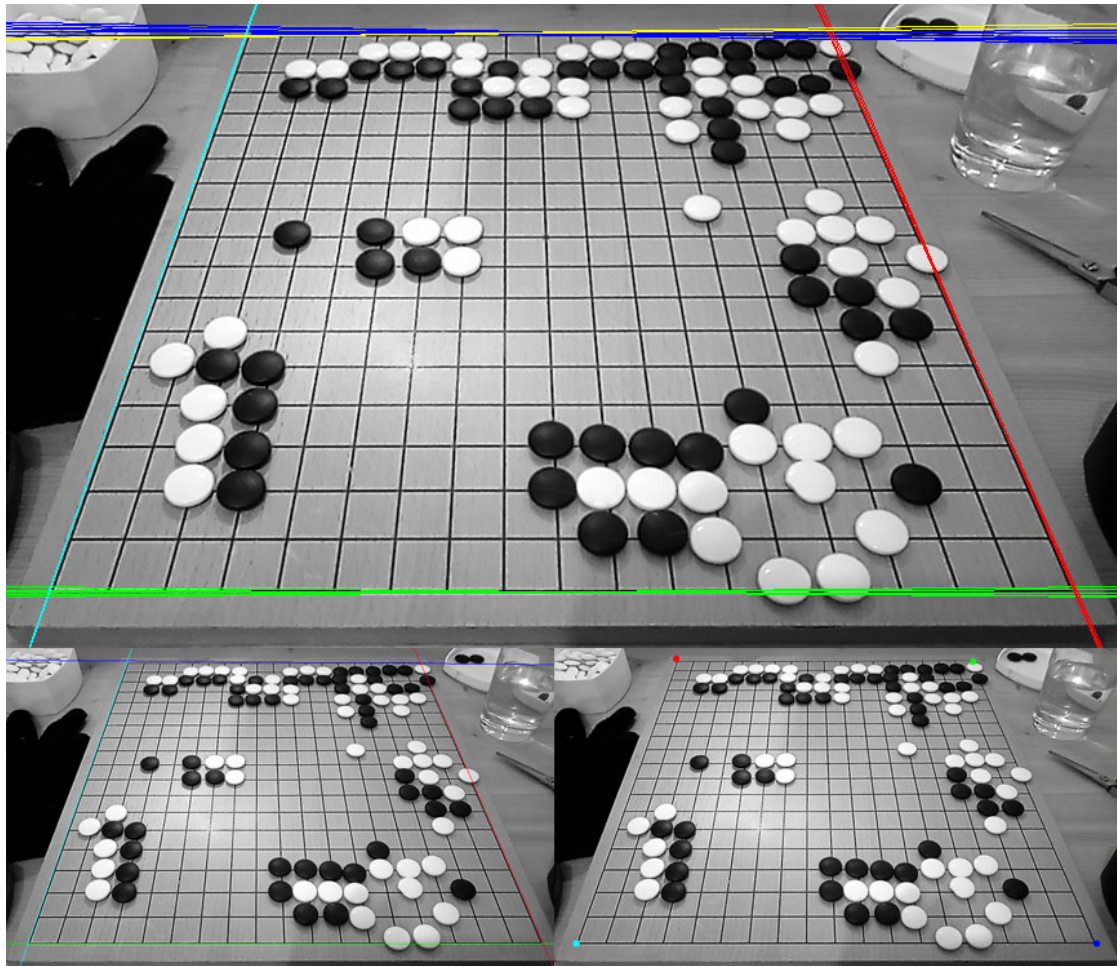


Abbildung 5.13: Kantenerkennung des Spielbrettes:
Oben: Erkannte Kanten-Cluster,
Links Unten: gemittelte Kanten,
Rechts Unten: ermittelte Eckpunkte

Nutzung der Brettkonturen zur Erkennung der Ecken

Die erkannte äußere Kontur der Gitterlinien wird nun genutzt, um die äußeren Kanten des Gitters zu erkennen. Dazu wird die Hough-Transformation für Linien auf die äußere Kontour angewendet. Durch diese Methode werden mehrere Linien pro Außenkante gefunden und teilweise auch Linien, die keiner Kante angehören. Diese Geraden werden nun durch k-Means-Clustering genau vier Clustern zugeordnet. Die K-Means-Clusteranalyse wählt dabei vier zufällige Clusterzentren aus. Jeder Datenpunkt wird dem nach euklidischer Distanz nächstgelegenen Cluster zugeordnet. Die Clusterzentren werden danach aktualisiert, und der Vorgang so lange wiederholt, bis sich die Clusterzentren nicht mehr signifikant ändern. Die Datenpunkte, nach denen die Geraden sortiert werden, bestehen hier aus ihrem Theta und Rho, wobei diese so gewichtet werden, dass sie miteinander vergleichbar sind. Geraden, die keine der Kanten repräsentieren, müssen von den Clustern entfernt werden. Dazu werden die Geraden eines Clusters, die um mehr als die dreifache Standardabweichung vom Durchschnitt abweichen, vom Cluster entfernt. Die Ergebnisse sind in Abbildung 5.13 dargestellt. Jeder Cluster von Linien hat eine Farbe, erkannte Ausreißer werden in Gelb dargestellt.

Der Java-Wrapper von OpenCV bietet keine Möglichkeit, auf die Ergebnisse der Akkumulation der Hough-Transformation zuzugreifen. Beim Aufruf wird lediglich eine untere Grenze angegeben, wie gut die Gerade eine Linie repräsentieren muss. Es kann also nicht einfach bestimmt werden, welche der Geraden in jedem Cluster die Gitterlinie am besten repräsentiert, ohne diese rechenintensive Operation mehrmals durchzuführen. Um eine repräsentative Gerade zu ermitteln, wird daher zur Vereinfachung der Mittelwert aller gefundenen Geraden pro Cluster berechnet. Von den gemittelten Geraden wird ermittelt, welche annähernd parallel und damit gegenüberliegend sind. Dann werden aus den Schnittpunkten der Geraden die Ecken des Gitters ermittelt. Die gemittelten Geraden und die gefundenen Eckpunkte sind in 5.13 dargestellt. In diesem Beispiel kann man erkennen, dass die obere Gerade weniger gut gewählt ist und die Schnittpunkte etwas neben den Ecken des Gitters liegen. Es ist aber für eine weitere Verarbeitung ausreichend genau.

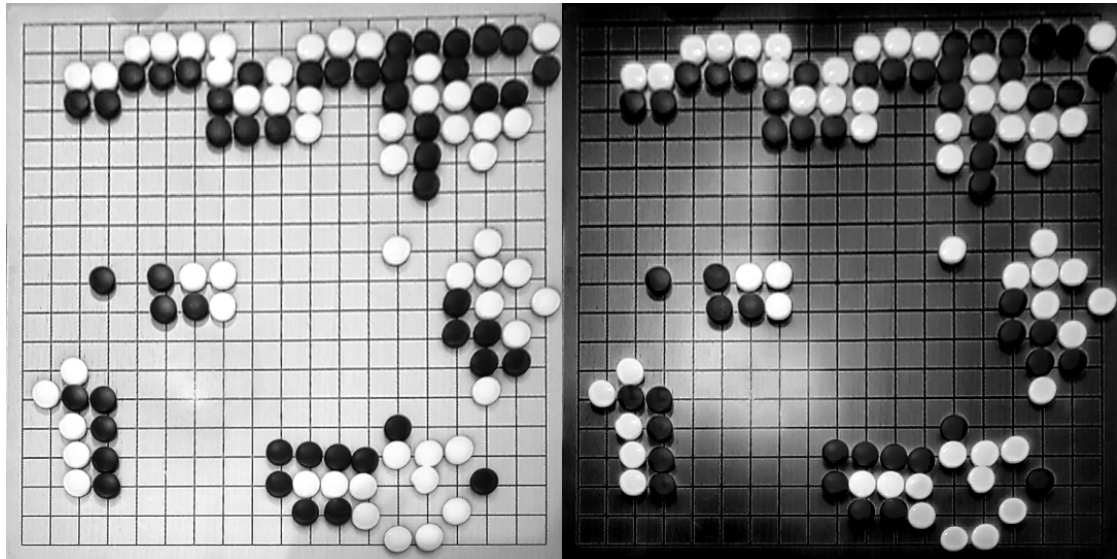


Abbildung 5.14: Die von der Perspektive befreiten transformierten Basisbilder

Entzerrung des Spielbrettes

Es ist bekannt, dass ein Go-Brett annähernd quadratisch ist. Mithilfe der gefundenen Eckpunkte kann deshalb nun die Transformationsmatrix und die inverse Transformationsmatrix ermittelt werden. In Abbildung 5.14 sind die Basisbilder dargestellt, nachdem man die Transformation angewendet hat. Man kann deutlich erkennen, wie die unter Perspektive aufgenommenen Bilder nun entzerrt und in eine Vogelperspektive überführt sind. Die entzerrten Bilder bilden die Basis für die Stellungserkennung, da die Steine nun deutlich einfacher und zuverlässiger identifiziert werden können. Das Gitter befindet sich immer an derselben Position, und jede Brettcoordinate kann einer festen Bildcoordinate zugewiesen werden.

5.4.3 Erkennung der schwarzen Spielsteine

Die Erkennung von schwarzen Steinen ist eine Grundlage für eine erfolgreiche Stellungserkennung. Sie unterscheidet sich von der Erkennung der weißen Steine, da die Voraussetzungen besser sind. Das Basisbild für die Erkennung ist bereits frei von Reflexionen und weist einen guten Kontrast zum Hintergrund auf. Es eignet sich daher sehr gut, um die Positionen zu identifizieren. Eines der Hauptprobleme, das es zu bewältigen gilt, ist der Zusammenschluss von schwarzen Steingruppen zu Farbelustern. Nicht jeder Stein ist durch seine Kontur klar von den anderen getrennt. Des Weiteren werden nicht alle Spielsteine zentral auf die Schnittpunkte gesetzt. Teilweise sind die Schnittpunkte unter dem Blickwinkel sogar noch sichtbar. Bei Go-Steinen handelt es sich um linsenförmige Steine. Ihr Mittelpunkt liegt auf einer anderen Ebene als die Schnittpunkte. Durch eine schräge Sicht schaut man also etwas unter die Go-Steine. Das beeinflusst auch die Steinpositionen nach der Transformation zur Vogelperspektive.

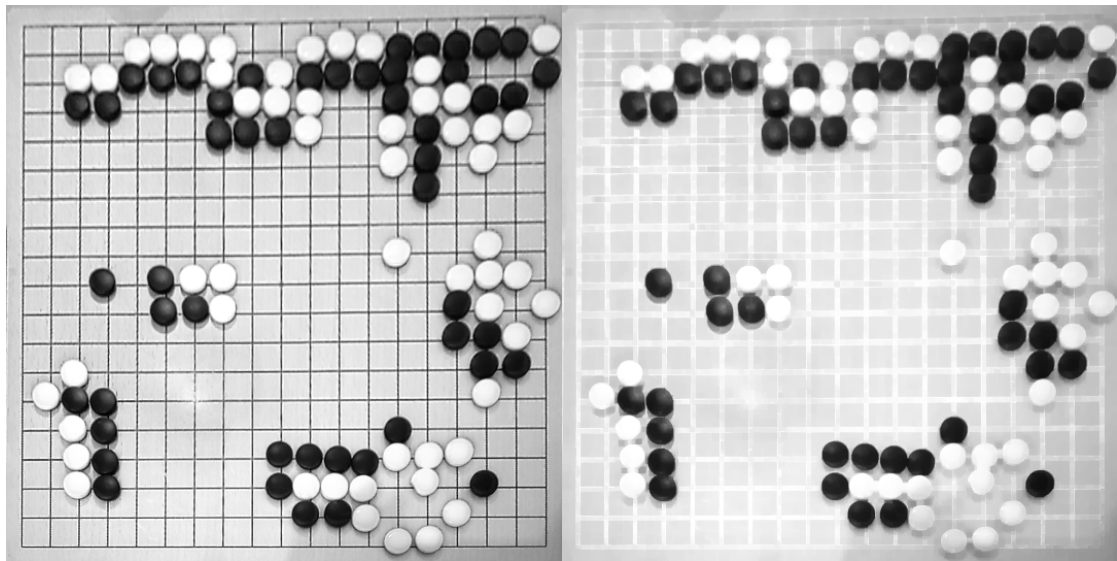


Abbildung 5.15: Das Basisbild zur Erkennung von schwarzen Steinen vor und nach dem Filtern von dünnen Elementen

Weitere Vorverarbeitung

Für die Erkennung der schwarzen Steine sind die Gitterlinien hinderlich. Deshalb werden vor der weiteren Verarbeitung zunächst dünne Elemente herausgefiltert. Zuerst wird das Bild dilatiert. Dadurch verschwinden alle kleinen schwarzen Areale. Um die Größe der Steine wiederherzustellen, wird das Bild danach erodiert. Das Resultat ist in Abbildung 5.15 zu sehen. Ein angenehmer Nebeneffekt ist, dass auch die Schatten der weißen Spielsteine zum Teil kompensiert werden. Es ist allerdings etwas überraschend zu sehen, dass die Gitterlinien nicht nur verschwunden, sondern sogar heller als der Bretthintergrund sind. Dies liegt an der internen Bildvorverarbeitung der mobilen Kamera. Nach Schärfungsoperationen sind hellere Punkte in der Nähe von dunklen Kanten üblich. Diese dominieren nun das lokale Areal.

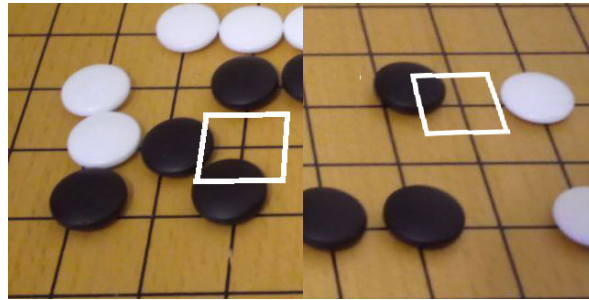


Abbildung 5.16: Ungenaue Platzierungen beeinflussen benachbarte Felder

Umgebungsanalyse jedes Schnittpunktes

Ein trivialer Ansatz ist die Untersuchung der Umgebung jedes Schnittpunktes, wie sie in 3.5 erfolgt. Die Position jedes Schnittpunktes ist auf dem entzerrten Bild fest definiert. Befinden sich in der Umgebung viele dunkle Pixel, handelt es sich um einen dort platzierten schwarzen Stein.

In der Praxis erzeugt diese Idee einige Probleme. Da die Spielsteine oft nicht mittig auf den Schnittpunkten platziert sind, können sie mehreren Schnittpunkten zugewiesen werden, sodass ein einzelner Stein auf unterschiedlichen Schnittpunkten erkannt wird. Betrachtet man beispielsweise drei aneinander liegende Schnittpunkte wie in Abbildung 5.16, so können zwei Steine so platziert sein, dass drei Felder zwei Drittel eines Steines beinhalten. Auch der Schattenwurf von Steinen kann als schwarzer Stein interpretiert werden. Aus diesen Gründen wird diese Methode im Weiteren nicht verwendet.

Erkennung von Kreisen mit der Hough-Transformation für Kreise

Ein anderer Ansatz ist es daher, die Spielsteine zu erkennen und ihnen den nächstgelegenen Schnittpunkt zuzuweisen. Dazu können Hough-Kreise wie in 3.3 benutzt werden. Ähnlich den Hough-Linien werden durch jeden Punkt Kreise verschiedener Größen gelegt und die Anzahl getroffener Punkte akkumuliert. Diese Technik ist in der Praxis sehr unzuverlässig. Die Methode achtet nicht darauf, ob die Kreise gefüllt sind. Daher werden schwarze Steine nicht erkannt, die teilweise durch weiße Steine verdeckt sind, die davor liegen. Cluster von schwarzen Steinen führen zu einer Vielzahl erkannter Kreise. Auch leere Felder zwischen den Gitterlinien werden oft als Kreise erkannt, da sie in etwa dieselbe Größe haben. Zusätzlich dazu ist der Ansatz sehr rechenintensiv. Die Kombination aus all dem macht diese Methode zu einer unzureichenden, die nicht weiter verfolgt wurde.

