

CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL

INSTITUT FÜR INFORMATIK
ARBEITSGRUPPE INTELLIGENTE SYSTEME

MASTERARBEIT

NUTZUNG VON REALITÄTSNAHEN, SYNTHETISCH ERZEUGTEN
DATEN ZUR VERBESSERUNG DES KI-GESTÜTZTEN SCORINGS
VON STEELDARTS IN EINEM SINGLE-CAMERA-SYSTEM

Betreut durch: Prof. Dr. Sven Tomforde
M.Sc. Simon Reichhuber

Justin Fürstenwerth
2025

Kurzfassung

In dieser Masterarbeit wird ein System zum automatischen Scoring im Steeldarts auf Basis einzelner Aufnahmen von Dartscheiben entwickelt. Dafür wird eine Kombination von Techniken herkömmlicher Computer Vision (CV) und neuronaler Netze eingesetzt, durch welche Dartscheiben in beliebigen Bildern identifiziert und normalisiert werden und Dartpfeile in diesen vorverarbeiteten Aufnahmen lokalisiert werden. Das Training des neuronalen Netzes beruht auf synthetisch generierten, realitätsnahen Bildern von Dartscheiben, die durch prozedurale 3D-Modellierung nahezu fotorealistisch erstellt und automatisch annotiert werden.

Diese Masterarbeit baut auf einem von McNally et al. in 2021 vorgestellten System mit dem Namen DeepDarts auf. In diesem System wurden Schwachstellen identifiziert und gezielt angegangen. Die durch DeepDarts gewonnenen Erkenntnisse und Herangehensweisen dienten als methodische Grundlage für die Weiterentwicklung des Systems in dieser Arbeit.

Durch die in dieser Thesis erarbeiteten Systeme werden zufriedenstellende Ergebnisse in allen drei Themenbereichen dieser Arbeit – Datenerstellung, Normalisierung und Lokalisierung – erzielt. Hinsichtlich der Datenerstellung wird ein System vorgestellt, welches nahezu fotorealistische und variable Aufnahmen von Dartscheiben generiert und diese korrekt annotiert. Der Algorithmus zur Findung und Normalisierung von Dartscheiben in beliebigen Bildern erzielt eine Erfolgsrate von 97 % und einen mittleren Fehler von weniger als 5 %. Das neuronale Netz zur Lokalisierung der Dartfeilspitzen ist in der Lage, in etwa 60 % der eingegebenen Bilder korrekte Vorhersagen zu erzielen ohne Overfitting auf spezifischen Daten. Die Ergebnisse zeigen, dass das entwickelte System robuste Vorverarbeitungsschritte mit Lokalisierung durch neuronale Netze erfolgreich kombiniert und damit die Grundlage für ein automatisches Dart-Scoring bildet.

Abstract

This master thesis presents a system for automatic steel dart scoring, based on images of dart boards captured with a single camera. To achieve this, a combination of standard computer vision techniques and neural networks is employed, which are able to identify and normalize dart boards in arbitrary images and locate dart tips in those preprocessed images. For the neural network training synthetically generated realistic images of dart boards are used, which are generated through 3D modeling and are nearly photorealistic.

This master thesis stems from a system called DeepDarts, presented by McNally et al. in 2021. In DeepDarts, some weak points were identified and addressed in a controlled manner. The insights gained from that system serve as a methodological basis for the further development in this thesis.

Through the systems developed in this thesis, satisfactory results were achieved in all subject areas: data generation, normalization and localization. Regarding data generation, a system is developed which is able to produce nearly photorealistic and variable images of dart boards and their correct annotations. The algorithm for identifying and normalizing dart boards in arbitrary images has a mean success rate of 97 % and a mean error of less than 5 %. The neural network for localizing the dart tips is able to correctly predict about 60 % of presented images without overfitting to specific cases. These results show that the presented system combines a robust preprocessing with neural network predictions to create a solid basis for automatic dart scoring.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Projektübersicht	1
1.2 DeepDarts	2
1.3 Einsatz synthetischer Datenerstellung	3
1.4 Einsatz herkömmlicher Computer Vision	3
1.5 Einsatz neuronaler Netze	4
1.6 Forschungsfragen	5
1.7 Themenbezogene Arbeiten	6
1.7.1 Prozedurale Datenerstellung	6
1.7.2 Synthetische Datenerstellung für das Training neuronaler Netze	6
1.7.3 Dart-Scoring	6
1.8 Aufbau der Arbeit	7
2 Simulation von Dartscheiben zur Generierung von Trainingsdaten	9
2.1 Grundlagen	9
2.1.1 Prozedurale Datenerstellung	10
2.1.2 3D-Rendering	10
2.1.3 Kameraparameter	10
2.1.4 Binärbilder und Masken	11
2.1.5 Geometrie einer Dartscheibe	12
2.1.6 Dart-Terminologie	12
2.1.7 Steeldarts und Softdarts	13
2.1.8 Material und Texturen	13
2.1.9 Noise-Texturen	14
2.1.10 Seeding	14
2.1.11 Thresholding und Maskierung	15
2.1.12 Prozedurale Texturen	15
2.2 Methodik	16
2.2.1 Der Aufbau der 3D-Szene	17
2.2.2 Material und Licht	18
2.2.3 Scripting	20
2.2.4 Nachverarbeitung und Fertigstellung der Daten	21
2.3 Implementierung	22
2.3.1 Parametrisierung der Dartscheibe	22
2.3.2 Zusammensetzung der Dartpfeile	23
2.3.3 Generierung von Dartpfeilpositionen	24
2.3.4 Ermittlung von Kameraparametern	26
2.3.5 Render-Einstellungen	28
2.3.6 Berechnung von Entzerrung	29
2.4 Ergebnisse	29
2.4.1 Beispiele gerenderter Bilder	29
2.4.2 Rahmenbedingungen der Erstellung	31
2.4.3 Qualitative Auswertung	31
2.4.4 Quantitative Metadatenauswertung	32
2.4.5 Korrekte Annotation der Daten	35
2.4.6 Ungenauigkeiten der Datenerstellung	35

3 Vorverarbeitung von Bildern durch klassische Computer Vision	37
3.1 Grundlagen	38
3.1.1 Polarlinien	38
3.1.2 Thresholding	38
3.1.3 Binning	38
3.1.4 Faltung	39
3.1.5 Kantenerkennung	39
3.1.6 Harris Corner Detection	40
3.1.7 Hough-Transformation	40
3.1.8 Transformationsmatrizen	41
3.1.9 Log-polare Entzerrung	43
3.1.10 Farbräume	43
3.1.11 Structural Similarity	44
3.1.12 RANSAC	44
3.2 Methodik	44
3.2.1 Motivation der Verwendung herkömmlicher Computer Vision	46
3.2.2 Vorverarbeitung	46
3.2.3 Kantenverarbeitung	46
3.2.4 Linienverarbeitung	48
3.2.5 Orientierung	51
3.2.6 Zusammenführen aller Komponenten	55
3.3 Implementierung	55
3.3.1 Winkelfindung aus gefilterten Linien	56
3.3.2 Farben-Identifizierung	56
3.3.3 Klassifizierung von Surroundings	57
3.4 Ergebnisse	58
3.4.1 Metriken	58
3.4.2 Verwendete Daten	58
3.4.3 Quantitative Auswertung	59
4 Identifizierung von Dartpfeilen mit neuronalen Netzen	63
4.1 Grundlagen	63
4.1.1 Was sind neuronale Netze?	64
4.1.2 Training neuronaler Netze	65
4.1.3 Terminologie	65
4.1.4 Augmentierung von Trainingsdaten	66
4.1.5 Die YOLOv8-Architektur	67
4.2 Methodik	67
4.2.1 Die verwendete Architektur	67
4.2.2 Loss-Funktionen	70
4.2.3 Training	72
4.3 Implementierung	76
4.3.1 Implementierung der YOLOv8*-Architektur	76
4.3.2 Training von YOLOv8* zur Identifizierung von Dartpfeilen	76
4.4 Ergebnisse	78
4.4.1 Metriken	78
4.4.2 Datenquellen und Herangehensweise	80
4.4.3 Auswertung der Existenz-Metrik μ_{xst}	80
4.4.4 Auswertung der Klassen-Metrik μ_{cls}	81
4.4.5 Auswertung der Positions-Metrik μ_{pos}	82
4.4.6 Auswertung der PCS-Metrik	82
5 Diskussion	83
5.1 Diskussion der Datenerstellung	83
5.1.1 Datenumfang	83
5.1.2 Realismus der Daten	84
5.1.3 Genauigkeit der Datenerstellung	84
5.1.4 Effizienz der Datenerstellung	85
5.2 Diskussion der algorithmischen Normalisierung	85

5.2.1	Verwendete Technik	85
5.2.2	Zuverlässigkeit des Systems	86
5.3	Diskussion der Lokalisierung durch Verwendung neuronaler Netze	86
5.3.1	Eigene Implementierung des Modells	87
5.3.2	Training des Modells	87
5.3.3	Vergleich mit DeepDarts	87
6	Fazit	89
6.1	Beantwortung der Forschungsfragen	89
6.2	Zusammenfassung des Systems	91
6.3	Ausblick	92
6.3.1	Ausblick der Datenerstellung	92
6.3.2	Ausblick der Normalisierung	92
6.3.3	Ausblick der Dartpfeilerkennung	93

List of todos

1 Abkürzungen sehen nicht gut aus, aber das zu fixen ist nervig.	101
--	-----

Kapitel 1

Einleitung

Darts ist ein beliebtes Spiel mit vielerlei Spielvariationen und geringer Einstiegsschwelle für neue Spieler. Mit zunehmender Beliebtheit von Steeldarts als Freizeitbeschäftigung steigt auch die Relevanz automatisierter Systeme für das Scoring. Diese Aufgabe muss auf eine Weise gelöst werden, die sowohl benutzerfreundlich wie auch effizient ist. Insbesondere im Bereich professioneller Anwendung ist die Zuverlässigkeit und Genauigkeit dieser Systeme unabdingbar. Vor diesem Hintergrund befasst sich die vorliegende Arbeit der Konzeption und Umsetzung eines Systems, das auf Basis eines einzelnen Bildes einer Dartscheibe sowohl die getroffenen Felder als auch die exakten Positionen der Dartpfeile identifiziert. Zur Umsetzung werden realistische Simulationen, algorithmische Bildverarbeitung und maschinelles Lernen kombiniert, um eine robuste und effiziente Lösung zu entwickeln.

1.1 Projektübersicht

Während für den Heimgebrauch sowie für Gaststätten elektronische Dartscheiben mit eingespeicherten Spielvarianten und automatischem Scoring existieren, werden in professionellen Kreisen weiterhin analoge Dartscheiben ohne integrierte Mechanismen verwendet, die ein automatisches Scoring ermöglichen. Das Ermitteln der erzielten Punktzahlen kann im Steeldarts entweder manuell oder automatisch geschehen. Manuelles Errechnen der erzielten Punktzahlen erfordert Konzentration und Übung, um die finale Punktzahl korrekt und schnell zu berechnen:

$$\text{Score} = \sum_{i=1}^3 \text{mult}_i \cdot \text{Feld}_i$$

Automatisierte Techniken zur Bestimmung der Punktzahl unterscheiden sich in ihren Herangehensweisen. Die zuverlässigste und in professionellen Kreisen am weitesten verbreitete Herangehensweise ist der Einsatz von Multi-Camera-Systemen. In diesem werden mehrere kalibrierte Kameras um die Dartscheibe platziert und aufeinander abgestimmt, sodass eine akkurate Rekonstruktion der Dartscheibe möglich ist. Durch diese Rekonstruktion ist eine Triangulation der Dartpfeilpositionen und folglich eine Bestimmung des Scorings ermöglicht. Diese Systeme sind gewerblich erhältlich, jedoch ist ihr Einsatz für den gelegentlichen Heimgebrauch nicht im Einklang mit ihren Preisen. Beispiele etablierter Systeme sind AutoDarts [2] und Scolia [3].

In Abschnitt 1.7 werden weitere Herangehensweisen des automatisierten Dart-Scorings aufgelistet, jedoch setzen diese Systeme allesamt spezielle Infrastruktur oder gesonderte Kalibrierung voraus. Ziel dieser Masterarbeit ist die Erarbeitung eines Systems, in welchem ein zuverlässiges Dart-Scoring ohne spezielle Hardware und ohne feine Kalibrierung ermöglicht wird. Dazu werden Techniken der herkömmlichen CV und neuronale Netze in einer Art und Weise miteinander kombiniert, die im Einklang mit der Ausführung auf mobilen Endgeräten ist.

Dieses System basiert auf Aufnahmen von Mobiltelefonen, in denen Dartscheiben abgebildet sind. In diesen Aufnahmen wird die Dartscheibe in einem ersten Verarbeitungsschritt algorithmisch identifiziert und es wird eine Entzerrung des Bildes durchgeführt, die die Dartscheibe normalisiert und in ihre runde Grundform transformiert. Diese normalisierten

Bilddaten werden in einem zweiten Schritt durch ein neuronales Netz verarbeitet, welches die Spitzen der Dartpfeile sowie die Feldfarbe durch Klassifizierung identifiziert und durch eine Regression spezifisch lokalisiert. Durch die ermittelten Positionen in dem normalisierten Bild und die zusätzlichen Informationen der jeweiligen Feldfarben ist eine Zuordnung der getroffenen Felder und folglich eine Ermittlung der erzielten Punktzahl trotz etwaiger Ungenauigkeiten der Normalisierung möglich.

Durch die Verwendung eines neuronalen Netzes besteht eine Notwendigkeit einer ausreichenden Anzahl korrekt annotierter Trainingsdaten. Diese Daten werden durch ein ebenfalls in dieser Arbeit enthaltenes System der Datengenerierung bezogen, in welchem realistische Aufnahmen von Dartscheiben simuliert werden. Die Generierung der Trainingsdaten basiert auf einer Kombination aus prozeduraler Datenerstellung und Zufallsprozessen, durch die eine beliebige Anzahl unterschiedlicher Daten erstellt werden kann.

1.2 DeepDarts

Der Anstoß dieses Projekts ist durch ein Paper von McNally et al. gegeben, in dem ein System zur Identifizierung von Dartpfeilen in Bildern mit anschließendem Scoring vorgestellt wurde [1]. Das unter dem Namen DeepDarts vorgestellte System ist ebenfalls ein Single-Camera-System, welches auf der Verwendung eines neuronalen Netzes fußt, um ein Dart-Scoring umzusetzen.

DeepDarts verwendet ein YOLOv4-tiny-Netzwerk, welches auf einem Datensatz, bestehend aus etwa 16.000 händisch annotierten Bildern, trainiert ist [4]. Das System ist durch die Implementierung einer eigenen Loss-Funktion zur Beurteilung der Vorhersagen umgesetzt und konnte auf den gegebenen Daten teils sehr gute Ergebnisse erzielen.

Die Arbeitsweise von DeepDarts verfolgt einen Single-Shot-Approach, indem in einem Eingabebild bis zu 7 Punkte identifiziert werden, die sich aufteilen in 4 Orientierungspunkte und bis zu 3 Dartpfeilspitzen. Die Orientierungspunkte sind festgelegte Positionen auf der Dartscheibe, die die Approximation der Dartscheiben-Geometrie und folglich eine Entzerrung dieser ermöglichen. Durch die relativen Positionen der Dartpfeilspitzen zu den Orientierungspunkten werden die getroffenen Felder anhand von nahezu standardisierten Werten der Dartscheibengeometrie abgeleitet.

Die Qualität der Vorhersagen der erzielten Punktzahl ist durch eine eigene Metrik, dem Percent Correct Score (PCS), ausgewertet. Dieser setzt die Anzahl der korrekt vorhergesagten Punktzahlen in Verhältnis zur Anzahl aller präsentierter Daten:

$$\text{PCS} = \frac{100}{N} \sum_{i=1}^N \delta \left(\left(\sum \hat{S}_i - \sum S_i \right) = 0 \right) \%$$

In dieser Formel steht N für die Anzahl der vorhergesagten Daten, $\sum S_i$ ist die erzielte Punktzahl und $\sum \hat{S}_i$ die vorhergesagte Punktzahl des Datensamples i . Die simple Aussage hinter dieser kompliziert ausgedrückten Formel ist, dass die Anzahl korrekt vorhergesagter Gesamtpunktzahlen ins Verhältnis zu der Gesamtzahl der vorhergesagten Daten gesetzt wird.

Das System ist mit unterschiedlichen Aufteilungen der gegebenen Daten trainiert. Zusammenfassend konnte DeepDarts auf Validierungs- und Testdaten Auswertungen mit PCS von etwa 84 % bis 95 % erzielen. Als zentrale Schwachstelle des Systems ist das Identifizieren der Orientierungspunkte aufgelistet, welches für fehlerhafte Erkennungen sorgte.