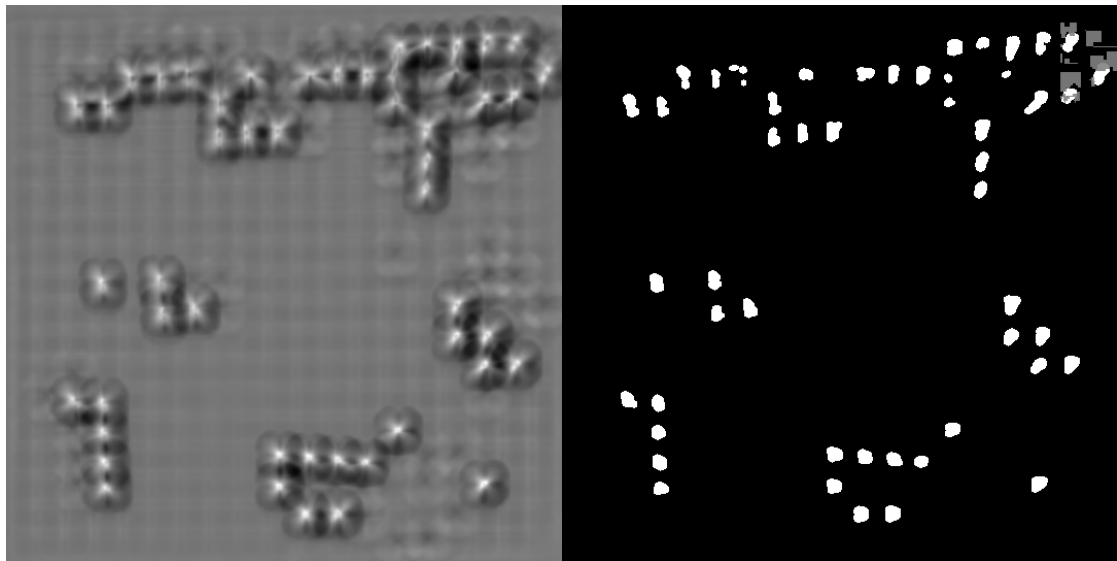


Abbildung 5.17: Die Heatmap der Steine und ihr Binärbild nach Schwellenwertanwendung

Erkennung von Steinen mit einem Kernel

Die gewählte Methode erkennt Steine, indem ein schwarzes rundes Template mit der erwarteten Steingröße auf das Bild angewendet wird. Das Resultat ist die Heatmap, die in Abbildung 5.17 dargestellt ist. Nach Anwendung eines Schwellenwertes sind die Positionen der erkannten Steine separiert. Aufgrund der Reflexionen auf den schwarzen Steinen werden einzelne Steine teilweise in mehrere Cluster segmentiert. Von jedem Cluster wird der Mittelpunkt gebildet. Danach wird die Heatmap geblurt. Für jeden Mittelpunkt wird nun das Maximum in der geblurten Heatmap im Umkreis eines Steinradius gesucht. Dies wird gemacht, um Steinen, die in mehrere Cluster segmentiert sind, wieder einen einzigen Punkt zuzuordnen. Die erkannten Steine sind in 5.18 dargestellt. Ein einzelner Stein, der in 3 Teile segmentiert wurde, wurde hier als 2 Steine erkannt, weil ihre Maxima mehr als 1 Steinradius voneinander entfernt liegen. Dies ist in diesem Fall nicht weiter problematisch, da beide erkannten Steine demselben Schnittpunkt zugeordnet werden.

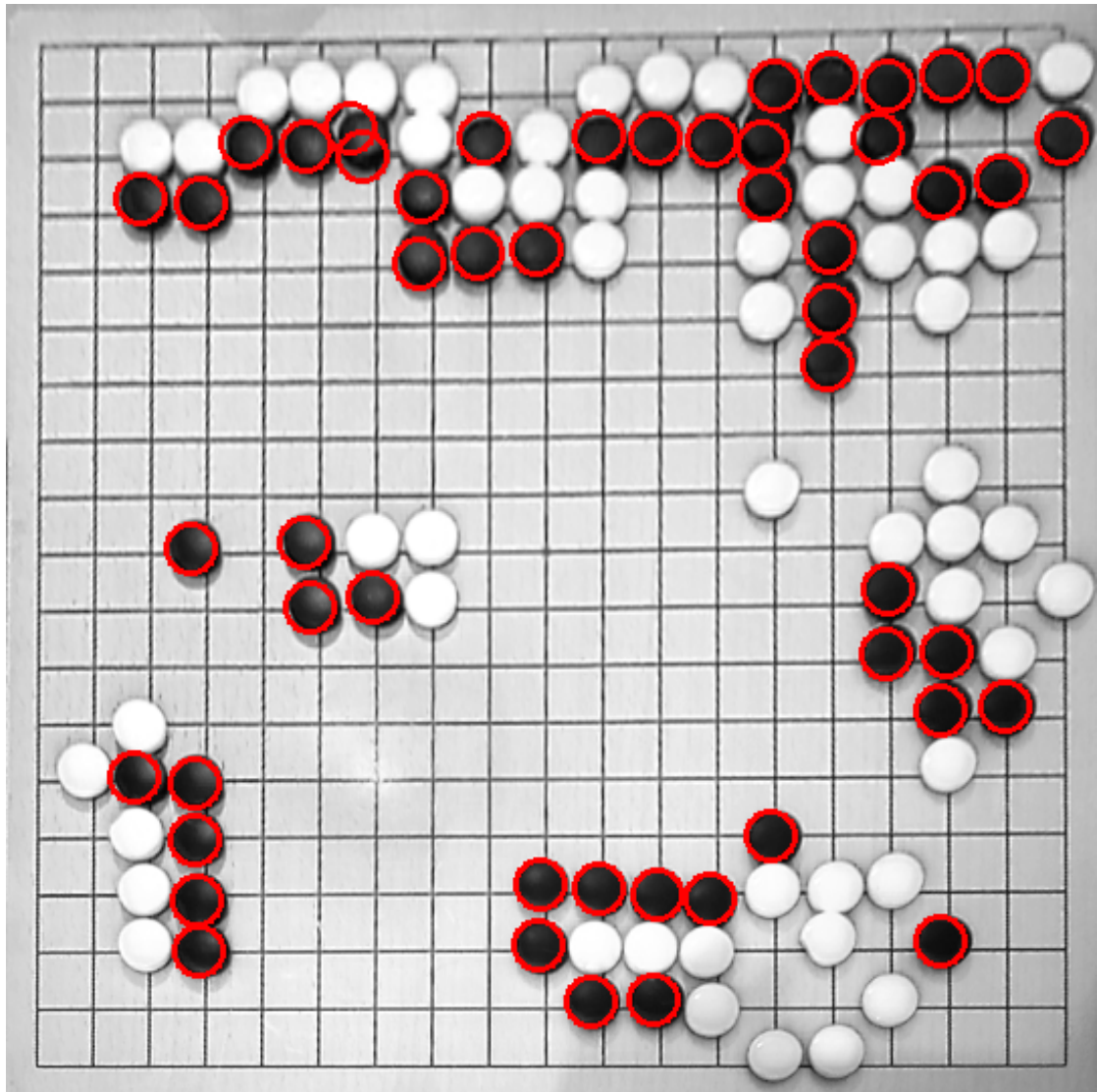


Abbildung 5.18: Das Ergebnis der Erkennung der schwarzen Steine

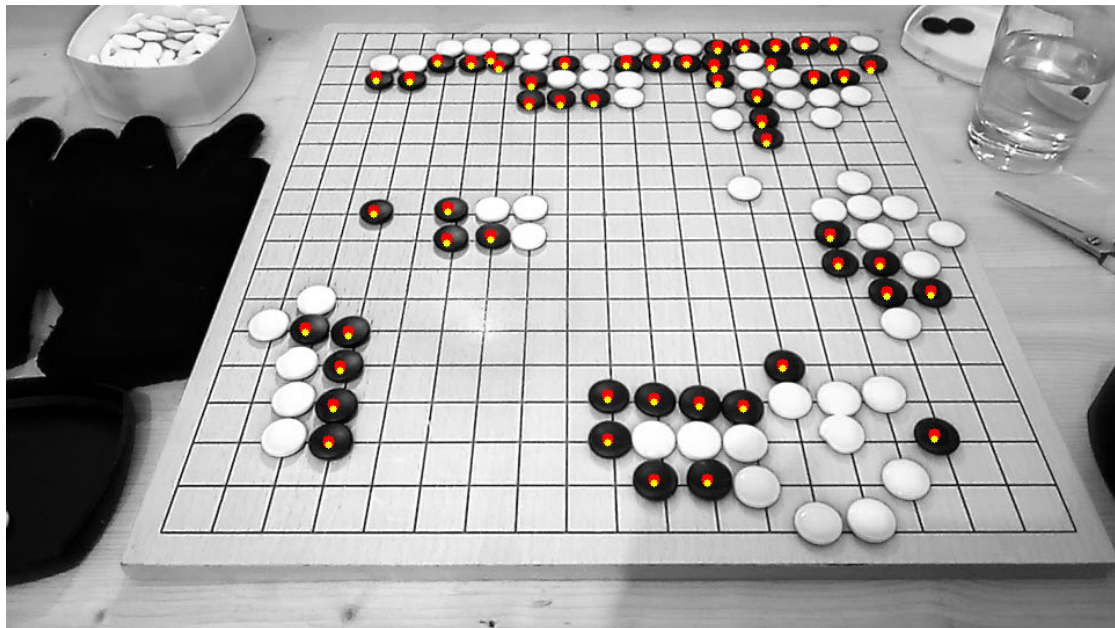


Abbildung 5.19: Die Mittelpunkte der Steine (Rot) und die unter der Perspektive angepassten Positionen (Gelb)

Anpassung der Mittelpunkte an das Gitter

Transformiert man die ermittelten Mittelpunkte wie in Abbildung 5.19 in das Originalbild zurück, wird deutlich, dass sie nicht auf den Gitterlinien liegen, auch wenn ein Stein mittig platziert wurde. Dies liegt daran, dass die Mittelpunkte der Spielsteine auf einer anderen Ebene liegen als die Schnittpunkte des Go-Brettes. Je flacher der Blickwinkel auf das Brett ist, desto stärker ist dieser Effekt. Um diesen Fehler zu kompensieren, wird der Blickwinkel zu einer Korrektur hinzugezogen. Der Blickwinkel wird aus den detektierten Ecken des Brettes approximiert, indem die Breite des Brettes im Verhältnis zur Höhe betrachtet wird. Je flacher der geschätzte Blickwinkel ist, desto mehr werden - unter Berücksichtigung trigonometrischer Funktionen - die Steinmittenpunkte in Y-Richtung nach unten verschoben. Dies resultiert in einer starken Verschiebung unter flachen Blickwinkeln und keinerlei Verschiebung unter einer steilen Ansicht von oben. Die angepassten Mittelpunkte kongruieren nun besser mit den darunterliegenden Gitterlinien. Zur weiteren Verarbeitung werden die angepassten Punkte nun wieder in das entzerrte Bild transformiert. Bei der Zusammenstellung der Go-Stellung wird später jedem dieser Punkte der dichteste Schnittpunkt zugeordnet werden.

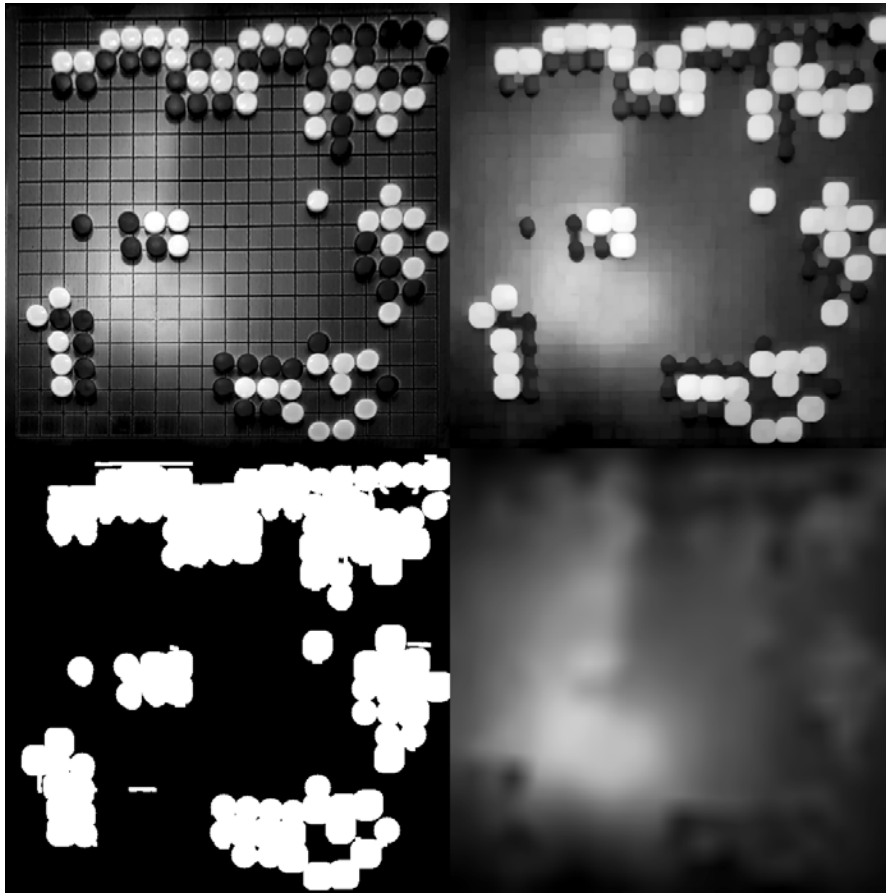


Abbildung 5.20: Arbeitsschritte zur Separierung des Hintergrundes:
Links oben: Ursprüngliches transformiertes Basisbild,
Rechts oben: Basisbild ohne Gitterlinien,
Links unten: Binärmaske der zu füllenden Areale,
Rechts unten: Der separierte Hintergrund

5.4.4 Erkennung der weißen Steine

Die Erkennung der weißen Steine stellt sich aufgrund der schlechteren Bildbasis als Herausforderung dar. Im Basisbild für die Erkennung schwarzer Steine sind Reflexionen bereits größtenteils kompensiert. Für die weißen Steine ist das nicht der Fall. Hier muss den Reflexionen mehr Beachtung geschenkt werden. Der Kontrast zwischen den weißen Steinen und von Lichtquellen aufgehellten Bereichen des Brettes ist teilweise sehr gering. Eine weitere Vorverarbeitung des Bildes wird daher benötigt. Diese wird erst nach der Erkennung der schwarzen Steine ausgeführt, da ihre Position hierfür benötigt wird.

Die Hauptidee ist es, die Steine und Gitterlinien vom Hintergrund, also dem Spielbrett, zu separieren. Dazu wird eine Maske erstellt, um diese Areale auf dem Brett zu markieren und anschließend mit der näheren Umgebung zu ersetzen. Danach kann der Helligkeitsverlauf des Hintergrunds vom Basisbild abgezogen werden, um eine gleichmäßige Beleuchtung über das ganze Bild zu erhalten. Von hier an kann dieselbe Erkennungsstrategie wie bei den schwarzen Steinen angewendet werden.