Bei genauerer Betrachtung und Inferenz des Systems auf eigenen Daten kristallisierte sich jedoch ein anderes Bild der tatsächlichen Performance heraus. Das System konnte auf Bildern außerhalb der für DeepDarts vorliegenden IEEE-Daten wenig Erfolge verzeichnen, wodurch ein starkes Overfitting des Netzwerks naheliegt. Die zum Training verwendete Datenlage war aufgrund der verwendeten Aufnahmetechniken stark limitiert. Etwa 14.000 der 16.000 Bilddaten sind frontal aufgenommen und zeigen die selbe Ausrichtung der selben Dartscheibe. Die restlichen 2.000 Daten sind mit einer zweiten Dartscheibe und zu Teilen aus unterschiedlichen Winkeln aufgenommen. Zudem wurde die Korrektheit der manuell annotierten Daten in Frage gestellt. Nach einer Auswertung von DeepDarts wurde für 1.200 ausgewählte Daten eine Deckungsgleichheit der notierten und annotierten Punktzahlen von 97,6 % ermittelt; 29 Bilder sind folglich nicht korrekt annotiert.

Durch diese Datenlage ist keine Abdeckung zu erwartender Aufnahmen in einem realen Einsatz des Systems dar, was bei der Inferenz auf eigenen Aufnahmen zu erkennen war. DeepDarts leidet unter starkem Overfitting und ist nicht in der Lage, zuverlässig auf unabhängigen Daten zu generalisieren. Diese Beobachtung ist eine zentrale Erkenntnis und hat die Form dieser Arbeit stark geprägt.

Ziel dieser Masterarbeit ist es, die identifizierten Schwachpunkte sowie die eingesetzten Techniken von DeepDarts in einem neuen Ansatz zusammenzuführen und auf den gewonnenen Erkenntnissen aufzubauen. Der Aufbau dieser Arbeit ergibt sich aus den zentralen Fehlerquellen von DeepDarts.

1.3 Einsatz synthetischer Datenerstellung

Eine wichtige Erkenntnis des DeepDarts-Systems ist der Mangel qualitativ hochwertiger Trainingsdaten. Die manuelle Aufnahme und Annotation von Daten ist sowohl zeitaufwändig wie fehleranfällig. Fehlerhafte Annotationen in den Daten werden von dem Netzwerk während des Trainings erlernt und beeinträchtigen die Qualität der Inferenz. Zudem zeichnet sich ein robustes Training durch eine möglichst uniforme Abdeckung aller zu erwartender Eingabedaten aus. In Hinsicht auf Bilder von Dartscheiben beinhaltet dies die Einbindung einer Vielzahl unterschiedlicher Dartscheiben sowie Umgebungsbedingungen. Eine reale Umsetzung dieser Anforderungen in Kombination mit einer ausreichenden Anzahl an Daten für ein geeignetes Training würde sehr viel Zeit in Anspruch nehmen, was weder dem Umfang noch dem Schwerpunkt dieser Arbeit entspricht.

Stattdessen wird die synthetische Erstellung von Trainingsdaten unter Verwendung von 3D-Modellierungssoftware verwendet. Auf diese Weise ist eine automatisierte Datenerstellung ermöglicht, die durch den Einsatz zeitgemäßer Technik prozedural und fotorealistisch erstellt werden kann. Durch einen anfänglichen Mehraufwand in der Aufsetzung dieses Systems können anschließend beliebige Datenmengen erstellt werden, ohne die Notwendigkeit manueller Eingriffe. Zusätzlich sind alle relevanten Informationen der Szene zugänglich, sodass eine korrekte Annotation ermöglicht ist.

1.4 Einsatz herkömmlicher Computer Vision

Die Erkennung der Dartscheibe geschieht in DeepDarts durch das selbe neuronale Netz, welches zugleich die Dartpfeile identifiziert. Das neuronale Netz ist darauf trainiert, spezifische Orientierungspunkte entlang der Außenseite der Dartscheibe zu identifizieren, anhand derer die Dartscheibe entzerrt wird. Während diese Herangehensweise für gute Ergebnisse in positiven Fällen gesorgt hat, ist sie an einfacher Verdeckung eben dieser spezifischen Positionen gescheitert. Diese Verdeckungen können sowohl durch externe Objekte wie einen Dartschrank als auch durch die Dartpfeile selbst entstehen. Sind die Punkte nicht eindeutig zu erkennen, werden sie nicht lokalisiert und die Entzerrung sowie alle nachfolgenden Identifizierungen schlagen fehl. Zusätzlich kann sich eine verschobene Erkennung der Orientierungspunkte negativ auf die Genauigkeit der Vorhersage auswirken.

Zur Lösung dieses Problems wird von McNally et al. vorgeschlagen, Redundanz durch das Einbinden weiterer Orientierungspunkte zu schaffen. Dieser Ansatz wird in dieser Masterarbeit verfolgt, jedoch nicht unter der Verwendung eines neuronalen Netzes. Ein zentrales Problem neuronaler Netze ist das Training, welches viele Ressourcen beansprucht und dessen Erfolg maßgeblich von den Trainingsdaten abhängt. Ihre Arbeitsweisen basieren auf den ihnen präsentierten Trainingsdaten und sind weder bekannt noch nachvollziehbar. Hintergründe fehlerhafter Erkennungen können daher nicht eingesehen werden und eine Adaption des Systems auf neue Gegebenheiten kann ausschließlich unter Verwendung einer ausreichenden Anzahl korrekt annotierter Daten geschehen.

Aus diesen Gründen wird für diese Thesis einen Schritt zurückgegangen und die Verwendung eines neuronalen Netzes für diese Aufgabe ausgeschlossen. Dartscheiben verbindet ein gemeinsamer Grundaufbau, der durch Richtlinien offizieller Regelwerke mit etwaigen Toleranzen festgelegt ist [5, 6]. Dieser Aufbau der Dartscheiben ist begünstigend für eine Verarbeitung mit herkömmlichen Techniken der CV. Kontrastreiche Farbgebung und festgelegte, markante Geometrien sind ideale Merkmale, an denen CV-Algorithmen ansetzen.

Der Ablauf der Entzerrung der Dartscheiben läuft in mehreren Schritten ab. Zuerst wird die Position einer Dartscheibe in einem Bild bestimmt. Diese unterliegt o. B. d. A. einer perspektivischen Verzerrung, die in den folgenden Schritten entzerrt wird. Dazu werden die Winkel der Felder radial um den Mittelpunkt bestimmt und ausgeglichen, sodass alle Winkel gleichwertig sind. Abschließend wird eine Vielzahl an Orientierungspunkten identifiziert, anhand derer eine perspektivische Entzerrung vorgenommen wird, die zu einer Normalisierung der Dartscheibe führt.

Durch diesen Algorithmus ist die Identifizierung sowie Entzerrung von Dartscheiben in Bildern beliebiger Größen möglich. Da dieser Algorithmus weitestgehend deterministisch agiert und die Arbeitsweise zu jeder Zeit transparent einsehbar ist, ist das Identifizieren von spezifischen Fehlerquellen und eine Adaption der Arbeitsweise mit weitaus weniger Aufwand verbunden als das Trainieren eines neuronalen Netzes. Zudem können neue Datenlagen anhand einer geringen Anzahl von Beispieldildern in den Algorithmus aufgenommen werden, beispielsweise die Integration neuer Farbschemata von Dartscheiben.

Der Erläuterung dieses Algorithmus ist Kapitel 3 zugewiesen. In diesem werden Grundlagen der CV, die Methodik und Hintergründe zu dem Algorithmus sowie Details spezifischer Implementierungen dargestellt. Die Ergebnisse dieses Algorithmus werden abschließend ebenfalls dargestellt und erläutert.

1.5 Einsatz neuronaler Netze

Trotz zuvor aufgezählter Schwachpunkte neuronaler Netze sind sie algorithmischen Techniken in vielerlei Hinsicht überlegen. In dieser Arbeit werden neuronale Netze zur Lokalisierung von Dartpfeilspitzen in normalisierten Bildern genutzt. Das Ziel dieses Teils der Thesis ist die korrekte Lokalisierung von Dartpfeilen in Bildern zur Ermittlung der erzielten Punktzahl nach einer Dartsrunde.

Die von McNally et al. geleistete Vorarbeit durch DeepDarts hat bereits die Fähigkeit neuronaler Netze gezeigt, diese Aufgabe zu bewältigen. Obwohl durch starkes Overfitting keine Generalisierbarkeit des Systems gezeigt werden konnte, ist die Möglichkeit der Bewältigung dieser Aufgabe ermittelt worden. In dieser Arbeit gilt es daher, auf den gewonnenen Erkenntnissen aufzubauen und ein System zu entwickeln, welches in der Lage ist, auf unbekannten Daten angewandt zu werden.

Es wird daher der Ansatz verfolgt, ein neuronales Netz basierend auf der selben Modell-Familie zu verwenden, wie es in DeepDarts genutzt wird. Seit der Veröffentlichung von DeepDarts in 2021 sind neue Architekturen der YOLO-Familie vorgestellt, deren Performances über der der verwendeten Architektur liegen. Diese Architekturen werden als Grundgerüst genutzt, um durch Adaptionen ihres Aufbaus ein auf diese Aufgabe zugeschnittenes neuronales Netz zu etablieren.

Eine wesentliche Änderung ist eine Veränderung der Netzwerkausgaben. DeepDarts verwendet die herkömmlichen Ausgaben der Modellarchitektur, in der Existenzen, Positionen mit Bounding Boxen und Klassen ausgegeben werden. Die Bounding Boxen werden in DeepDarts lediglich im Training verwendet, jedoch nicht bei der Inferenz. In DeepDarts werden fünf Klassen vorhergesagt, indem jeder der vier Orientierungspunkte eine eigene Klasse besitzt und eine weitere Klasse für Dartpfeilspitzen verwendet wird. Für diese Arbeit sind die Ausgaben derart angepasst, dass keine Bounding Boxes vorhergesagt werden und die vorhergesagten Klassen die Farben der getroffenen Felder abbilden.

Diese Adaptionen der Ausgaben sorgen dafür, dass keine irrelevanten Informationen vorhergesagt werden und eine robustere Rückrechnung der Punktzahl ermöglicht ist. Wohingegen die Punktzahl bei DeepDarts durch relative Positionen zu den Orientierungspunkten ermittelt wird, wird in dieser Arbeit ein System verwendet, welches Positionen und Feldfarben auf einer normalisierten Dartscheibe kombiniert, um eine Punktzahl zu ermitteln. Der wesentliche Vorteil dieser Herangehensweise liegt in der Robustheit gegenüber Verschiebungen in der Normalisierung: Wird ein Dartpfeil an der Grenze zweier Felder erkannt, ist die Vorhersage trotz ungenauer Normalisierung möglich, da eine Ermittlung des Feldes durch die identifizierte Feldfarbe unterstützt wird.

1.6 Forschungsfragen

In dieser Masterarbeit werden mehrere Systeme erarbeitet, die sich zu einem Gesamtkonzept zusammensetzen. Die zu behandelnden Forschungsfragen ergeben sich aus den jeweiligen Teilprojekten dieser Arbeit. Aus diesem Grund wird für jeden Teil dieser Thesis eine Forschungsfrage aufgestellt und das Zusammenspiel aller Bestandteile in einer abschließenden Forschungsfrage kombiniert.

1. Welche Qualität synthetischer, realistischer und variabler Daten können in einer automatisierten Pipeline zur Erstellung von Bildern von Dartscheiben mit korrekter Annotation erreicht werden?

Diese Fragestellung bezieht sich inhaltlich auf die Ausarbeitung der synthetischen Datenerstellung, welche den ersten Teil dieser Arbeit ausmacht. Sie umfasst in erster Instanz die Umsetzbarkeit der automatisierten Erstellung von Bildern mit Dartscheiben. Anschließend ist diese Datenerstellung mit einer Randomisierung von Parametern verbunden, einschließlich Annotation der Daten hinsichtlich der Position der Dartpfeilspitzen in den Bildern. Zuletzt gilt es, diese Datenerstellung in eine automatisierte Pipeline einzubinden, mit der eine autonome Erstellung einer Vielzahl unterschiedlicher Daten ermöglicht wird.

Diese Daten unterliegen zusätzlich der Bedingung, ein möglichst realistisches Erscheinungsbild aufzuweisen und eine große Spanne unterschiedlicher Aussehen und Beleuchtungen abzudecken, sodass die Wahrscheinlichkeit der Datenverzerrung durch einseitige Darstellung minimiert wird. Die Qualität bezüglich des Grades der Realitätsnähe quantitativ einzuordnen ist nicht trivial und wird daher durch qualitative Vergleiche mit realen Daten ermittelt.

2. Zu welchem Grad lässt sich eine zuverlässige algorithmische Erkennung und Normalisierung von Dartscheiben in Bildern ohne den Einsatz neuronaler Netze umsetzen?

Ziel dieser Forschungsfrage ist die Untersuchung des zweiten Themenbereichs dieser Arbeit. Die algorithmische Identifizierung und Normalisierung hinsichtlich der Dartscheibengeometrie ist in dieser Arbeit als Vorverarbeitungsschritt eingebunden und geschieht ohne den Einsatz von Systemen, deren Arbeitsweise auf Daten basierend automatisiert erlernt werden. Jegliche Arbeitsschritte und Parameter des Systems unterliegen der sorgfältigen Analyse zu erwartender Daten.

Zur Beantwortung dieser Forschungsfrage wird eine Analyse auf Daten unterschiedlicher Quellen durchgeführt, die Einblicke in die Arbeitsweise und Vielseitigkeit des erarbeiteten Systems liefern. Die Auswertung geschieht durch Metriken zur Beurteilung der Erfolgsrate und Genauigkeit des Algorithmus.

3. Wie zuverlässig ist eine Generalisierung eines durch out-of-distribution (OOD)-Training mit synthetischen Daten trainiertes neuronales Netzwerk auf Daten realer Dartscheiben?

Die im ersten Teil dieser Thesis erstellten Daten werden für das Training eines neuronalen Netzes eingesetzt. Durch die Verwendung synthetischer Daten als Datenlage sind Unterschiede zwischen Trainings- und Inferenzdaten zu erwarten. Der Grad des Unterschieds unterliegt der Qualität der Datenerstellung, kann jedoch als nicht unerheblich eingestuft werden. Diese Forschungsfrage untersucht die Fähigkeit eines neuronalen Netzes, zuverlässige Vorhersagen auf realen Daten trotz dieser Differenzen zu erbringen. Zur Beantwortung dieser Forschungsfrage wird die Inferenz auf unterschiedlichen Datensätzen protokolliert, die sowohl synthetische als auch reale Daten umfasst. Auf diese Weise wird die Fähigkeit der Generalisierung des Netzes auf synthetischen Daten im Vergleich mit realen Daten ins Verhältnis gesetzt, um einen relativen Unterschied aufzeigen zu können.

4. Ist das in dieser Thesis erarbeitete Gesamtsystem in der Lage, signifikante Verbesserungen hinsichtlich der Performance und Genauigkeit im Vergleich zu DeepDarts zu erzielen?

Anstoß dieser Thesis ist die Arbeit von McNally et al., in der ein System mit dem Namen DeepDarts vorgestellt ist [1]. Dieses System kann sehr gute Ergebnisse auf eigenen Daten erzielen und gilt damit als Maßstab für Genauigkeit und Geschwindigkeit. Die Konfigurationen und Gewichte, die für die Auswertung von DeepDarts verwendet sind, werden in dieser Arbeit ebenfalls genutzt, um Vergleichswerte zu ermitteln. Da dieses System als Erweiterung von DeepDarts vorgesehen ist, ist ein Übertreffen des Systems hinsichtlich unterschiedlicher Metriken das zentrale Ziel dieser Arbeit, auf das alle vorgestellten Systeme dieser Thesis ausgelegt sind.

1.7 Themenbezogene Arbeiten

Diese Arbeit weist mehrere Schwerpunkte auf, hinsichtlich derer verwandte Arbeiten betrachtet werden können. In den folgenden Unterabschnitten werden unterschiedliche Bereiche beleuchtet, in denen verwandte Arbeiten veröffentlicht sind.

1.7.1 Prozedurale Datenerstellung

Das prozedurale Erstellen von Daten für eine vielfältige Auswahl unterschiedlicher Szenarien ist kein Themengebiet, dessen spezifischer Einsatz im Bereich des Trainings neuronaler Netze verankert ist. Im Fachgebiet der Spieleindustrie wird diese Technik verwendet, gesteuerten Zufall in das Spielerlebnis einfließen zu lassen [7, 8, 9]. Die Brücke zu dieser Arbeit wird dabei durch die zufällige Erstellung von Welten geschlagen, die analog zu der Randomisierung der Umgebung und dem Aussehen der Dartscheibe zu sehen sind, wie prozedurale Datenerstellung in dieser Arbeit verwendet wird. Diese Umsetzungen basieren auf dem gleichen Konzept, jedoch ist die Art der Umsetzung unterschiedlich gehandhabt.