Die Verarbeitungsschritte zur Separierung des Hintergrundes sind in Abbildung 5.20 dargestellt und werden im Folgenden erläutert.

Entfernen der Gitterlinien

Um den Hintergrund des Brettes zu separieren, müssen die dunklen Gitterlinien entfernt werden. Das Bild wird dazu zuerst einer leichten Dilatation unterzogen. Die Gitterlinien werden also zunächst verdickt. Dies entfernt die hellen Pixel an den Gitterlinienrändern, die durch die kamerainterne Vorverarbeitung des Bildes entstanden sind. Durch diesen zusätzlichen Schritt wird verhindert, dass die Gitterlinien heller als die Umgebung werden, so wie es im vorherigen Unterkapitel erkennbar war. Nun wird eine erheblichere Erosion durchgeführt, welche die Gitterlinien verschwinden lässt. Auf eine abschließende Dilatation zur Wiederherstellung der vorherigen Steingröße wird diesmal verzichtet, da das nächste Ziel ist, eine Maske zu erstellen, um die Steine herauszufiltern. Es ist daher hilfreich, auch die nähere Umgebung der Steine herauszufiltern, um auch die Schatten der Steine zu entfernen.

Erstellung der Maske zum Füllen

Bei der Erstellung der Binärmaske geht es darum, die Areale, die schwarze und weiße Steine beherbergen, grob zu identifizieren, um diese dann mit der Umgebung zu füllen. Dazu wird eine Canny Edge Detection verwendet, um die Konturen der Steincluster zu erkennen. Diese Konturen werden verdickt und an den Bildrändern geschlossen. Die Konturen der Steincluster werden dann gefüllt. Für den Fall, dass aufgrund eines zu hoch gewählten Thresholds das ganze Brett als Cluster erkannt wird, wird der Threshold so lange verringert, bis die zu füllenden Flächen weniger als 50% des Bildes beinhalten. Anschließend werden noch kreisförmige Areale um die Positionen der als schwarz erkannten Steine ergänzend maskiert.



Abbildung 5.21: Reflexionsminderung durch separierten Hintergrund:
Links: Das ursprüngliche transformierte Basisbild,
Mitte: Der separierte Hintergrund,
Rechts: Das Basisbild nach Abzug des Hintergrundes

Füllen der maskierten Gebiete

Die Binärmaske wird nun genutzt, um diese Areale mit der näheren Umgebung zu füllen. Dies geschieht mit der OpenCV-Methode `Photo.inpaint`. Das Ergebnis ist ein separierter Bretthintergrund, der nur leicht von den Steinen und ihren Schatten beeinflusst wird.

Entfernen der Reflexionen aus dem Basisbild

Der separierte Hintergrund wird nun genutzt, um die Reflexionen im Basisbild zu beseitigen. Dazu wird er einfach vom Basisbild abgezogen. Das Ergebnis ist in Abbildung 5.21 zu sehen.

Extraktion der Steinpositionen

Von hier an ist das Vorgehen weitgehend analog zur Verarbeitung der schwarzen Steine in Kapitel 5.4.3 (ab “Erkennung von Steinen mit einem Kernel”). Es wird also mithilfe eines Kernels eine Heatmap erstellt. Auf diese wird ein Threshold angewendet, um die Steine zu separieren. Dann werden auch hier die Mittelpunkte der Perspektive angepasst. Die erkannten Positionen sind in Abbildung 5.22 dargestellt.

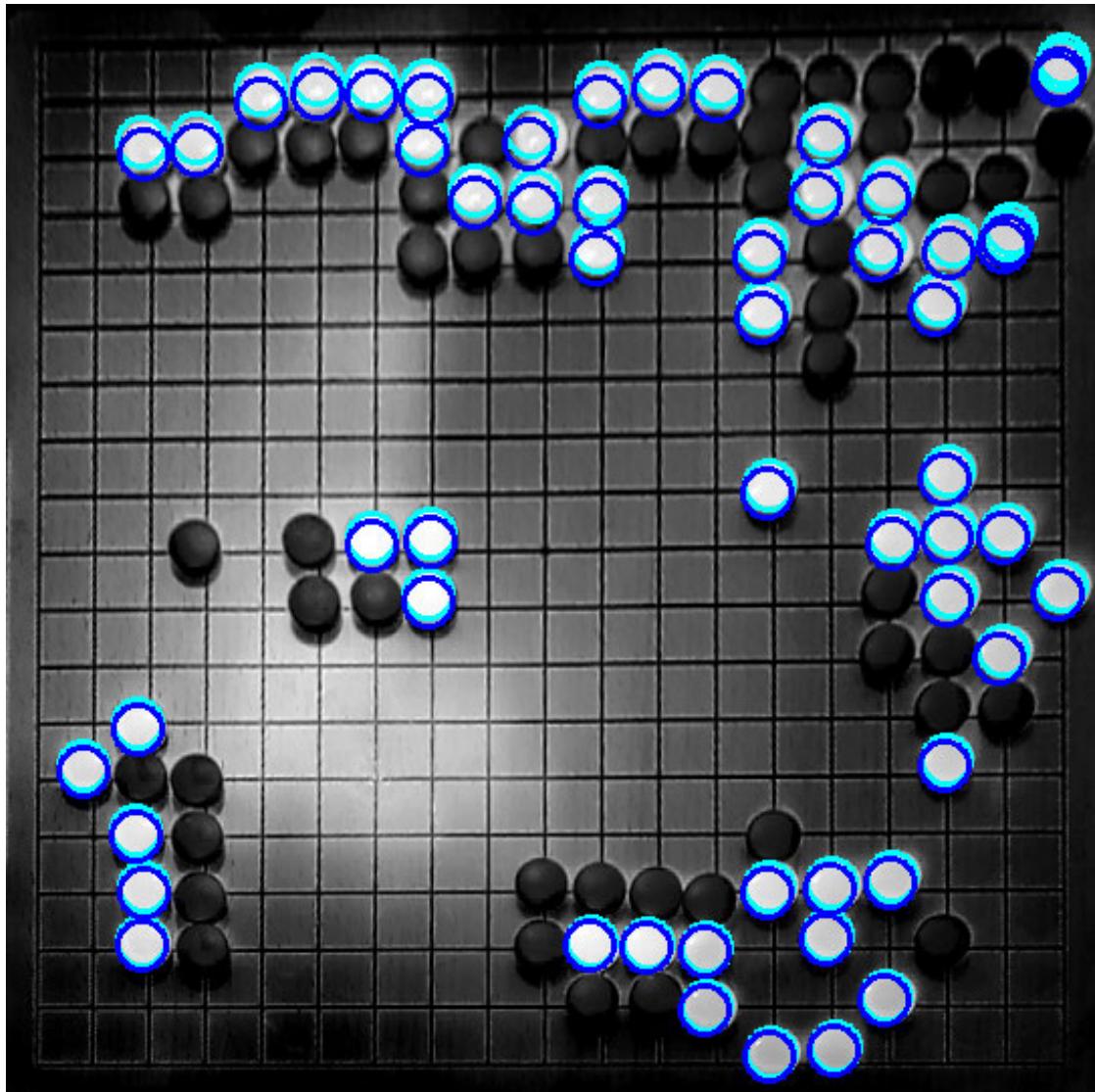


Abbildung 5.22: Die erkannten Positionen vor (Türkis) und nach der Perspektivanpassung (Dunkelblau)

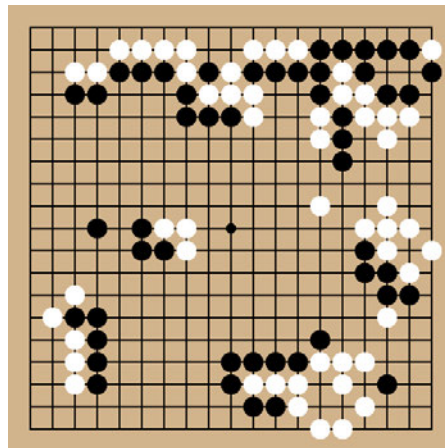


Abbildung 5.23: Eine Visualisierung der erkannten abstrahierten Go-Stellung

5.4.5 Abstraktion der Go-Stellung

Zur anschließenden Spielanalyse muss für die Go-Stellung eine abstrakte Darstellungsform gewählt werden. In einer Klasse wird in einem 2D 19x19 Array für jeden Schnittpunkt des Spielbrettes der Status gespeichert. Dieser kann „EMPTY“, „BLACK“ oder „WHITE“ sein, je nachdem ob er leer ist, oder von einem schwarzen oder weißen Stein besetzt wird. Außerdem wird in einem Feld gespeichert, welcher Spieler am Zug ist, also „BLACK“ oder „WHITE“.

In der unverzerrten Ansicht (der Vogelperspektive) sind die Positionen der Schnittpunkte durch ein regelmäßiges Grid statisch festgelegt, da die Eckpunkte des Brettes auf feste Positionen transformiert werden. Um nun zu bestimmen, welcher Schnittpunkt von einem Stein belegt ist, werden die zuvor erkannten weißen und schwarzen Steinpositionen genutzt. Jeder erkannten Steinposition wird der nächstgelegene Schnittpunkt zugeordnet und damit auch eine Brettcoordinate. Diese wird dann entsprechend im 19x19 Array der Klasse festgehalten.

Der in der Stellung zu ziehende Spieler wird dem Status der Benutzeroberfläche entnommen.

Eine Visualisierung der abstrahierten Go-Stellung ist in Abbildung 5.23 dargestellt. Sie bildet die Grundlage zur nun folgenden Analyse der Go-Stellung und ist somit komplett unabhängig vom Kamerabild.

5.5 Implementierung der Spiel-Analyse

Dieses Unterkapitel beschäftigt sich mit der Analyse der aus den Kamerabildern abstrahierten Go-Stellungen. Hierzu wird eine externe API genutzt. Die API verwendet KataGo aus den in 4.8 genannten Gründen. Sie wird genutzt, um den besten Zug zu finden und den momentanen Spielstand zu approximieren.

Architektur der Analyse

Wie in Kapitel 5.2 beschrieben, verläuft die Analyse asynchron und ist in Abbildung 5.24 dargestellt. Nachdem die Go-Stellung im ImageProcessor vom Main Thread erkannt wurde, wird sie zur Analyse an einen thread-sicheren Analyse Manager übergeben. Direkt danach erkundigt sich der Main Thread beim Analyse Manager darüber, ob bereits ein Analyseergebnis vorliegt. Dies ist dann der Fall, wenn die Stellung bereits in einem vorherigen Durchlauf erkannt und in der Zwischenzeit verarbeitet wurde. Liegt ein Analyseergebnis vor, wird der beste Zug und der momentane Spielstand extrahiert, und es wird mit der Darstellung fortgefahren. Vier weitere Threads kümmern sich im GoBoardAnalyzer nun um die eigentliche Analyse der Go-Stellungen. Sie pollen vom Analyse Manager neue zu analysierende Stellungen. Liegt keine Stellung vor, wird kurz gewartet, bevor erneut nachgefragt wird. Falls neue Stellungen vorliegen, werden diese entgegengenommen und an die externe API gesendet. Es wird blockierend auf das Analyseergebnis gewartet, welches anschließend wieder an den Manager weitergeleitet wird.

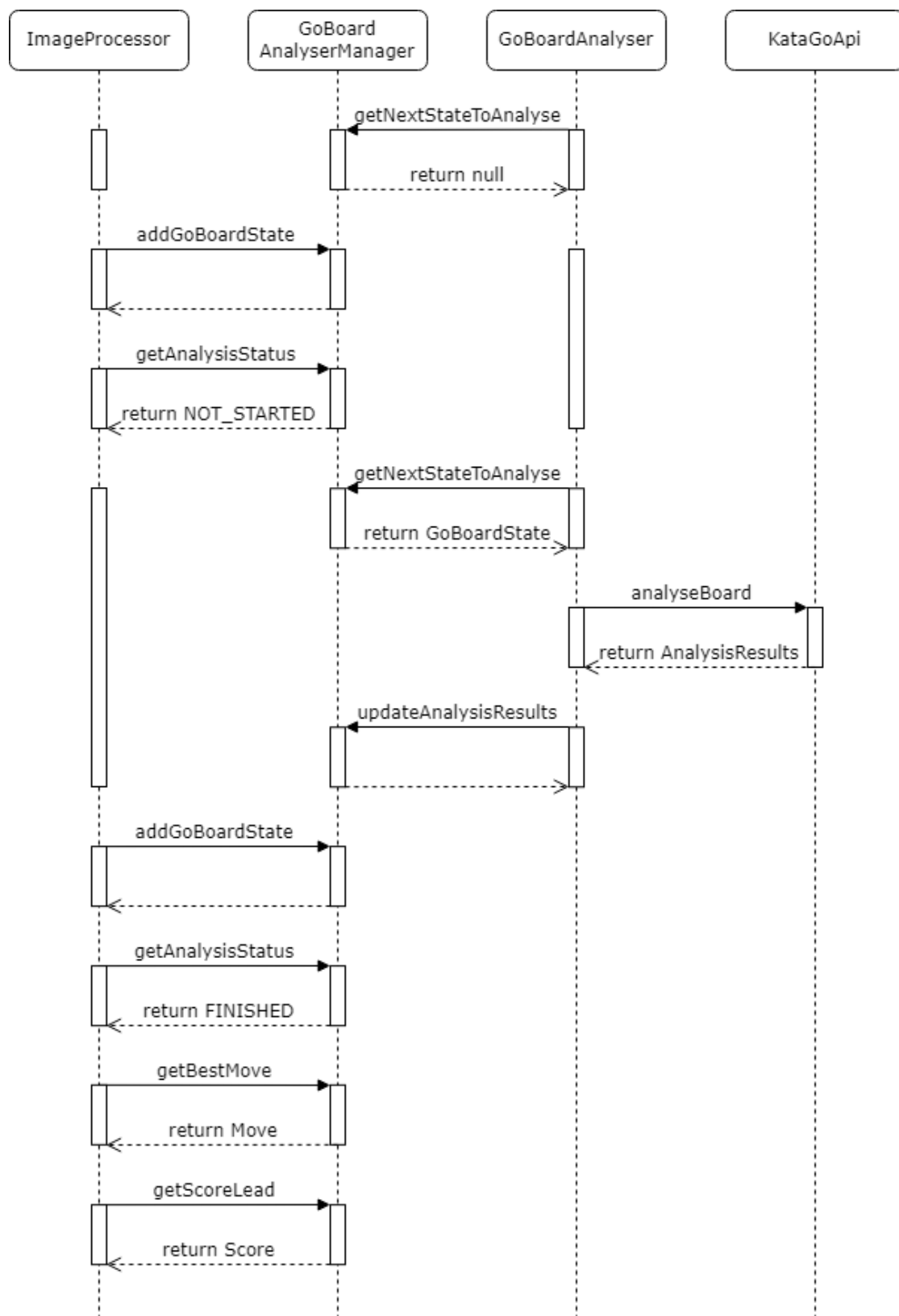


Abbildung 5.24: Eine vereinfachte Darstellung des Datenverlaufs zur Spiel-Analyse 61

Der Analyse Manager

Der Manager verwaltet die Go-Stellungen und ihre Analyseergebnisse thread-sicher in einer HashMap. Hierbei sind die Go-Stellungen die Keys und die Analyseergebnisse die Values. Analyseergebnisse verfügen über einen Status, einen besten Zug und einen Spielstand. Wird dem Manager eine neue Stellung übergeben, wird ermittelt, ob dieselbe Go-Stellung bereits in der HashMap vorhanden ist. Ist dies nicht der Fall, wird sie hinzugefügt und der Status auf `NOT_STARTED` gesetzt. Fragt nun ein Analyser nach einer Go-Stellung, so erhält er die erste Stellung, deren Analyse noch nicht gestartet ist oder fehlerhaft war. Der Status der übergebenen Stellung wird dann auf `IN_PROGRESS` gesetzt. Meldet ein Analyser ein Ergebnis zu einer Stellung zurück, so wird der Status auf `FINISHED` gesetzt und der ermittelte beste Zug sowie der Spielstand werden im Analyseergebnis festgehalten. Beim Auftreten eines Fehlers kann der Status durch eine Analyserückmeldung auch auf `ERROR` gesetzt werden. Ist eine Analyse zu einer Stellung abgeschlossen, so kann für eine Stellung der beste Zug und der Spielstand beim Manager abgefragt werden.