1.7.2 Synthetische Datenerstellung für das Training neuronaler Netze

Die Vorgehensweise synthetischer Datenerstellung zur Generierung von Trainingsdaten wird in unterschiedlichen Systemen eingesetzt [10]. Die Art der synthetisch erstellten Daten umfasst dabei eine große Spanne unterschiedlicher Szenarien. So konnte bereits die Effektivität der Nutzung von 3D-Modellen in der Datengenerierung erfolgreich eingesetzt werden [11, 12, 13]. Mit der Verwendung von 3D-Software zur Generierung von Trainingsdaten wurden ebenfalls positive Ergebnisse erzeugt [14, 15, 16]. Obwohl Unterschiede zwischen generierten Daten und realen Daten bestehen, konnte gezeigt werden, dass selbst durch reines OOD-Training Systeme trainiert werden konnten, die auf reale Daten generalisieren konnten [17].

1.7.3 Dart-Scoring

Neben bereits etablierten Multi-Camera-Systemen wie Autodarts und Scolia [2, 3] existieren weitere Herangehensweisen an diese Aufgabe. Autodarts und Scolia verwenden jeweils 3 Kameras, um eine 3D-Rekonstruktion der Dartscheibe und den auf ihr eingetroffenen Pfeilen zu ermöglichen. Diese Systeme sind jedoch kostspielig und nicht für Amateurnutzung vorgesehen. Um diese Hürden zu umgehen, wurde eine Arbeit um einen Algorithmus zur Nutzung günstiger Kameras für ein kostengünstiges und ebenfalls akkurate System mit fünf Kameras veröffentlicht [18].

Die Idee des einfacher zugänglichen und erschwinglichen Dart-Scorings hat in Hobbyprojekten Anklang gefunden, sodass Systeme mit unterschiedlichen Herangehensweisen der Ermittlung des Scorings entstanden sind. Diese Systeme reichen von der Verwendung von Mikrofonen zur akustischen Triangulation der Einstichstellen [19] über die Verwendung von fünf kostengünstigen Kameras in Kombination mit ausgeklügelter Kalibrierung [18], stereoskopische Rekonstruktion der Dartscheibe mittels zweier Kameras [20, 21] bis hin zu Single-Camera-Systemen [22, 23]. Für alle diese Systeme ist jedoch mindestens einer der

folgenden Schwachpunkte zutreffend: Komplexes bzw. sehr genaues Setup, Notwendigkeit besonderer Hardware oder mangelhafte Genauigkeit des Systems. Basieren die Systeme auf Differenzbildern, ist eine Verschiebung der Kamera verheerend; ebenso bei der Verwendung von Kameras, deren Positionen zueinander kalibriert werden müssen.

Der Schwachpunkt eines vorherigen Aufbaus oder der Kalibrierung ist durch ein System, wie es mit DeepDarts erhoben wurde, hinfällig, indem ein Scoring auf Grundlage eines beliebigen Kameramodells und ohne weitreichende Ansprüche an die benötigte Infrastruktur als Zielsetzung verfolgt wurde [1]. Die Verwendung neuronaler Netze für die Verarbeitung von Bilddaten ist dabei ein neuer Schritt der Einbindung zeitgemäßer Technologien in dieses Forschungsfeld. Dieser Ansatz wird mit dieser Arbeit weiter verfolgt mit der Zielsetzung, neue Technologien in dem Bereich des Dart-Scorings einzusetzen, um ein in der Bedienung simples Dart-Scoring mit geringen Anforderungen an den Benutzer zu erzielen.

1.8 Aufbau der Arbeit

Diese Masterarbeit befasst sich mit den Themenbereichen der Datengenerierung, Bildnormalisierung und dem Training eines neuronalen Netzes. Das Zusammenspiel dieser Komponenten ist in Abbildung 1.1 veranschaulicht. Zunächst wird in Kapitel 2 auf die synthetische Datengenerierung eingegangen. Anschließend wird in Kapitel 3 ein Algorithmus vorgestellt, der eine Normalisierung von Bildern mit Dartscheiben ermöglicht. Kapitel 4 rundet den thematischen Schwerpunkt dieser Masterarbeit mit der Konzeption und dem Training eines neuronalen Netzes zur Identifizierung und Lokalisierung von Dartpfeilspitzen in normalisierten Bildern ab. Diese Kapitel sind jeweils unterteilt in die Abschnitte Grundlagen, Methodik, Implementierung und Ergebnisse. In den Abschnitten „Grundlagen“ werden Themen und Konzepte eingeführt, die für das Verständnis des Kapitels notwendig sind und nicht im erwarteten Wissensgebiet der Leserschaft liegen. In den Abschnitten „Methodik“ werden die Konzepte und Herangehensweisen der Kapitel dargestellt. Auf Details zur Umsetzung dieser Methodiken werden in den jeweiligen Abschnitten „Implementierung“ eingegangen. Den Abschluss aller Kapitel bilden die Darstellung und Auswertung der erzielten Ergebnisse. Im Anschluss an die themenbezogenen Kapitel folgt eine zusammenfassende Diskussion der Ergebnisse in Kapitel 5. Danach wird in Kapitel 6 das Fazit dieser Masterarbeit gezogen, indem die Forschungsfragen erneut aufgegriffen werden und mit einem Ausblick über mögliche Erweiterungen und Verbesserungen des Systems abgeschlossen wird.

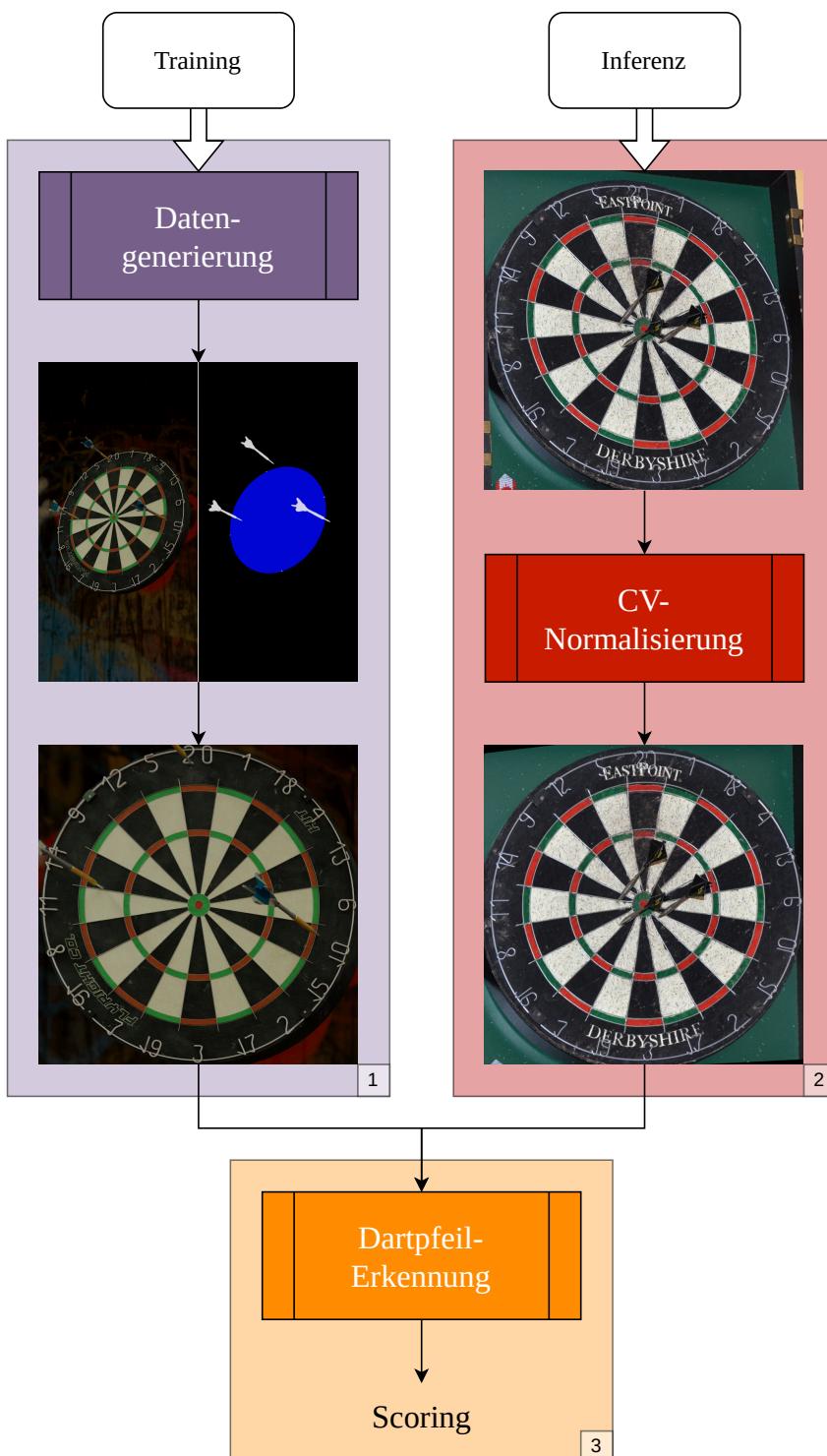


Abbildung 1.1: Überblick über die Projektstruktur. (1) Datenerstellungs-Pipeline; bei der Datengenerierung werden Bilder, Masken und Annotationen erstellt und automatisch normalisiert. (2) Inferenz-Pipeline; beliebige Bilder von Dartscheiben werden algorithmisch normalisiert. (3) Dartpfeil-Erkennung und Scoring; die Erkennung geschieht durch ein neuronales Netz, das Scoring durch Nachverarbeitung der Outputs.

Kapitel 2

Simulation von Dartscheiben zur Generierung von Trainingsdaten

Der erste Themenbereich dieser Masterarbeit ist die Erstellung von Daten. Diese Daten bestehen aus Bildern und Annotationen von Objekten in diesen Bildern. Bei den Bildern handelt es sich um Aufnahmen von Dartscheiben, in denen Dartpfeile stecken, deren Positionen und die mit diesen erzielten Punktzahlen die Annotationen ausmachen. In diesem Kapitel wird ein System vorgestellt, mit welchem die automatisierte Erstellung dieser Daten ermöglicht wird.

Das Aussehen dieser Bilder zielt darauf ab, Aufnahmen von Mobiltelefonen zu imitieren, wie sie im Einsatz in einer App zum automatischen Dart-Scoring anhand einzelner Bilder aufgenommen werden. Insbesondere bezieht diese Zielsetzung die charakteristischen Optiken von Kameras in Mobiltelefonen mit ein, beispielsweise erhöhte Nachverarbeitung durch Kontrasterhöhung oder Rauschen durch hohe ISO-Werte. Die Verteilung der Dartpfeile auf der Dartscheibe folgt typischen Mustern, die in Amateurspielen zu erwarten sind.

Umgesetzt wird die Datenerstellung durch das Modellieren einer 3D-Szene bestehend aus Dartscheibe, Dartpfeilen, verschiedenen Beleuchtungen und einer Kamera. Diese Objekte sind auf eine Weise konzipiert, dass die Erscheinungsbilder dieser Objekte prozedural gestaltet sind und eine beliebige Anzahl verschiedener Objekte von diesen abgeleitet werden kann. Dadurch wird eine große Variabilität der erstellten Daten ermöglicht.

Dieses Kapitels ist unterteilt in vier Unterkapitel. In dem ersten Unterkapitel werden Grundlagen hinsichtlich der Datenerstellung und die in ihr verwendeten Techniken vorgestellt. Diese bieten die Grundlage zum Verständnis der Methodik, welche im darauf folgenden Unterkapitel erläutert wird. Die Methodik umfasst die Konzepte, die zur Erstellung der Daten verwendet werden und auf denen sich die Implementierung stützt. Auf bestimmte Bereiche der Implementierung wird in dem dritten Unterkapitel eingegangen. Abschließend werden die Ergebnisse der synthetischen Datenerstellung im letzten Unterkapitel betrachtet und ausgewertet.

2.1 Grundlagen

Bevor in die Methodik der Datenerstellung eingestiegen werden kann, ist die Klärung von Grundlagen notwendig. Diese ermöglichen ein Verständnis der Thematik und der verwendeten Methodik.

Die Grundlagen der Datenerstellung sind thematisch geordnet in die Abschnitte der Datenerstellung, Darts und Texturierung. Die generelle Datenerstellung umfasst prozedurale Datenerstellung und Grundlagen des 3D-Renderings sowie Kameraparameter und Binärbilder und Masken. Der Darts-Abschnitt beginnt mit der Geometrie einer Dartscheibe, gefolgt von für diese Thesis relevante Darts-Terminologie. Danach folgt die Unterscheidung zwischen Soft- und Steeldarts. Zuletzt folgt der Abschnitt der Texturierung, in dem mit der Klärung der Funktionsweise von Material und Texturen eingestiegen wird. Darauf folgt eine Übersicht über Noise-Texturen und Seeding von Zufallsvariablen. Abschließend werden die Begriffe Thresholding und Maskierung und die Hintergründe prozeduraler Texturen erläutert.

2.1.1 Prozedurale Datenerstellung

Prozedurale Datenerstellung kommt in einer Vielzahl unterschiedlicher Anwendungsbereiche zum Einsatz. In dieser Arbeit dient die prozedurale Datenerstellung der Generierung von Trainingsdaten für neuronale Netze. Dieser Unterabschnitt erläutert, was unter prozeduraler Datenerstellung zu verstehen ist und welche Vorteile sich aus ihrer Anwendung ergeben.

Die Erhebung von Daten kann sowohl manuell als auch automatisch geschehen. Prozedurale Datenerstellung bezeichnet eine Methode zur automatisierten Generierung von Daten unter Verwendung zufälliger Werte. Zur Datengenerierung werden zunächst Rahmenbedingungen definiert, innerhalb derer Zufallswerte erzeugt werden, die wiederum zur Steuerung relevanter Parameter dienen. Die Art der Zufallswerte ist dabei je nach Einsatzbereich unterschiedlich; für diese Thesis finden hauptsächlich ein- und zweidimensionale Zufallswerte ihren Einsatz. Eindimensionale Zufallswerte werden beispielsweise genutzt, um Parameter wie Sättigung oder Helligkeit von Texturfarben zu bestimmen. Zweidimensionale Zufallswerte finden Anwendung in Form von Rauschtexturen, die beispielsweise die Oberflächenbeschaffenheit von Objekten beeinflussen. Die Wertebereiche dieser Zufallsgrößen lassen sich durch Skalierung, Clipping oder andere Verfahren der Nachverarbeitung an den jeweiligen Einsatzzweck anpassen. Durch diese Art der Parametrisierung durch Zufallswerte ist das automatisierte Erstellen einer Vielzahl variierender Daten möglich.

Ein wesentlicher Vorteil der prozeduralen Datenerstellung liegt in der hohen Komplexität der erzeugten Daten, die durch die Kombination zahlreicher Zufallsparameter erreicht werden kann. Die für ein Datensample verwendeten Zufallsparameter definieren dessen Position im hochdimensionalen Parameterraum. Durch diese Positionierung ist der Aufbau eines Samples eindeutig beschrieben und eine deterministische Reproduktion dessen ist durch diese ermöglicht.

2.1.2 3D-Rendering

Für die Erstellung von fotorealistischen Bildern aus 3D-Szenen wird eine Technik verwendet, die als Raytracing bezeichnet wird. Beim Raytracing wird die Farbe von Pixeln einer Kamerafläche durch von dieser ausgehenden Strahlen bestimmt [24, 25, 26]. Treffen diese Strahlen auf Objekte, werden sie in Abhängigkeit von dessen Oberflächenbeschaffenheit modifiziert. Auf diese Weise geschieht eine rekursive Aussendung weiterer Strahlen, bis ein Abbruchkriterium erreicht ist. Durch diese von der Kamera ausgesandten Strahlen wird die Farbgebung der entsprechenden Pixel bestimmt, indem Strahlen durch Auftreffen auf Materialien und Lichtquellen Farbinformationen sammeln. Die verteilte Anwendung dieser Technik für alle Pixel der Kamera sorgt für eine realistische Ausleuchtung einer modellierten Szene.

Raytracing ist der Industriestandard für die Erstellung realistischer Bilder und findet seinen Einsatz beispielsweise in Videospielen oder bei der Generierung realistischer Aufnahmen sowohl für Momentaufnahmen als auch für Bewegtbilder. In dieser Arbeit wird Raytracing zur Generierung realistischer Bilder von Dartscheiben verwendet, indem eine 3D-Szene erstellt wird, aus welcher Bilder erstellt werden. Diese Bilder werden im Kontext dieser Arbeit als Grundlage des Trainings eines neuronalen Netzes verwendet.