Die externe Analyse API

Die hier verwendete Analyse API ist eine REST API. Das ist eine Programmierschnittstelle, die auf den Prinzipien von “Representational State Transfer” basiert, um über das Internet eine standardisierte Kommunikation zu ermöglichen. Für weiterführende Informationen wird auf das Buch “Architectural Styles and the Design of Network-based Software Architectures” von Roy Fielding verwiesen (Fielding, 2000). Die API ermöglicht die Auswertung des aktuellen Spielzustands und die Vorhersage des besten nächsten Zuges. Die Felder der Anfragen und Antworten sind wie folgt aufgebaut.

API-Anfrage (jsonRequest):

- *model*: Dieser Wert bleibt konstant und signalisiert, dass kein menschenähnliches Spielmodell verwendet wird.
- *alpha*: Ein Wert von 0 bedeutet, dass KataGo mit 1 Payout für die Analyse verwendet wird.
- *temperature*: Ein Wert von 0 sorgt dafür, dass immer der beste Zug deterministisch zurückgegeben wird, ohne zufällige Variationen.
- *black* und *white*: Diese sind Listen von Koordinaten, die die Positionen der schwarzen und weißen Steine auf dem Brett aufzählen.
- *moves*: Dieses Feld bleibt immer leer, da die Zugreihenfolge nicht bekannt und relevant ist.
- *start_player*: Zeigt an, welcher Spieler am Zug ist.

API-Antwort (response):

- *move*: Dies zeigt den besten nächsten Zug an, dargestellt als Koordinaten auf dem Brett.
- *score*: Der aktuelle Punktestand wird hiermit zurückgegeben.
- *ownership*: Diese Angabe ist eine Schätzung darüber, welche Teile des Brettes welchem Spieler gehören und somit für die Anwendung nicht relevant.

Die Analyser

Die Analyser folgen dem in 5.5 beschriebenen Zyklus. Eine zu analysierende Go-Stellung wird zunächst für die externe API in ein entsprechendes JSON-Format konvertiert. Mithilfe der HTTP-Client-Bibliothek `okhttp3` wird ein HTTP POST-Request aus der URL und dem JSON-Objekt erstellt. Mit einem `OkHttpClient` wird die Nachricht versendet und auf eine Antwort gewartet. Aus der JSON-Antwort wird der beste Zug und der Score extrahiert und weitergeleitet.

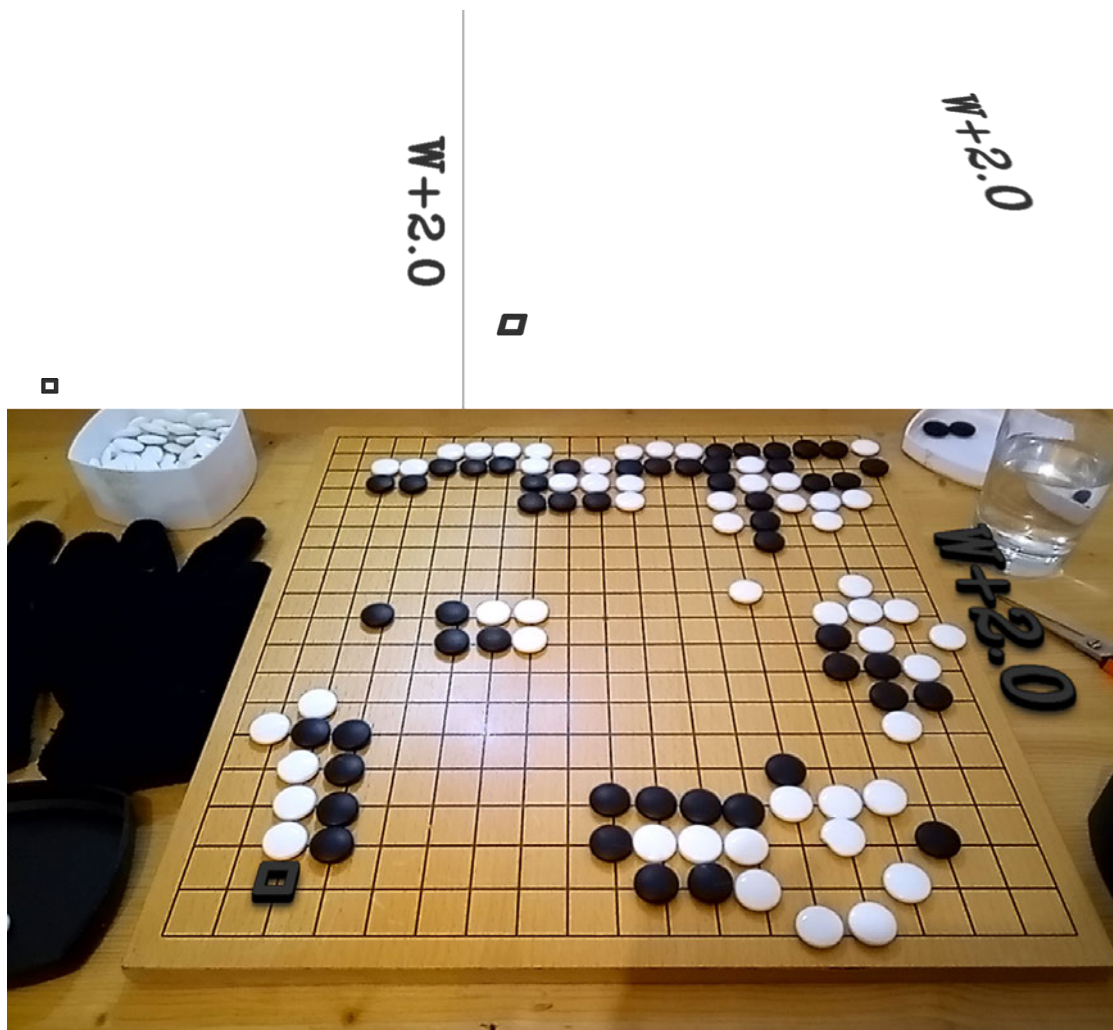


Abbildung 5.25: Die Schritte zur augmentierten Darstellung der Ergebnisse

5.6 Integration der augmentierten Darstellung

Nachdem das Analyseergebnis feststeht, wird es in das Original-Kamerabild integriert. Der optimale Spielzug wird mittels eines abgerundeten, viereckigen, perspektivisch angepassten Rings hervorgehoben. Der aktuelle Spielstand erscheint als dreidimensionaler Text am Rand des Brettes. Die Schritte zur Erzielung dieses Effekts sind in Abbildung 5.25 illustriert. Zuerst wird der Ring und der Text in 2D-Form in der unverzerrten Vogelperspektive kreiert. Dann erfolgt die Anpassung an die Originalperspektive mittels der ermittelten Transformationsmatrix. Schließlich wird durch das Stapeln mehrerer Schichten in vertikaler Richtung eine Volumenillusion für Ring und Text geschaffen.

Darstellung in Vogelperspektive

Die Ergebnisse werden zuerst in die Vogelperspektive integriert, auf die auch das Spielbrett zur Entzerrung transformiert wurde. Die Position des Spielbrettes ist hier statisch und bekannt. Der Text mit dem Spielstand kann deshalb hier an einer festen Position seitlich der Position des Spielbretts eingebracht werden. Hierzu wird der Spielstand in einen String konvertiert und der Text zentriert positioniert. Die Brettkoordinaten des ermittelten besten Zuges werden zurück in Bildkoordinaten konvertiert. An dieser Stelle wird ein viereckiger Marker fester Größe platziert.

Darstellung in Originalperspektive

Die Transformationsmatrix, die zur Entzerrung des Bretts diente, wird nun invertiert. Mithilfe der invertierten Matrix wird ein Warp zur Originalperspektive umgesetzt. Das Ergebnis wird im Folgenden als Informationslayer bezeichnet.

Layering für 3D-Illusion

Das originale Kamerabild wird mit dem Informationslayer mehrfach überlagert. Nach jeder Schicht wird der Layer um einen Pixel nach oben verschoben. Der zuvor bestimmte Blickwinkel entscheidet über die Anzahl der Überlagerungen. Ein flacherer Winkel erfordert mehr Schichten. Mindestens drei Schichten werden jedoch immer hinzugefügt, um den 3D-Effekt zu gewährleisten.

Techniken zur Verstärkung der Illusion

Zur realistischeren Darstellung werden zusätzliche Techniken eingesetzt. Zwei Schattenlayer bilden die Basis: ein schwach und ein stark geblurter Layer. Der schwach geblurte Layer kreiert die Illusion eines harten, direkten Schattens, während der stark geblurte Layer eine Umgebungsverdunkelung simuliert. Der oberste Layer wird heller dargestellt, um die natürliche Beleuchtung nachzuahmen und dem Objekt mehr Struktur zu verleihen.

5.7 Qualitätssicherung und Tests

Im Rahmen dieser Arbeit wurde besonderer Wert auf eine praxisnahe Qualitätssicherung gelegt. Aufgrund der Schwerpunktlegung auf Exploration verschiedener Techniken und Ansätze lag der Fokus auf manuellen Tests und einer detaillierten Analyse der Ergebnisse. Ein Kern der Qualitätssicherung war daher die Entwicklung eines spezialisierten Debug-View-Systems. Dieses Tool ermöglicht es, jeden Schritt der Bildverarbeitungspipeline in Echtzeit zu visualisieren und zu überprüfen. Diese Form der Visualisierung war entscheidend, um verschiedene Ansätze schnell und effektiv zu bewerten.

Ergänzend zu den manuellen Tests und dem Debug-View-System wurden automatisierte Integrationstests implementiert, um die Zuverlässigkeit und Genauigkeit des Systems weiter zu gewährleisten. Diese Tests umfassen die automatische Analyse von fünf Bildern, die speziell ausgewählt wurden, um unterschiedliche Beleuchtungsbedingungen und Partiefortschritte darzustellen. Für jedes Bild wird die Präzision der Go-Stellungserkennung bewertet. Die Ergebnisse dieser Tests werden in einer CSV-Datei gespeichert, die eine detaillierte Übersicht bietet. Mit diesen Daten konnten Veränderungen und Verbesserungen im Laufe der Entwicklung schnell beurteilt werden.

Die Kombination aus manuellen Tests, dem Debug-View-System und den automatisierten Integrationstests bildete somit eine umfassende Strategie zur Qualitätssicherung, die es ermöglichte, das System unter verschiedenen Bedingungen zu testen und stetig zu verbessern.

6 Evaluation und Diskussion der Ergebnisse

Dieses Kapitel befasst sich mit der Auswertung und Diskussion der Ergebnisse des Projekts. Das Hauptziel ist die Bereitstellung einer benutzerfreundlichen All-in-One-Lösung zur Analyse physischer Go-Partien in Echtzeit, ohne die Notwendigkeit einer vorherigen Digitalisierung oder fest installierten Kameras. Die Leistung des Systems wurde in verschiedenen Punkten bewertet, um Effektivität und Verbesserungen zu ermitteln. Herausforderungen und mögliche Lösungsansätze werden ebenfalls erörtert. Abschließend wird die Bedeutung dieser Erkenntnisse für die zukünftige Entwicklung des Projekts betrachtet.

6.1 Verwendete Hardware

Für die Evaluation kam das CUBOT P60 Smartphone als Testgerät zum Einsatz. Dieses Modell, mit einem Neupreis von rund 115€ (Stand: Dezember 2023), gehört zur preisgünstigsten und leistungsärmsten Kategorie. Es betrieb die Android-Version 12 bzw. API-Level 31. Die technischen Merkmale des CUBOT P60 spiegeln die Mindestanforderungen der anvisierten Zielgruppe wider. Die Bildauflösung der Kamera beträgt 960x540 Pixel. Diese Konfiguration untersucht die Anwendbarkeit der Lösung auch auf Geräten mit grundlegenden technischen Spezifikationen.



Abbildung 6.1: Eine repräsentative Auswahl des verwendeten Testsets

6.2 Verwendetes Testset

Für die Bewertung der Computer-Vision-Technologie wurde ein Testset aus 50 Video-Frames zusammengestellt, die in verschiedenen Umgebungen, unter verschiedenen Winkeln, mit unterschiedlichen Go-Spielsets und unter diversen Lichtbedingungen aufgenommen wurden. Eine repräsentative Auswahl dieses Testsets ist in Abbildung 6.1 dargestellt. Das Testset ist anspruchsvoll und deckt alles von realistischen bis hin zu herausfordernden Situationen ab. Die Aufnahmen entstanden in privaten Wohnungen und Bars, umgeben von Alltagsgegenständen wie Tellern, Gläsern und technischen Geräten, um alltägliche Szenarien widerzuspiegeln.

6.3 Evaluation der Spielbretterkennung

6.3.1 Bewertungskriterien (Spielbretterkennung)

Folgende Bewertungskriterien wurden den Anforderungen (4.3) entnommen:

- Hohe Genauigkeit der Brett-Erkennung
- Erkennung des Spielbretts bei variierenden Kamerawinkeln
 - Erkennung bei direktem Sitzwinkel
 - Erkennung bei diagonalem Blickwinkel
- Toleranz gegenüber unterschiedlichen Hintergründen und Tischfarben
- Toleranz gegenüber niedrigen Bildauflösungen.
- Ignorieren von nicht relevanten Objekten im Kamerabild
- Kompensation verschiedener Lichtverhältnisse

6.3.2 Methodik und Ergebnisse (Spielbretterkennung)

Die Genauigkeit der Bretterkennung wurde mit dem zuvor genannten Testset geprüft, das speziell entwickelt wurde, um alle relevanten Bewertungskriterien abzudecken. Eine Ecke wird als korrekt erkannt angesehen, wenn der ermittelte Punkt weniger als ein Viertel des Abstands zwischen den Gitterlinien von der tatsächlichen Position abweicht. Ein Spielbrett wird als erfolgreich erkannt betrachtet, wenn alle vier Ecken korrekt identifiziert wurden.

Bis auf eine Ausnahme wurden alle Spielbretter korrekt identifiziert, was einer Erfolgsquote von 98% entspricht.



Abbildung 6.2: Ein gefülltes Brett separiert das Gitter in Segmente

6.3.3 Probleme (Spielbretterkennung)

Problem bei nicht geschlossenen Konturen

Das Problem im nicht erkannten Frame ist in Abbildung 6.2 dargestellt. Es besteht darin, dass es keine große geschlossene Kontur mehr gibt, die als Spielbrettgitter interpretiert werden kann.

Das liegt daran, dass alle großflächigeren Gebiete herausgefiltert wurden, sodass nur die dünnen Gitterlinien übrig bleiben. Dies verhindert zwar, dass die Steine die Gitterlinien mit den Bretträndern verbinden und so Brettschatten als Gitterlinien erkannt werden können, jedoch kann es auch verhindern, dass die Gitterlinien ein einziges verbundenes Netz bilden. Dieser Fehler wird insbesondere bei dicht besetzten Brettern sichtbar und hat somit größere Auswirkungen, als im aktuellen Testset ersichtlich ist, da dieses nicht viele voll besetzte Bretter beinhaltet.