2.1.3 Kameraparameter

Bei der Datenerstellung mittels 3D-Software und Raytracing ist die Präsenz einer Kamera unabdingbar. Erste Modellierungen von Kameramodellen begannen mit einer Pinhole-Kamera, die eine Projektion des 3D-Raumes in den 2D-Raum vornimmt [27]. Auf diesem Modell aufbauend wurden weitere Kameraparameter modelliert, bis eine vollständige Simulation von Kameras ermöglicht wurde. In aktuellen 3D-Softwares zum Rendern von Szenen sind eine Vielzahl an Kameraparametern implementiert und modifizierbar, sodass fotorealistische Aufnahmen simuliert werden können. Die Unterschiede zwischen der Verwendung einer simulierten und einer realen Kamera sind daher für das ungeschulte Auge verschwindend gering und für das Erstellen von Trainingsdaten für ein neuronales Netz ideal geeignet. Die wichtigsten Parameter einer Kameraaufnahme werden in diesem Unterabschnitt grob erläutert, um ein grundlegendes Verständnis der Arbeitsweise einer Kamera zu erlangen.

Brennweite und Öffnungswinkel Die Brennweite einer Kamera bzw. eines Objektives bestimmt die Lichtbrechung bei der Aufnahme eines Bildes. Diese Lichtbrechung resultiert in einem unterschiedlich großen Bereich, der von der Kamera eingefangen wird. Optisch ist die Brennweite für den Zoom des Bildes zuständig. Eine Brennweite von 50 mm ist eine typische Brennweite, die dem menschlichen Blickwinkel nahe kommt. Geringere Brennweiten sorgen für ein größeres Sichtfeld während größere Brennweiten mit einem größeren Zoom einhergehen [28]. Die Brennweite bestimmt den Öffnungswinkel der Kamera. Bei einer geringen Brennweite ist der Öffnungswinkel groß und bei einer großen Brennweite klein.

Belichtungsdauer und Bewegungsunschärfe Für die Aufnahme von Bildern wird Licht auf einem Sensor eingefangen. Die Dauer, über die das Licht eingefangen wird, wird als Belichtungsdauer bezeichnet. Während dieser treffen Photonen auf den Sensor, welche für die Helligkeit der Bildpartien verantwortlich sind. Eine lange Belichtungsdauer resultiert in mehr eingefangenen Photonen und einem helleren Bild. Wird die Kamera während der Belichtung des Sensors bewegt, treffen Photonen unterschiedlicher Ursprünge auf dieselbe Position des Sensors und es entsteht Bewegungsunschärfe. In einem Bild äußert sich diese durch verzogene Linien und unscharfe Aufnahmen [29].

ISO und Rauschen Bei der Aufnahme von Bildern wird zwischen zwei Arten von Rauschen unterschieden: temporales und fixiertes Rauschen. Wohingegen fixiertes Rauschen zwischen Aufnahmen gleich bleibt, ändert sich temporales Rauschen zwischen Aufnahmen nichtdeterministisch. Die Ursprünge dieses Rauschens sind weitreichend von Imperfektionen in Objektiven zu physikalischen Gegebenheiten durch die Diskretisierung einer Szene auf dem Kamerasensor. Ein wesentlicher Grund für die Existenz von Rauschen ist die ISO. Der ISO-Wert gibt die Empfindlichkeit des Kamerasensors und damit die Lichtmenge an, die bei der Aufnahme mit einer Kamera auf den Sensor gelangt. Je höher der ISO-Wert ist, desto sensitiver ist der Kamerasensor für eintreffende Lichtstrahlen, wodurch hohe ISO-Werte übermäßiges Rauschen mit sich ziehen können [30]. Dieses Rauschen wird insbesondere bei automatischer Einstellung der Kameraparameter in dunklen Umgebungen deutlich, wodurch dunkle Aufnahmen mit starkem Rauschen entstehen. Diese Art von Bildern ist insbesondere bei Aufnahmen in Mobiltelefonen vermehrt zu finden, weshalb es im Kontext dieser Thesis besonders relevant ist.

Farbsäume Kameralinsen bestehen aus Glas mit einem Refraktionsindex, der von der Wellenlänge des eintreffenden Lichtes abhängt. Daraus resultiert eine unterschiedliche Lichtbrechung der jeweiligen Lichtwellen und es entstehen Farbsäume in den aufgenommenen Bildern [31, 30]. Bei Farbsäumen handelt es sich um die prismatische Auf trennung der Farbinformationen, die besonders an Kanten von Objekten und am Rand des aufgenommenen Bildes verstärkt auftreten. In Aufnahmen äußern sich Farbsäume als bunte Artefakte, die typischerweise entlang von Kanten verlaufen. Aufgrund der Geometrie von Linsen sind Farbsäume stärker vertreten, je weiter sie vom Bildmittelpunkt entfernt sind.

2.1.4 Binärbilder und Masken

Binärbilder und Masken sind Bilddaten, in denen die Pixelwerte entweder 0 oder 1 sind. Mit dieser Art von Bilddaten können Informationen in Bildern gezielt hervorgehoben werden. Zur Hervorhebung eines Objektes in einem Bild wird eine Maske erstellt, deren Pixelwerte aller Pixel im Originalbild 0 sind, an denen das Objekt nicht vorhanden ist; alle Pixel, die im Originalbild Teil des Zielobjekts sind, werden durch einen Pixelwert von 1 dargestellt. Auf diese Weise wird eine binäre Maske erstellt, die Informationen zu einem Objekt in dem Bild anzeigt und die für die weitere Extraktion von Informationen verwendet werden kann.

In dieser Arbeit werden binäre Maskenbilder bei der Erstellung synthetischer Daten verwendet, um die Position und Orientierung von Objekten in gerenderten Bildern darzustellen. Dazu werden diejenigen Pixel hervorgehoben, die gewisse Eigenschaften aufweisen. Diese Eigenschaften reichen von der Geometrie der Dartscheibe bis zur exakten Bestimmung der Einstichstellen der Dartpfeile in die Dartscheibe.

2.1.5 Geometrie einer Dartscheibe

Die Geometrie und der Aufbau einer Dartscheibe ist für die Erstellung der Daten zentral. Eine schematische Darstellung einer Dartscheibe ist in Abbildung 2.1a gegeben.

Die Dartscheibe Die Dartscheibe besteht grundlegend aus einer Scheibe mit 451 mm Durchmesser. Sie besteht aus 20 einheitlich großen, radial angeordneten Feldern mit Zahlenwerten von 1 bis 20. Jedes Feld besitzt einen Double- und einen Triple-Ring mit einem Durchmesser von 8 – 10 mm, deren Wertigkeit der doppelten bzw. dreifachen Punktzahl des jeweiligen Feldes entspricht. Der Triple-Ring ist etwa 10 cm vom Mittelpunkt entfernt; der Double-Ring etwa 16 cm. Insgesamt ergibt sich ein Durchmesser von 34 cm punktezielnder Felder, der Bereich jenseits der 17 cm Abstand des Mittelpunktes gibt keine Punkte. In der Mitte der Dartscheibe befinden sich die Felder Bull und Double Bull (oder Bull's Eye). Sie haben Durchmesser jeweils ca. 32 mm und 12,7 mm. Das Bull gibt 25 Punkte, das Double Bull 50 Punkte [5].

Die Felder mit einfachen Punktzahlen sind abwechselnd schwarz und weiß gefärbt, die Mehrfach-Felder der schwarzen Felder sind rot und die der weißen Felder grün gefärbt. Das Bull ist grün und das Double Bull rot.

Die Außenseite der Dartscheibe Die Punktzahlen der Dartfelder werden durch aus Drahten gefertigten Zahlen angezeigt. Diese befinden sich an einem Ring am äußeren Ende der Dartscheibe und sind radial um die Dartscheibe angeordnet. Dieser Zahlenring ist nicht fest an der Dartscheibe montiert, sodass er nach Belieben rotiert werden kann, um eine ungleichmäßige Abnutzung der Dartscheibe auszugleichen.

Material Die Felder der Steeldarts-Dartscheibe werden aus Fasern hergestellt, typisch sind dabei Sisal-Fasern. Diese besitzen die Eigenschaft, dass ein eintreffender Dartpfeil nicht in sie eindringt, sondern diese lediglich verdrängt, wodurch bleibende Schäden durch Einstichlöcher gering gehalten werden.