Als Lösungsansätze bieten sich folgende Ideen an. Zum einen können mehrere kleine Gitterkonturen zu einer großen kombiniert werden. Hier kann es jedoch problematisch werden zu unterscheiden, welche Konturen zu Gittern und welche zu Schatten gehören. Ein weiterer Ansatz könnte darin bestehen, den Hotspot in der Mitte des Bildes als Spielbrett zu definieren. Konturen, die an diesen zentralen Bereich grenzen, könnten so verbunden werden.

Problem der ungenauen Brettkantenermittlung

In Kapitel 5.4.2 (Nutzung der Brettkonturen zur Erkennung der Ecken) wurde besprochen, wie die Brettkonturen benutzt werden, um die Kanten des Spielbrettes zu erkennen. Obwohl die Kanten der Konturen gut sind, kann es vorkommen, dass die gemittelten Geraden diese nicht optimal approximieren.

Der Grund hierfür liegt, wie beschrieben, in der Limitierung des Java-Wrappers für OpenCV. Die gefundenen Geraden beinhalten keine Information darüber, wie hochwertig ihre Approximierung ist. Deshalb werden alle gefundenen Geraden gleich gewichtet.

Die Lösung hierfür ist es, stattdessen das OpenCV Native Interface zu nutzen, welches diese Informationen zur Verfügung stellt. Jedoch ist dies mit einer deutlich erhöhten Komplexität und Entwicklungsaufwand verbunden.

6.3.4 Analyse und Interpretation der Ergebnisse (Spielbretterkennung)

Die Evaluation zeigt eine erfolgreiche Spielbretterkennung mit hoher Genauigkeit. Die App bewältigt auch bei den niedrigen Bildauflösungen des Testset verschiedene Kamerawinkel, Hintergründe, irrelevante Objekte und Lichtverhältnisse effektiv. Herausforderungen bestehen noch bei der Erkennung nicht geschlossener Konturen und ungenauer Brettanten. Für diese Probleme gibt es allerdings gute Lösungsansätze, wodurch sie leicht behoben werden können.

6.3.5 Weitere Verbesserungsmöglichkeiten (Spielbretterkennung)

Nutzung mehrerer Frames

Die Verwendung mehrerer Frames könnte die Spielbretterkennung verbessern, indem sie die Bildqualität erhöht oder es ermöglicht, Merkmale über die Zeit zu verbinden. Eine detaillierte Implementierung dieser Methode liegt jedoch außerhalb des Rahmens dieser Arbeit, stellt aber ein vielversprechendes Thema für die Zukunft dar.

Validierung der Brett-Position

Die Gitterlinien werden nach der Transformation in die Vogelperspektive stets an fest definierten Stellen erwartet. Durch ein vertikales und horizontales Histogramm könnte die Brett-Position validiert werden, indem diese auf regelmäßige Peaks an fest positionierten Stellen untersucht wird. Ein Frame, in dem das Brett nicht korrekt erkannt wird, kann somit verworfen werden.

6.3.6 Vergleich mit bestehender Literatur (Spielbretterkennung)

Die Arbeit von Seewald ist die bisher erfolgreichste Arbeit bei der Erkennung von Spielbrettern und erreichte eine Genauigkeit von 94% (Seewald, 2010b). Es handelte sich hierbei allerdings um kleinere 8x8 Spielbretter. Die Ergebnisse sind darüber hinaus nicht qualitativ vergleichbar, da sie unterschiedliche Testsets verwenden.

6.4 Evaluation der Go-Stellungserkennung

6.4.1 Bewertungskriterien (Stellungserkennung)

Folgende Bewertungskriterien wurden den Anforderungen (4.3) entnommen:

- Genauigkeit der Positionserkennung schwarzer Steine
- Genauigkeit der Positionserkennung weißer Steine
- Erkennung von leeren Punkten
- Fehlerresistenz gegenüber Schatten und Reflexionen
- Kompensation der Steinpositionen bei verschiedenen Blickwinkeln
- Genauigkeit der Erkannten Go-Stellung

6.4.2 Methodik (Stellungserkennung)

Das Testset wurde ebenfalls verwendet, um die Erkennungsrate einzelner Spielsteine zu überprüfen. Es beinhaltet Fälle zu allen festgelegten Bewertungskriterien. Zur Erläuterung der verwendeten Metriken dient exemplarisch das Testfile „Ref15“ im Reiter „Black Stones“ in Tabelle 6.3. Von den 13 schwarzen Steinen wurden 11 korrekt identifiziert (True Positives [TP]), während 2 nicht erfasst wurden (False Negatives [FN]). Von den restlichen 348 Schnittpunkten, die nicht von schwarzen Steinen belegt waren, wurden 5 irrtümlich als von schwarzen Steinen belegt klassifiziert (False Positives [FP]), während 343 korrekt als nicht von schwarzen Steinen belegt erkannt wurden (True Negatives [TN]).

Würde man die Genauigkeit hier mit der üblichen Formel

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

bestimmen, ergäbe sich hier eine Genauigkeit von über 98%. In Anbetracht dessen, dass von 13 Steinen 2 nicht erkannt wurden, ist dies keine Metrik, welche die Erkennungsrate befriedigend repräsentiert. Der Wert wird beeinflusst von den vielen leeren Feldern, die korrekt als nicht schwarze Steine erkannt werden. Bessere Ergebnisse liefern der Recall und die Precision.

6 Evaluation und Diskussion der Ergebnisse

TestFile	Black Stones								White Stones							
	TP	FP	TN	FN	F1Score	Accuracy	Precision	Recall	TP	FP	TN	FN	F1Score	Accuracy	Precision	Recall
Digital	39	0	322	0	100,0%	100,0%	100,0%	100,0%	35	0	322	4	94,6%	98,9%	100,0%	89,7%
Ref1	50	0	311	0	100,0%	100,0%	100,0%	100,0%	53	0	308	0	100,0%	100,0%	100,0%	100,0%
Ref2	13	0	348	0	100,0%	100,0%	100,0%	100,0%	12	0	349	0	100,0%	100,0%	100,0%	100,0%
Ref3	13	0	348	0	100,0%	100,0%	100,0%	100,0%	12	0	349	0	100,0%	100,0%	100,0%	100,0%
Ref4	13	0	348	0	100,0%	100,0%	100,0%	100,0%	9	0	349	3	85,7%	99,2%	100,0%	75,0%
Ref5	11	5	343	2	75,9%	98,1%	68,8%	84,6%	9	0	349	3	85,7%	99,2%	100,0%	75,0%
Ref6	31	0	330	0	100,0%	100,0%	100,0%	100,0%	31	0	330	0	100,0%	100,0%	100,0%	100,0%
Ref7	31	0	330	0	100,0%	100,0%	100,0%	100,0%	31	0	330	0	100,0%	100,0%	100,0%	100,0%
Ref8	31	0	330	0	100,0%	100,0%	100,0%	100,0%	31	0	330	0	100,0%	100,0%	100,0%	100,0%
Ref9	30	2	328	1	95,2%	99,2%	93,8%	96,8%	31	0	330	0	100,0%	100,0%	100,0%	100,0%
Ref10	62	0	298	1	99,2%	99,7%	100,0%	98,4%	64	0	297	0	100,0%	100,0%	100,0%	100,0%
Ref11	61	1	297	2	97,6%	99,2%	98,4%	96,8%	64	0	297	0	100,0%	100,0%	100,0%	100,0%
Ref12	62	0	298	1	99,2%	99,7%	100,0%	98,4%	64	0	297	0	100,0%	100,0%	100,0%	100,0%
Ref13	8	0	353	0	100,0%	100,0%	100,0%	100,0%	9	0	352	0	100,0%	100,0%	100,0%	100,0%
Ref14	62	0	298	1	99,2%	99,7%	100,0%	98,4%	64	0	297	0	100,0%	100,0%	100,0%	100,0%
Reflectic	48	0	313	0	100,0%	100,0%	100,0%	100,0%	50	0	311	0	100,0%	100,0%	100,0%	100,0%
Tisch1	13	0	348	0	100,0%	100,0%	100,0%	100,0%	12	0	349	0	100,0%	100,0%	100,0%	100,0%
Tisch2	13	0	348	0	100,0%	100,0%	100,0%	100,0%	8	0	349	4	80,0%	98,9%	100,0%	66,7%
Tisch3	13	0	348	0	100,0%	100,0%	100,0%	100,0%	12	0	349	0	100,0%	100,0%	100,0%	100,0%
Tisch4	13	0	348	0	100,0%	100,0%	100,0%	100,0%	12	0	349	0	100,0%	100,0%	100,0%	100,0%
Tisch5	22	0	339	0	100,0%	100,0%	100,0%	100,0%	22	0	339	0	100,0%	100,0%	100,0%	100,0%
Tisch6	21	0	339	1	97,7%	99,7%	100,0%	95,5%	22	0	339	0	100,0%	100,0%	100,0%	100,0%
Tisch7	22	0	339	0	100,0%	100,0%	100,0%	100,0%	22	0	339	0	100,0%	100,0%	100,0%	100,0%
Tisch8	22	0	339	0	100,0%	100,0%	100,0%	100,0%	22	0	339	0	100,0%	100,0%	100,0%	100,0%
Tisch9	36	0	325	0	100,0%	100,0%	100,0%	100,0%	37	0	324	0	100,0%	100,0%	100,0%	100,0%
Tisch10	36	0	325	0	100,0%	100,0%	100,0%	100,0%	37	1	323	0	98,7%	99,7%	97,4%	100,0%
Tisch11	47	0	314	0	100,0%	100,0%	100,0%	100,0%	45	1	315	0	98,9%	99,7%	97,8%	100,0%
Tisch12	47	0	314	0	100,0%	100,0%	100,0%	100,0%	45	1	315	0	98,9%	99,7%	97,8%	100,0%
Pc1	30	0	331	0	100,0%	100,0%	100,0%	100,0%	30	0	331	0	100,0%	100,0%	100,0%	100,0%
Pc2	30	0	331	0	100,0%	100,0%	100,0%	100,0%	30	0	331	0	100,0%	100,0%	100,0%	100,0%
Pc3	19	0	342	0	100,0%	100,0%	100,0%	100,0%	18	0	343	0	100,0%	100,0%	100,0%	100,0%
Pc4	9	0	352	0	100,0%	100,0%	100,0%	100,0%	9	0	352	0	100,0%	100,0%	100,0%	100,0%
Pc5	9	0	352	0	100,0%	100,0%	100,0%	100,0%	9	0	352	0	100,0%	100,0%	100,0%	100,0%
Pc6	19	0	342	0	100,0%	100,0%	100,0%	100,0%	18	1	342	0	97,3%	99,7%	94,7%	100,0%
Pc7	19	0	342	0	100,0%	100,0%	100,0%	100,0%	18	0	343	0	100,0%	100,0%	100,0%	100,0%
Bar1	0	0	361	0	NaN	100,0%	NaN	NaN	1	0	360	0	100,0%	100,0%	100,0%	100,0%
Bar2	5	0	356	0	100,0%	100,0%	100,0%	100,0%	5	0	356	0	100,0%	100,0%	100,0%	100,0%
Bar3	8	0	353	0	100,0%	100,0%	100,0%	100,0%	8	0	353	0	100,0%	100,0%	100,0%	100,0%
Bar4	8	0	353	0	100,0%	100,0%	100,0%	100,0%	8	0	353	0	100,0%	100,0%	100,0%	100,0%
Bar5	21	0	340	0	100,0%	100,0%	100,0%	100,0%	21	0	340	0	100,0%	100,0%	100,0%	100,0%
Bar6	21	0	340	0	100,0%	100,0%	100,0%	100,0%	21	0	340	0	100,0%	100,0%	100,0%	100,0%
Bar7	21	0	340	0	100,0%	100,0%	100,0%	100,0%	21	0	340	0	100,0%	100,0%	100,0%	100,0%
Bar8	36	0	325	0	100,0%	100,0%	100,0%	100,0%	35	0	326	0	100,0%	100,0%	100,0%	100,0%
Bar9	37	2	322	0	97,4%	99,4%	94,9%	100,0%	33	0	324	4	94,3%	98,9%	100,0%	89,2%
Bar10	41	0	320	0	100,0%	100,0%	100,0%	100,0%	41	1	319	0	98,8%	99,7%	97,6%	100,0%
Bar11	42	0	319	0	100,0%	100,0%	100,0%	100,0%	41	0	320	0	100,0%	100,0%	100,0%	100,0%
Bar12	40	0	319	2	97,6%	99,4%	100,0%	95,2%	41	0	320	0	100,0%	100,0%	100,0%	100,0%
Bar13	63	0	296	2	98,4%	99,4%	100,0%	96,9%	62	0	293	6	95,4%	98,3%	100,0%	91,2%
Bar14	65	0	296	0	100,0%	100,0%	100,0%	100,0%	66	0	293	2	98,5%	99,4%	100,0%	97,1%
SUM	1413	10	16253	13	99,19%	99,94%	99,30%	99,09%	1401	5	16257	26	98,91%	99,97%	99,64%	98,18%

Abbildung 6.3: Die Ergebnisse der Steinerkennung

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Ein hoher Recall bedeutet, dass das Modell die meisten der tatsächlich positiven Fälle korrekt identifiziert hat. Ein hoher Wert für Precision bedeutet, dass ein großer Anteil der vom Modell als positiv klassifizierten Fälle tatsächlich positiv ist. Mit einem Recall von 85% und einer Precision von 69% repräsentieren diese Werte die Probleme bei diesem Frame besser. In der Praxis wird oft ein Kompromiss zwischen Recall und Precision gesucht, da die Verbesserung des einen oft auf Kosten des anderen geht. Der F1-Score ist die Standardmetrik, die beide Aspekte berücksichtigt, und liegt für diesen Frame bei 76%.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

6.4.3 Ergebnisse (Stellungserkennung)

Die Resultate der Steinerkennung sind in Tabelle 6.3 dargestellt. Über alle Testcases hinweg beträgt der F1-Score für die Erkennung der schwarzen Steine 99,2% und für die weißen Steine 98,9%. Die Erkennung einzelner Steine ist also sehr zuverlässig.

Obwohl die Erkennung einzelner Steine sehr genau ist, führt dies zu einem Trugschluss bezüglich der Erkennungs-Qualität ganzer Go-Stellungen. Go-Stellungen bestehen aus Dutzenden von Steinen. Wird dabei nur ein Stein falsch oder zu viel erkannt, bedeutet dies, dass die ganze Go-Stellung falsch erkannt wurde. Daher weisen nur 30 der insgesamt 50 Go-Stellungen überhaupt keinen Fehler auf. Damit liegt die Zuverlässigkeit der Go-Stellungserkennung im Testset bei 60%.

6.4.4 Probleme (Stellungserkennung)

Probleme bei starken Reflexionen der Spielbrettoberfläche

Ein Problem, das bei der Erkennung weißer Steine auftreten kann, ist, dass Reflexionen auf dem Spielbrett so hell sind, dass sich die Spielsteine nicht gut vom Hintergrund absetzen. In Abbildung 6.4 ist in der linken oberen Ecke des Spielbretts an den Koordinaten 3-7 ein weißer Stein zu erkennen, der sogar dunkler ist als der Bretthintergrund. Dadurch, dass der lokale Bretthintergrund vom Basisbild abgezogen wird und der Hintergrund heller ist als der Spielstein, verschwindet der Stein im binären Bild.

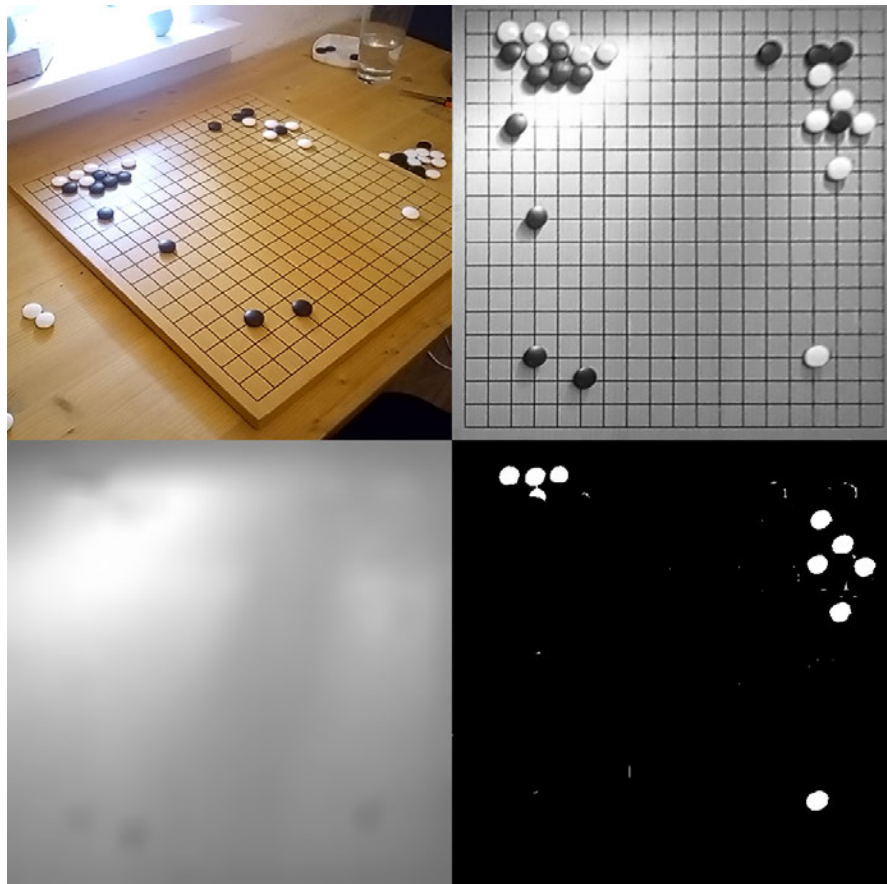


Abbildung 6.4: Fehlende Erkennung weißer Steine bei starken Brettreflexionen:
Links oben: Originalbild,
Rechts oben: Transformiertes Basisbild,
Links unten: Separierter Hintergrund,
Rechts unten: Binärbild der weißen Steine nach Abzug des Hintergrundes

Dieses Problem ist nicht trivial zu lösen. Der Mensch erkennt diesen Spielstein nur aufgrund des sichelförmigen dunklen Schattenwurfs des weißen Spielsteins. Die implementierte Steinerkennung nutzt aber nur Kontrastunterschiede zwischen den Spielsteinen und dem Spielbrett zur Erkennung. In der Praxis sollte der Nutzer der App die Belichtungsverhältnisse verbessern oder sich einen Blickwinkel suchen, der weniger Reflexionen unterliegt.

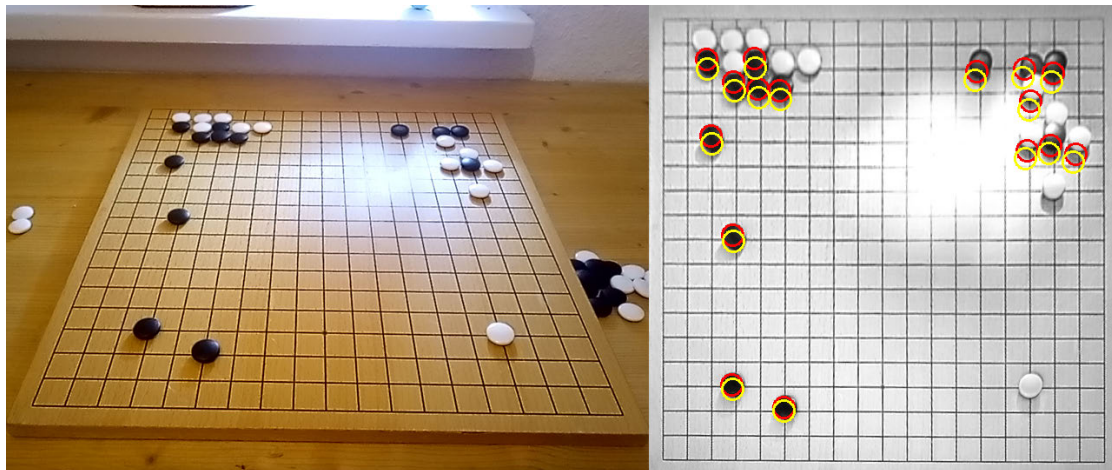


Abbildung 6.5: Missinterpretation von Schatten als schwarze Spielsteine:
Links: Originalbild,
Rechts: Erkannte Positionen schwarzer Spielsteine (gelb markiert)

Probleme durch starken Schattenwurf

Die sichelförmigen Schatten der Spielsteine können auch fehlerhaft als schwarze Spielsteine erkannt werden. Dieses Problem ist in Abbildung 6.5 in der rechten oberen Ecke des Spielbretts an den Koordinaten 6-4 und 6-2 zu sehen. Die runden, dunklen Schatten heben sich genau wie schwarze Spielsteine vom Bretthintergrund ab. Sie unterscheiden sich lediglich leicht durch ihre Form und ihre Intensität im lokalen Vergleich. Wird der Schatten zu groß, sodass er in etwa der Dimension eines Spielsteines entspricht, wird er als solcher fehlinterpretiert.

Auch dieses Problem ist nicht trivial zu lösen. Weiße Spielsteine erzeugen durch ihre Schatten sichelförmige schwarze Gebiete. Doch ein schwarzer Stein, mit einer hellen Reflexion wie auf 6-3, erzeugt ebenso eine ähnliche sichelförmige Struktur. Setzt man also eine komplett kreisförmige Struktur voraus, so werden schwarze Steine mit Reflexionen nicht mehr erkannt. Der Ansatz, nur mit Kontrastunterschieden zu arbeiten, gelangt hier also an seine Grenzen.

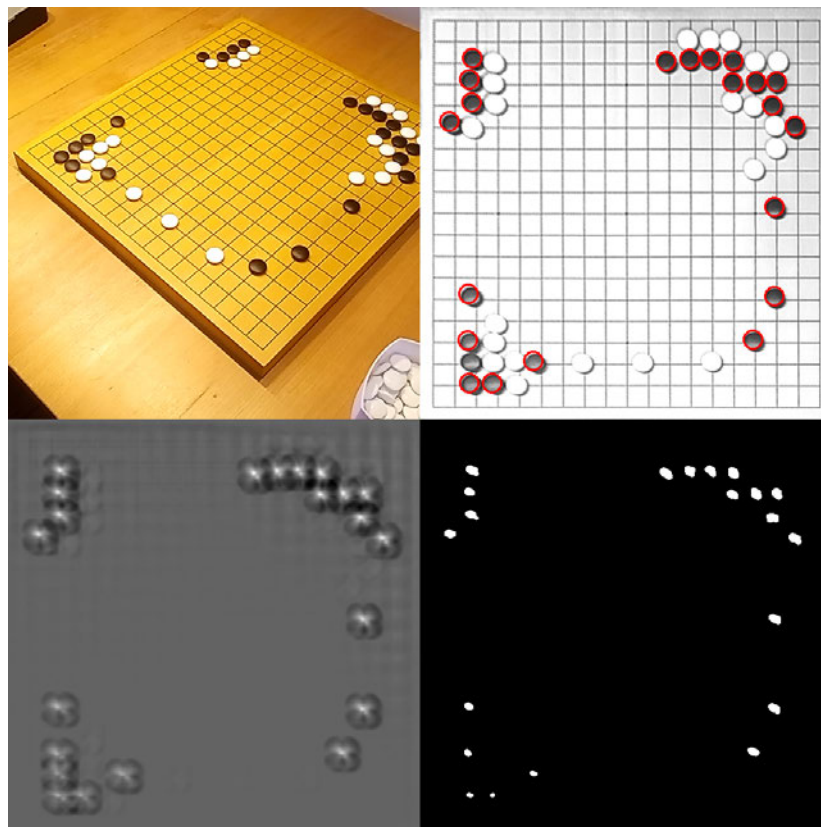


Abbildung 6.6: Fehlende Erkennung schwarzer Steine bei Reflexionen:
Links oben: Originalbild,
Rechts oben: Transformiertes Basisbild mit Steinerkennung,
Links unten: Heatmap der schwarzen Steine,
Rechts unten: Binärbild der schwarzen Steine nach Anwendung des Grenzwertes

Probleme durch Reflexionen auf schwarzen Spielsteinen

Sind Reflexionen auf einzelnen schwarzen Steinen zu stark, so sind sie nicht dunkel oder rund genug und werden als Schatten interpretiert. Ein solcher Fall ist in den unteren linken Brettarealen in 6.6 dargestellt. Die Heatmap ist nicht intensiv genug, um den benötigten Grenzwert zu überschreiten.

Um dieses Problem zu lösen, kann man den Grenzwert anpassen, sodass auch nur leicht dunkle Steine korrekt als schwarze Spielsteine erkannt werden. Dies ist jedoch ein unmittelbarer Trade-off mit den als schwarze Spielsteine erkannten Schatten. Diese Lösung verschiebt das Problem also nur.

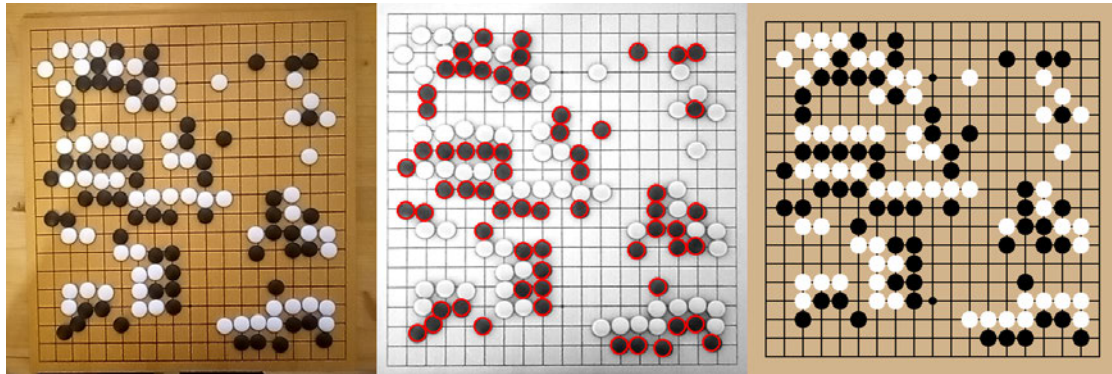


Abbildung 6.7: Missinterpretation von Schatten als schwarze Spielsteine:
Links: Originalbild,
Mitte: Erkannte schwarze Spielsteine,
Rechts: Spielsituation nach der Schnittpunktzuordnung

Probleme durch ungenaue Platzierung von Spielsteinen

Ungenau platzierte Spielsteine stellen eine Fehlerquelle dar, die unerwartet oft auftritt. Durch Ärmel- und Tischbewegungen kommt es oftmals zu ungeplanten Verschiebungen von Spielsteinen. In Abbildung 6.7 ist unten links eine solche Situation dargestellt. Die Steine werden zwar richtig erkannt, jedoch entspricht der dichteste Schnittpunkt nicht mehr dem erwünschten Schnittpunkt und zwei Steine werden demselben Schnittpunkt zugeordnet.

Für Spieler ist eine solche Situation einfach zu erkennen und zu korrigieren. Das Programm verfügt jedoch nicht über den Kontext der Spielhistorie und kann daher keinen Fehler detektieren. Spieler sollten sich daher vor einer Analyse über eine korrekte Spielsituation versichern.

6.4.5 Analyse und Interpretation der Ergebnisse (Stellungserkennung)

Bereits eine leichte Änderung der Stellung kann die Spielsituation bedeutsam ändern und bei der Analyse weitreichende Konsequenzen haben. Eine Erkennungsrate von nur 60% ist daher zunächst enttäuschend. Betrachtet man allerdings die Gründe hierfür, sind alle Probleme auf Fehler durch Beleuchtungsverhältnisse zurückzuführen. Unter guten Lichtverhältnissen werden schwarze und weiße Steine, leere Felder und damit die gesamte Go-Stellung stets korrekt erkannt. Ebenso wird die Spielsteinposition mit der in Kapitel 5.4.3 vorgestellten Methode erfolgreich an den Kamerawinkel angepasst.

In der Praxis lässt sich nicht leugnen, dass es häufig unzureichende Beleuchtungsverhältnisse gibt, die sich nicht einfach verbessern lassen. Ein Großteil dieser Beleuchtungsszenarien wird durch die in der Implementierung erläuterten Verarbeitungsschritte korrekt behandelt. Die vorgestellten Probleme sind jedoch mit herkömmlichen Methoden der Bildverarbeitung schwer zu bewältigen. Obwohl es für einen Menschen leicht ist, weiße und schwarze Steine zu unterscheiden, ist es schwierig, abstrakte Regeln zu formulieren, die alle Sonderfälle berücksichtigen. Wenn ein Regelwerk so komplex wird, dass es für einen Menschen schwierig ist, dieses zu gestalten, deutet dies darauf hin, dass herkömmliche Ansätze hierfür nicht mehr passend sind. Stattdessen legt dies nahe, dass der Einsatz von Convolutional Neural Networks eine erstrebenswertere Lösung darstellt. Mittels überwachtem Lernen kann das Programm selbst die Regeln erlernen, die notwendig sind, um Schnittpunkte als schwarze, weiße oder leere Felder zu klassifizieren.

6.4.6 Weitere Verbesserungsmöglichkeiten in der Stellungserkennung

Automatisches tuning der Parameter

Die Parameter, die zur Bestimmung von Schwellenwerten und anderen Faktoren genutzt werden, könnten durch Training an einem umfangreichen Testset justiert werden, um optimale Ergebnisse zu erzielen.

Nutzung mehrerer Frames

Steine könnten zuverlässiger erkannt werden, indem sie über mehrere Bilder hinweg verfolgt werden. Dies ermöglicht es, bei Bewegungen der Kamera die Historie in Betracht zu ziehen, um durch eine Mehrheitsentscheidung die Steine zuverlässiger zu erkennen.

6.4.7 Vergleich mit bestehender Literatur (Stellungserkennung)

Die Arbeit von Seewald stellt auch in diesem Bereich die bisher erfolgreichste Arbeit bei der Erkennung von ganzen Go-Stellungen aus Einzelbildern dar und erreichte eine Genauigkeit von 73% (Seewald, 2010b). Allerdings handelte es sich hierbei um 8x8-Spielbretter und nicht um die üblichen 19x19 Spielbretter. Ein kleineres Spielbrett erleichtert die korrekte Erkennung der gesamten Stellung bei einer gleichmäßigen Erkennungsrate einzelner Steine, da insgesamt weniger Steine auf dem Brett vorhanden sind. Zudem wurde ein anderes Testset verwendet, was einen qualitativen Vergleich erschwert.

6.5 Evaluation der Spielanalyse

6.5.1 Bewertungskriterien (Spielanalyse)

Folgende Bewertungskriterien wurden den Anforderungen (4.3) entnommen:

- Angabe einer Spielstandsbewertung
- Angabe einer Zugempfehlung
- Effizienz und Schnelligkeit der Analyse
- Vermeidung von Überlastung der Analyse-API
- Asynchronität der Spielanalyse

6.5.2 Methodik und Ergebnisse (Spielanalyse)

Bei jedem Bild aus dem Testset, bei dem eine Go-Stellung erfolgreich erkannt wird, erfolgt stets auch die Ermittlung einer Spielstandsbewertung samt Zugempfehlung. Nach Angaben der Entwickler beträgt die durchschnittliche Antwortzeit der API und damit auch für diese App etwa 0,25 Sekunden. Durch die Implementierung von Caching für Spielstellungen und einer begrenzten Anfragerate wurde die Schnittstelle zu keinem Zeitpunkt überlastet. Die gewählte Architektur implementiert eine asynchrone Spielanalyse.

6.5.3 Probleme (Spielanalyse)

Da die Analyse über eine externe API ausgeführt wird, ist stets eine Internetverbindung nötig.

6.5.4 Analyse und Interpretation der Ergebnisse (Spielanalyse)

Alle gestellten Anforderungen wurden erfüllt. Jedoch ist eine Evaluation, ob es sich stets um eine korrekte Spielstandsbewertung und Zugempfehlung handelt, nicht möglich. Dies liegt daran, dass die künstliche Intelligenz in diesem Bereich deutlich leistungsfähiger als jeder Mensch ist, wodurch eine Vergleichsbasis fehlt.



Abbildung 6.8: Die Benutzeroberfläche mit Debug-Menü

6.6 Evaluation der Ergebnisdarstellung und des Userinterfaces

6.6.1 Bewertungskriterien (Ergebnisdarstellung und Userinterface)

Folgende Bewertungskriterien wurden den Anforderungen (4.3) entnommen:

- Minimalistische und intuitive Benutzeroberfläche.
- Einstellungsmöglichkeit zur Festlegung, welcher Spieler am Zug ist.
- Darstellung des optimalen nächsten Zuges und Anzeige des aktuellen Punktestands.
- Immersive und perspektivisch korrekte Präsentation der Ergebnisse.
- Klarheit und Verständlichkeit in der Darstellung der Ergebnisse.

6.6.2 Methodik und Ergebnisse (Ergebnisdarstellung und UI)

Wie in Abbildung 6.8 zu sehen, besteht die Benutzeroberfläche lediglich aus dem modifizierten Kamerabild und einem Button zur Auswahl des aktiven Spielers, dessen Farbe sich entsprechend anpasst. Die Darstellung des Punktestands und des optimalen Zuges erfolgen integriert im Bild, perspektivisch korrekt und stets in der Farbe des aktiven Spielers.

6.6.3 Probleme (Ergebnisdarstellung und Userinterface)

Durch das minimalistische Interface und der augmentierten Darstellung kann der Nutzer nicht überprüfen, ob die Spielstellung korrekt von der App erfasst wurde. Dies ist bei der derzeit niedrigen Erkennungsrate der Stellungen problematisch. Eine mögliche Lösung wäre die Integration des bestehenden Debug-Modus, der beispielsweise durch längeres Drücken auf dem Bildschirm die abstrakte erkannte Stellung anzeigt.

6.6.4 Analyse und Interpretation der Ergebnisse (Ergebnisdarst./UI)

Die Frage nach der Immersivität der Anwendung stellt sich als anspruchsvoll dar. Nach Agrewal ist immersion "a phenomenon experienced by an individual when they are in a state of deep mental involvement in which their cognitive processes (with or without sensory stimulation) cause a shift in their attentional state such that one may experience disassociation from the awareness of the physical world" [however] "It is important to mention that the conviction of immersion being an objective property of a system/technology is held by a minority in the literature" (Agrewal u. a., 2020). Normalerweise erfolgt daher die Bewertung der Immersion durch Benutzertests und Fragebögen, doch aufgrund des begrenzten Umfangs dieser Arbeit ist dies nicht umsetzbar. In diesem Projekt ist Immersion definiert als ein interaktives Eintauchen in die Analyse des Go-Spiels, das die Spieler in ihrer physischen Umgebung verharren lässt, anstatt sie in ein digitales Interface zu ziehen. Dieses Ziel wird in dieser Anwendung durch die Integration von Augmented Reality erreicht, wobei die Darstellung des besten Zuges und des aktuellen Spielstands direkt in das physische Bild integriert wird und somit auf jegliche abstrakte Darstellung verzichtet. Es ist wichtig zu beachten, dass die Darstellung der Ergebnisse gemäß der Definition von Azuma (2.4) technisch gesehen nicht als Augmented Reality betrachtet werden kann, da für dies eine 3D-Registrierung erforderlich ist. In diesem Projekt wird die Positionierung jedoch lediglich durch 2D-Positionierung mit geschickter Verzerrung und Skalierung imitiert. Praktisch ist jedoch kein Unterschied zu einer echten AR-Darstellung feststellbar. Die Interaktivität wird gewährleistet, indem die App dynamisch auf die vom Nutzer vorgenommenen Änderungen der Brettstellungen reagiert.

Insgesamt wurden alle vorgegebenen Anforderungen erfüllt. Das Interface gestaltet sich mit nur zwei Elementen minimalistisch und ist intuitiv bedienbar – der Nutzer muss lediglich die Kamera auf das Spielbrett richten. Durch die Farbgebung des empfohlenen Zuges wird schnell ersichtlich, ob der falsche aktive Spieler ausgewählt wurde.

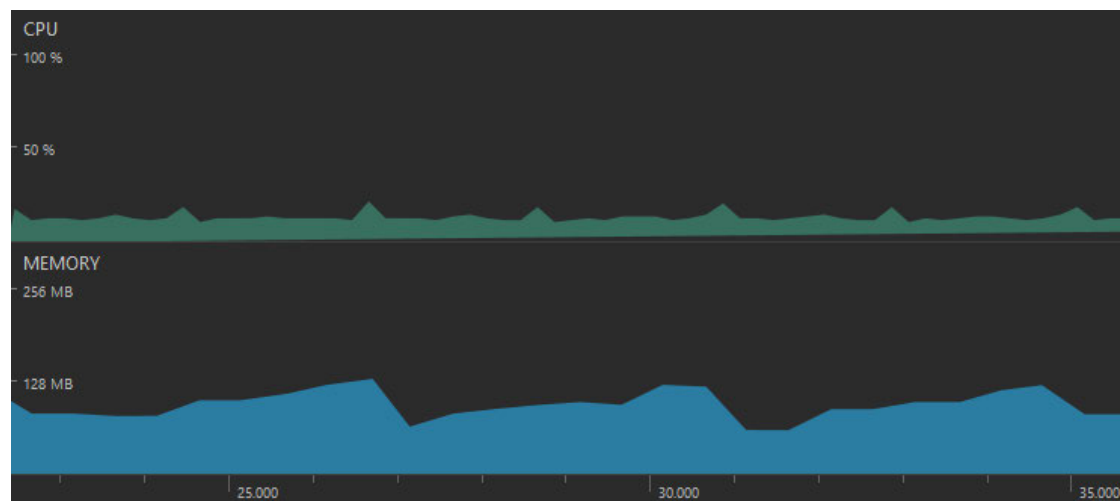


Abbildung 6.9: Speicher- und CPU-Auslastung der App

6.7 Evaluation der Technischen Leistungsfähigkeit

6.7.1 Bewertungskriterien (Technischen Leistungsfähigkeit)

Folgende Bewertungskriterien wurden den Anforderungen (4.3) entnommen:

- Reaktionszeit der App von unter 5 Sekunden.
- Stabilität und Zuverlässigkeit.
- Prävention von Speicherlecks.

6.7.2 Methodik und Ergebnisse (Technischen Leistungsfähigkeit)

Für die Leistungsbewertung der Anwendung wurde der „Low Overhead Profiler“ in Android Studio verwendet. Dieses Tool ermöglichte das Monitoring von CPU-Auslastung und Speicherverbrauch während der Analyse des Testsets. Die Durchführung aller Tests verlief stabil, und es kam während der Testphase zu keinen Programmabstürzen.

Die Ergebnisse des Monitorings sind in Abbildung 6.9 dargestellt. Die CPU-Auslastung beträgt ungefähr 15%. Vom Speicher werden bis zu 130 MB genutzt.

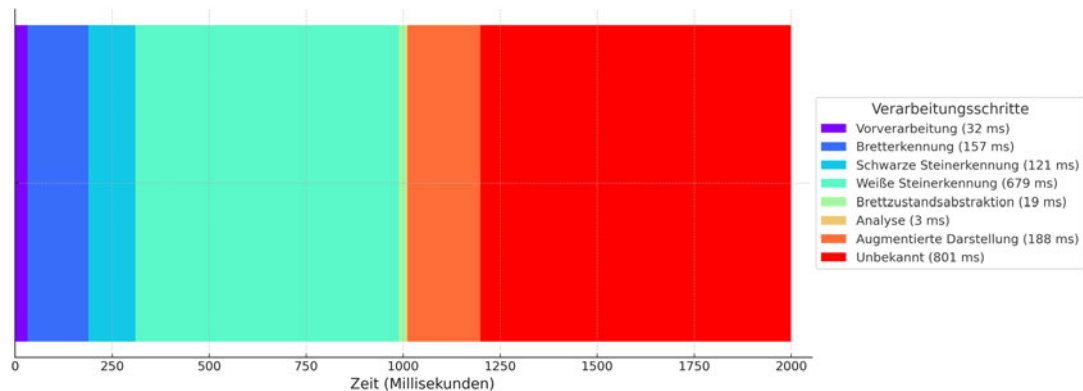


Abbildung 6.10: Zeitliche Aufschlüsselung der Verarbeitungsschritte

Die regelmäßigen Peaks in der CPU-Auslastung deuten darauf hin, dass ein Verarbeitungszyklus eines Bildes ungefähr 2 Sekunden in Anspruch nimmt. Eine detailliertere Aufschlüsselung der Zeitverwendung der einzelnen Komponenten ist in Abbildung 6.10 dargestellt. Da die Analyse der Go-Stellungen extern durchgeführt wird, fällt ihr Ressourcenverbrauch vernachlässigbar gering aus. Geringfügige Zeitaufwendungen entstehen durch das Preprocessing und die Abstraktion des Brettzustands. Wesentliche, gleichmäßige Anteile der Verarbeitungszeit werden von der Brett-Erkennung, der Erkennung schwarzer Steine und der augmentierten Darstellung in Anspruch genommen. Der überwiegende Teil der Verarbeitungszeit entfällt auf die Erkennung weißer Steine. Ein unerklärlicher Zeitabschnitt von 800 ms lässt sich keinem spezifischen Modul zuordnen.

In Abbildung 6.11 sind die Funktionen aller Module der App aufgelistet, die am meisten Rechenzeit verbrauchen. `inpaintWithMask` nimmt etwa 25% der Rechenzeit ein und ist bei der Erkennung der weißen Steine dafür zuständig, die maskierten Bereiche mit der näheren Umgebung zu füllen, wie es in 5.4.4 beschrieben wurde. Einen ähnlich großen Anteil nimmt die Funktion `getBoardCoordinatesForStones` ein. Sie ist dafür zuständig, den erkannten Mittelpunkt eines Steines dem nächstgelegenen Schnittpunkt zuzuordnen. `draw3D` ist für die augmentierte Darstellung des besten Zuges und des Spielstandes zuständig und benötigt etwa 6%.

Method	Call Count	Execution Time (ms)
processImage() (com.example.b459958694...)	12,343,888	87.55
inpaintWithMask() (com.example.b4599...)	3,408,010	24.17
getBoardCoordinatesForStones() (com.e...)	3,246,857	23.03
draw3D() (com.example.b459958694.Im...)	796,097	5.65
findGridEdgeClusters() (com.example.b...)	764,786	5.42
refineStonePositionsUsingHeatmap() (c...)	754,431	5.35
findGridContour() (com.example.b4599...)	720,208	5.11
getBlackStonesHeatmap() (com.examp...)	517,960	3.67
getWhiteStonesHeatmap() (com.examp...)	427,624	3.03
drawGridOnImage() (com.example.b459...)	170,892	1.21
getContours() (com.example.b45995869...)	167,573	1.19

Abbildung 6.11: Profilierung der Methodenaufrufe

6.7.3 Probleme (Technischen Leistungsfähigkeit)

Speicherleaks durch Matrizen

In OpenCV werden Bilder als Matrizen gespeichert. Obwohl das Programm in Java implementiert ist, nutzt die OpenCV-Bibliothek im Hintergrund C-Code. Da C keine automatische Speicherverwaltung bietet, muss der Speicher für diese Matrizen manuell verwaltet werden. Der Java-Wrapper von OpenCV verweist lediglich auf die im C-Code erstellten Matrizen. Daher ist es notwendig, Matrizen auch auf der Java-Seite manuell freizugeben, um Speicherleaks zu vermeiden. Dies wurde erfolgreich implementiert.

Geringe Bildrate bei der Ausrichtung der Kamera

Die Bildfrequenz ist gering. Dies ist darauf zurückzuführen, dass die Bilddarstellung und Analyse gemeinsam in einem großen Mainloop stattfinden. Diese niedrige Bildrate führt zu Schwierigkeiten bei der Orientierung und Ausrichtung der Kamera auf das Spielbrett. Um dieses Problem zu beheben, sollte die Darstellung von der Analyse entkoppelt werden, was eine Erhöhung der Framerate ermöglichen würde.

6.7.4 Analyse und Interpretation der Ergebnisse (Technischen Leistungsfähigkeit)

Die durchschnittliche Analysezeit eines einzelnen Bildframes liegt bei etwa 2 Sekunden. Dies wurde sowohl während der Ausführung des Testsets als auch bei manuellen Messungen festgestellt, ohne dass es zu Speicherlecks oder Systemabstürzen kam. Somit erfüllt das Programm alle definierten Anforderungen.

Jedoch ergeben sich einige interessante Beobachtungen. Manche Funktionen sind so ineffizient, dass sie das Gesamte Moduls ausbremsen. Ein Beispiel hierfür ist die Erkennung weißer Steine, bei der eine einfache Fill-Funktion einen Großteil der Rechenkapazität beansprucht.

CPU- und Speicherauslastung bleiben stets niedrig. Die Applikation wird primär durch die Leistung eines einzelnen CPU-Threads begrenzt, da sie sequenziell aufgebaut ist und nicht für Multithreading optimiert wurde.

Der unidentifizierte Zeitabschnitt kann teilweise dem Overhead des Frameworks und den GUI-Updates zugeschrieben werden. Auch der Profiler selbst könnte die Zeitmessungen beeinflussen. Schätzungsweise sollten jedoch all diese Faktoren zusammen nicht mehr als 50 ms beanspruchen. Ein beträchtlicher Teil der unzugeordneten Zeit bleibt somit unerklärt und bedarf weiterer Untersuchungen.

Leistungsverbesserungen einzelner Funktionen

Die Funktion `inpaintWithMask` wird durch die OpenCV-Funktion `Photo.inpaint` limitiert. Diese ist ineffizient beim Auffüllen größerer Lücken mit ihrer Umgebung. Das zugrundeliegende Problem ist jedoch nicht komplex und sollte durch die Anwendung einer effizienteren Methode auf wenige Millisekunden reduzierbar sein.

`getBoardCoordinatesForStones` beruht hauptsächlich auf mathematischen Operationen, wie der Berechnung des 2-Norm-Abstands der Steinmittelpunkte zu verschiedenen Schnittpunkten. Da diese Operation trivial ist, ist der hohe Zeitaufwand hier ungewöhnlich und unnötig. Bei einer korrekten Implementierung sollte der Zeitaufwand weniger als 1 Millisekunde betragen.

draw3D wird verwendet, um das Overlay über das Originalbild zu legen und einen 3D-Effekt durch Layering zu erzeugen. Da OpenCV nicht primär für grafisches Rendering konzipiert ist und die Implementierung nicht optimiert wurde, sollte für die grafische Darstellung des Overlays eine alternative gesucht werden.

Parallelisierung von Funktionen

Viele Verarbeitungsschritte der Pipeline innerhalb eines Moduls könnten parallel ausgeführt werden. Zum Beispiel könnte das Preprocessing für schwarze und weiße Steine auf verschiedene Worker aufgeteilt werden.

Entkopplung der Module

Die Performance der App könnte weiter verbessert werden, indem die einzelnen Module entkoppelt werden und asynchron miteinander kommunizieren. Die Module könnten dann auf unterschiedliche Threads verteilt werden. Schnellere Module würden bei ihren Aufgaben nicht von langsameren ausgebremst. Dies könnte umgesetzt werden, indem die Module für Bretterkennung, Stellungserkennung und Darstellung entkoppelt werden und eigenständig operieren. Für die aktuelle Anwendung wurde diese Architektur nicht verwendet, da sie deutlich komplizierter und somit fehleranfälliger ist und der explorativen Natur dieses Projektes entgegenstand.

6.7.5 Ausblick (Technischen Leistungsfähigkeit)

Angesichts des explorativen Charakters dieser Arbeit wurden bisher keine Anstrengungen zur Performance-Optimierung der App unternommen. Durch einen gezielten Fokus auf dieses Gebiet und der Umsetzung der erwähnten Maßnahmen sollte die vollständige Analysezeit schätzungsweise auf unter 500 Millisekunden reduziert werden können, wobei 250 Millisekunden auf die Implementierung der App und 250 Millisekunden auf die externe Analyse durch die API entfallen würden. Zusätzlich sollte die Reaktionsgeschwindigkeit der App so weit verbessert werden können, dass Frames in Echtzeit gerendert werden, ohne Frames zu überspringen. Dabei müssten gecachte Informationen aus der langsameren Analyse zum Einsatz kommen, die nicht vollkommen aktuell sind, also bis zu etwa 500 Millisekunden alt sein könnten.

6.8 Evaluation der Benutzerfreundlichkeit und Kompatibilität

6.8.1 Bewertungskriterien (Benutzerfreundlichkeit und Kompatibilität)

Folgende Bewertungskriterien wurden den Anforderungen (4.3) entnommen:

- Kompatibilität mit Android API 26 oder höher.
- Funktionstüchtigkeit auf schwächeren Geräten.
- Anforderung benötigter Zugriffsrechte.
- Sicherstellung des Datenschutzes.

6.8.2 Methodik und Ergebnisse (Benutzerfreundlichkeit und Kompatibilität)

Die App wurde auf einem Smartphone mit Android API 31 getestet, wobei das Projekt in Android Studio für die Kompatibilität mit Android API 26 konfiguriert wurde. Sämtliche Tests wurden auf dem zuvor genannten leistungsschwachen Gerät durchgeführt. Die erforderlichen Zugriffsrechte für die Kamera werden beim Start der App angefragt, sofern diese noch nicht erteilt wurden. Es werden keine personenbezogenen Daten durch die App erhoben. Die Analyse beschränkt sich auf die Daten des Go-Spiels, und es werden keinerlei Daten gespeichert.

6.8.3 Analyse und Interpretation der Ergebnisse (Benutzerfreundlichkeit und Kompatibilität)

Die Ergebnisse zeigen, dass die App die gesetzten Anforderungen in den Bereichen Kompatibilität, Leistung auf schwächeren Gerätetypen, Zugriffsrechten und Datenschutz erfüllt. Die App sollte jedoch noch auf einem physischen Smartphone mit Android API 26 getestet werden. Dies stand dem Autor jedoch nicht zur Verfügung.

6.9 **Fazit der Evaluation**

Insgesamt zeigt die Evaluation, dass das Projekt ein leistungsstarkes und benutzerfreundliches System zur Echtzeitanalyse physischer Go-Partien darstellt. Es ist wichtig zu erwähnen, dass die Qualität der Analyseergebnisse und die Performance des Systems maßgeblich von der verwendeten Hardware abhängen. Ein modernes Testgerät mit einer hochwertigen Kamera würde bessere Ergebnisse bei der Erkennung der Go-Stellungen liefern. Ebenso könnte die Verarbeitungsgeschwindigkeit, durch leistungsfähigere Hardware erheblich verbessert werden.

Während einige Herausforderungen und Verbesserungsmöglichkeiten identifiziert wurden, insbesondere in Bezug auf Beleuchtungsbedingungen und Erkennungsrate ganzer Go-Stellungen, liefert die Arbeit einen wertvollen Beitrag zur Weiterentwicklung von Analysetools in diesem Bereich.

7 Fazit und Ausblick

In einer Zeit, in der digitale Technologie und Realität zunehmend verschmelzen, zielt diese Arbeit darauf ab, das traditionelle Go-Spiel mit moderner Technologie zu erweitern. Das Hauptziel bestand darin, ein mobiles Anwendungssystem zu entwickeln, das sich nahtlos in das gewohnte Umfeld der Spielenden einfügt. Es sollte eine Brücke zwischen den sozialen Vorteilen des physischen Brettspiels und den Vorzügen der digitalen Analyse schlagen. Die Forschungsfrage lautete: Wie kann ein mobiles Anwendungssystem entwickelt werden, das eine intuitive, schnelle und automatisierte Analyse von physischen Brettspielen, speziell anhand des Beispiels Go, ermöglicht und sich immersiv in spontane Spielsituationen einfügt, ohne dabei auf eine manuelle Digitalisierung angewiesen zu sein?

Die vorgestellte Anwendung unterscheidet sich von bestehenden Analyse-Tools durch ihre Fähigkeit, eine unmittelbare Analyse physischer Partien in Echtzeit zu ermöglichen. Sie nutzt die integrierte Kamera eines Smartphones für die automatische Ad-hoc-Erkennung der Spielsituationen.

Die Arbeit leistet einen bedeutenden Beitrag zur Forschung durch die Einführung innovativer Methoden. Dazu gehört die Farbintensitätsanalyse zur Reduzierung von Reflexionen und die verbesserte Positionserkennung des Brettes durch Konturennutzung. Darüber hinaus stellt die Integration von Augmented Reality zur unmittelbaren Darstellung von Analyseergebnissen einen zukunftsweisenden Ansatz dar, der durch Immersion die Spielatmosphäre aufrechterhält.

Mit einer Erkennungsrate von 98% bei der Spielbretterkennung meistert das System verschiedene Herausforderungen wie unterschiedliche Kamerawinkel und Umgebungen. Nach der Recherche des Autors stellt dies das bisher erfolgreichste Projekt in diesem Bereich dar. Auch die Einzelsteinerkennung erreicht einen F1-Score von 99%. Dies resultiert in einer Gesamtstellungserkennungsrate von 60% und ist damit, trotz niedriger Kameraqualität, nach Einschätzung des Autors ebenfalls das erfolgreichste Ergebnis auf

dem 19x19-Brett. Obwohl dies einen wichtigen Fortschritt darstellt, genügt dies nicht für den zuverlässigen Betrieb der App. Während die Erkennung in kontrollierten Umgebungen durchgängig erfolgreich ist, zeigt das anspruchsvolle Testset mit seinen vielen Reflexionen, wie sie in natürlichen Umgebungen auftreten, die Limitationen herkömmlicher Bildverarbeitungsmethoden auf. Trotz umfassender Kompensationsbemühungen stößt das System hier an seine Grenzen. Diese Erkenntnisse unterstreichen die wachsende Bedeutung von Convolutional Neural Networks in diesem Forschungsbereich der Klassifizierung. Die App bietet Spielanalysen auch auf schwächeren Systemen in angemessener Zeit, bietet aber noch einen deutlichen Raum für Performance-Optimierungen. Die minimalistische und intuitive Benutzeroberfläche, kombiniert mit einer innovativen Augmented-Reality-Darstellung, trägt zu einem benutzerfreundlichen Spielerlebnis bei.

Zukünftige Forschungen sollten sich vor allem darauf konzentrieren, eine zuverlässige Erkennung von Spielsteinen mithilfe von Convolutional Neural Networks zu ermöglichen. Ein wichtiger Schritt hierbei ist die Zusammenstellung eines umfangreichen, abwechslungsreichen und realitätsnahen Testsets für Spielsteine und Brett-Positionen. Ein solches Testset würde nicht nur die Voraussetzungen für ein erfolgreiches überwachtes Training schaffen, sondern auch eine einheitliche Basislinie für den Vergleich verschiedener Forschungsarbeiten bieten.

Insgesamt zeigt diese Arbeit, wie Computer-Vision-Techniken und Augmented Reality genutzt werden können, um die Analyse und das Spielerlebnis von physischen Brettspielen mit mobilen Geräten zu revolutionieren. Die Integration moderner Technologien in einer umfassenden All-in-One-Lösung bietet eine innovative Lösung, die die Forschungsfrage zielgerichtet beantwortet und das Potenzial hat, die Art und Weise, wie wir lernen, grundlegend zu verändern.

Literaturverzeichnis

- [Agrewal u. a. 2020] AGREWAL, Sarvesh ; SIMON, Adèle Maryse D. ; BECH, Søren ; BÆRENSEN, Klaus B. ; FORCHAMMER, Søren: Defining immersion:: literature review and implications for research on audiovisual experiences. In: *Journal of the Audio Engineering Society* 68 (2020), Nr. 6, S. 404–417
- [Association 2023] ASSOCIATION, Japan G.: *How to Play Go*. 2023. – URL <https://www.nihonkiin.or.jp/english/howto/howto0.htm>. – Zugriffsdatum: 2023-10-22
- [Azuma 1997] AZUMA, Ronald T.: A survey of augmented reality. In: *Presence: teleoperators & virtual environments* 6 (1997), Nr. 4, S. 355–385
- [Browne u. a. 2012] BROWNE, Cameron B. ; POWLEY, Edward ; WHITEHOUSE, Daniel ; LUCAS, Simon M. ; COWLING, Peter I. ; ROHLFSHAGEN, Philipp ; TAVENER, Stephen ; PEREZ, Diego ; SAMOTHRAKIS, Spyridon ; COLTON, Simon: A survey of monte carlo tree search methods. In: *IEEE Transactions on Computational Intelligence and AI in games* 4 (2012), Nr. 1, S. 1–43
- [Fielding 2000] FIELDING, Roy T.: *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000
- [Hartley und Zisserman 2003] HARTLEY, Richard ; ZISSERMAN, Andrew: *Multiple view geometry in computer vision*. Cambridge university press, 2003
- [Heaton 2018] HEATON, Jeff: Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning: The MIT Press, 2016, 800 pp, ISBN: 0262035618. In: *Genetic programming and evolvable machines* 19 (2018), Nr. 1-2, S. 305–307
- [Hirsimäki 2005] HIRSIMÄKI, Teemu: Extracting go game positions from photographs. In: *Helsinki University of Technology, Tech. Rep.* (2005)

- [Kang u. a. 2005] KANG, Deuk C. ; KIM, Ho J. ; JUNG, Kyeong H.: Automatic extraction of game record from TV Baduk program. In: *The 7th International Conference on Advanced Communication Technology, 2005, ICACT 2005*. Bd. 2 IEEE (Veranst.), 2005, S. 1185–1188
- [Loy und Zelinsky 2003] LOY, Gareth ; ZELINSKY, Alexander: Fast radial symmetry for detecting points of interest. In: *IEEE Transactions on pattern analysis and machine intelligence* 25 (2003), Nr. 8, S. 959–973
- [Molla und Lepetit 2010] MOLLA, Eray ; LEPETIT, Vincent: Augmented reality for board games. In: *2010 IEEE International Symposium on Mixed and Augmented Reality* IEEE (Veranst.), 2010, S. 253–254
- [Müller 2002] MÜLLER, Martin: Computer go. In: *Artificial Intelligence* 134 (2002), Nr. 1-2, S. 145–179
- [Müller 2020] MÜLLER, Vincent C.: Ethics of artificial intelligence and robotics. (2020)
- [Seewald 2010a] SEEWALD, Alexander K.: Automatic extraction of go game positions from images: A multi-strategical approach to constrained multi-object recognition. In: *Applied Artificial Intelligence* 24 (2010), Nr. 3, S. 233–252. – Bespricht die Ergebnisse von Scher, S. (2006). Hidden Markov Model Improves a Weak Classifier. Unveröffentlichter Bericht, zugänglich unter: <http://www.soe.ucsc.edu/classes/cmcs290c/Winter06/proj/stevenreport.doc>. Zugriff erfolgte über Seewald.
- [Seewald 2010b] SEEWALD, Alexander K.: Automatic extraction of go game positions from images: A multi-strategical approach to constrained multi-object recognition. In: *Applied Artificial Intelligence* 24 (2010), Nr. 3, S. 233–252
- [Shiba und Mori 2004] SHIBA, Kojiro ; MORI, Kunihiko: Detection of Go-board contour in real image using genetic algorithm. In: *SICE 2004 Annual Conference* Bd. 3 IEEE (Veranst.), 2004, S. 2754–2759
- [Shotwell 2011] SHOTWELL, Peter: *Go! More than a game*. Tuttle Publishing, 2011
- [Silver u. a. 2016] SILVER, David ; HUANG, Aja ; MADDISON, Chris J. ; GUEZ, Arthur ; SIFRE, Laurent ; VAN DEN DRIESSCHE, George ; SCHRITTWIESER, Julian ; ANTONOGLOU, Ioannis ; PANNEERSHELVAM, Veda ; LANCTOT, Marc u. a.: Mastering the game of Go with deep neural networks and tree search. In: *nature* 529 (2016), Nr. 7587, S. 484–489

- [Silver u. a. 2017] SILVER, David ; SCHRIITWIESER, Julian ; SIMONYAN, Karen ; ANTONOGLOU, Ioannis ; HUANG, Aja ; GUEZ, Arthur ; HUBERT, Thomas ; BAKER, Lucas ; LAI, Matthew ; BOLTON, Adrian u. a.: Mastering the game of go without human knowledge. In: *nature* 550 (2017), Nr. 7676, S. 354–359
- [Sonka u. a. 2013] SONKA, Milan ; HLAVAC, Vaclav ; BOYLE, Roger: *Image processing, analysis and machine vision*. Springer, 2013
- [Szeliski 2022] SZELISKI, Richard: *Computer vision: algorithms and applications*. Springer Nature, 2022
- [Wu 2019] WU, David J.: Accelerating self-play learning in go. In: *arXiv preprint arXiv:1902.10565* (2019)

A Anhang

A.1 Verwendete Hilfsmittel


In der Tabelle A.1 sind die im Rahmen der Bearbeitung des Themas der Bachelorarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

Tool	Verwendung
Adobe Photoshop	Bearbeitung und Optimierung von Bildern und Grafiken
Android Studio	Entwicklungsumgebung für Android-Apps
ChatGPT	Bei dieser Arbeit: Zur Korrektur der Rechtschreibung und Grammatik, zur Findung besserer Formulierungen einzelner Sätze, zum Generieren von LaTeX-Code; nicht verwendet um ganze Textpassagen zu generieren. Bei Entwicklung: zum Refactoring einzelner Funktionen, zum Hinzufügen von Kommentaren, als Dokumentation zur Erklärung von OpenCV-Funktionen und ihrer Nutzung.
CUBOT P60	Testgerät
Datanalyst	Plotten von Daten und Abbildungen
Draw.io	Werkzeug zur Erstellung von Diagrammen und Skizzen
Fork for Git	Versionskontroll-Tool
GitHub	Plattform für die Verwaltung von Software-Projekten und Versionskontrolle
Google Drive	Cloud-Speicher für Dokumente und Dateien
Gradle 7.4.2	Build-Automatisierungstool
L ^A T _E X	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments
OpenCV 4.8.0	Bibliothek für Computer Vision und Bildverarbeitung

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.

<hr/>	<hr/>	
Ort	Datum	Unterschrift im Original