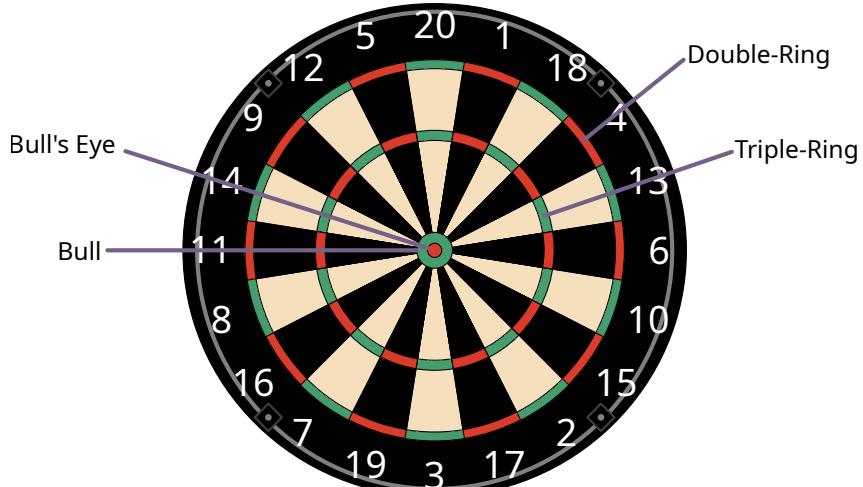
Double und Triple Mit Double und Triple (auch Treble) werden die Ringe an Feldern bezeichnet, die ein Vielfaches der Feldgrundzahl als Score erteilen. Der Triple-Ring befindet sich zwischen den Einzelfeldern der Dartscheibe und gewährt das Dreifache der Grundpunktzahl des Feldes. Der Double-Ring befindet sich auf der Außenseite der Dartscheibe und gewährt seinem Namen entsprechend das Doppelte der Feldzahl. Das Feld mit der größten Punktzahl ist die Triple-20 mit $3 \cdot 20 = 60$ Punkten und wird dadurch typischerweise am häufigsten anvisiert.

2.1.6 Dart-Terminologie

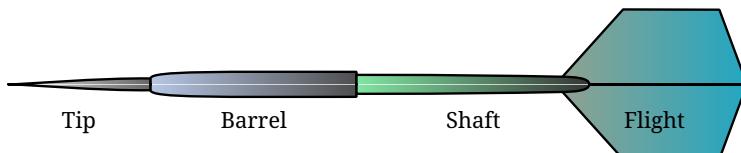
Im Darts gibt es eine Vielzahl an Begrifflichkeiten, von denen einige auch in dieser Thesis genutzt werden. Die grundlegenden Begriffe und ihre Bedeutungen werden in diesem Unterkapitel erläutert.

Tip, Barrel, Shaft, Flight Ein Dartpfeil besteht aus unterschiedlichen Bestandteilen. Neben möglichen weiteren Bestandteilen von Dartpfeilen sind die vier wesentlichen Bestandteile Tip, Barrel, Shaft und Flight¹, aufgezählt von der Spitze beginnend [5, 6]. Die Tip ist die Spitze des Dartpfeils, die in die Dartscheibe eintrifft. Die Barrel ist der Teil des Dartpfeils, an dem er gegriffen wird und liegt direkt hinter der Tip. Auf die Barrel folgt der Shaft, der die Brücke zum Flight, dem Flügelende des Dartpfeils, schließt. Der Flight besteht aus vier Einzelflügeln, die in Abständen von 90° zueinander stehen und orthogonal zum Shaft verlaufen. Der Aufbau eines Dartpfeils ist in Abbildung 2.1b schematisch dargestellt.

¹Abweichungen dieser Bezeichnungen sind möglich, sodass der Shaft auch häufig als Stem bezeichnet wird. Zur Vereinheitlichung werden die genannten Begriffe verwendet.



(a) Schematische Darstellung einer Dartscheibe.



(b) Schematische Darstellung eines Dartpfeils.

Abbildung 2.1: Schematische Darstellungen von Dartscheibe und Dartpfeil.

Spinne Als Spinne wird der Metallrahmen der Dartscheibe bezeichnet, der die Feldsegmente voneinander trennt. Die Ausprägung der Spinne ist unterschiedlich, jedoch ist ein Trend zu erkennen, dass neue Dartscheiben dünner und unauffälligere Spinnen besitzen als alte Dartscheiben. Je unauffälliger die Spinne ist, desto geringer ist die Wahrscheinlichkeit eines Bounce-Outs, bei dem der Dartpfeil auf die Spinne trifft und nicht in der Dartscheibe verbleibt, sondern reflektiert wird.

2.1.7 Steeldarts und Softdarts

Steeldarts ist die klassische Form von Darts, in der Dartpfeile mit Metallspitzen auf eine Dartscheibe geworfen werden. Die Bezeichnung des Steeldarts bezieht sich auf die Verwendung von Dartpfeilspitzen, die aus Metall gefertigt sind und ein spitzes Ende zum Eindringen in eine Dartscheibe besitzen. Dem gegenüber stehen Softdarts, bei welchem Pfeile mit Spitzen aus Kunststoff verwendet werden, die typischerweise auf eine elektronische Dartscheibe mit vordefinierten Löchern zum Eindringen der Dartpfeile geworfen werden.

Wesentlicher Unterschied zwischen Soft- und Steeldarts ist neben der unterschiedlichen Ausrüstung die Art des Scoring. Während die meist elektronischen Dartscheiben von Softdarts ein automatisches Scoring verwenden, muss die Punktzahl bei Steeldarts manuell gezählt werden. Eben dieser Unterschied ist Kern dieser Thesis, in der ein automatisches Scoring-System für Steeldarts entwickelt wird. Während der Einsatz von Softdarts auf Gelegenheitsspiele und Hobbynutzung ausgelegt ist, kommt Steeldarts in professionellen Umgebungen zum Einsatz.

2.1.8 Material und Texturen

Materialien und Texturen finden ihren Einsatz in der Erstellung von 3D-Szenen. Objekten sind Materialien zugewiesen, die das Aussehen und Verhalten der Objekte bei Lichteinfall bestimmen. Eine Textur beschreibt die Farbgebung eines Materials, das Material beschreibt die Interaktion der Textur mit Licht. Beim Eintreffen von Licht auf ein Material kann dieses unterschiedlich abgeändert werden. Wesentliche Eigenschaften von Materialien sind der Grad der Diffusität, Reflektivität und Absorption. Ein diffuses Material strahlt

eingehende Lichtstrahlen in zufällige Richtungen; reflektive Materialien spiegeln eingehende Lichtstrahlen; absorbierende Materialien haben keine Reaktion auf einfallendes Licht. Jedes Material gewichtet diese Parameter unterschiedlich, um eine charakteristische Lichtreaktion hervorzurufen.

Die Interaktion mit Licht ist zudem durch die sogenannte Normal Map eines Materials beeinflusst, die die Oberflächenbeschaffenheit des Objekts beschreibt. Neben der Geometrie des Objekts ermöglicht die Normal Map eine detaillierte Oberfläche, die die Lichtbrechung beeinflusst und realistische Interaktionen mit Licht ermöglicht.

2.1.9 Noise-Texturen

Essenziell für prozedurale Datenerstellung ist die Verwendung von Noise-Texturen. Diese ermöglichen es, Kontrolle über die Variation der Daten zu behalten während der Zufall der Datenerstellung erhalten bleibt. Es existieren unterschiedliche Arten von Noise-Texturen, die für die Generierung zufälliger Texturen verwendet werden [32]. In dieser Arbeit werden vorgehend White Noise und Perlin Noise in der Datengenerierung genutzt, welche in den folgenden Unterabschnitten genauer erläutert werden.

2.1.9.1 White Noise

White Noise – zu deutsch: weißes Rauschen – ist eine Zufallsverteilung von Zahlenwerten, die nicht vorhersehbar ist und doch einem Muster folgt. Sie ist dadurch charakterisiert, dass jeder Ausgangswert mit der selben Wahrscheinlichkeit versehen ist [33]. Im eindimensionalen Beispiel von Audiodaten entspricht weißes Rauschen der Charakteristik, dass jede Frequenz in einem Signal gleichermaßen vertreten ist. Hinsichtlich einer diskreten zweidimensionalen Textur ist die Wahl jedes Intensitätswerts eines Pixels unabhängig voneinander gleichverteilt über das Intervall $[0, 1]$.

2.1.9.2 Perlin Noise

Perlin Noise bildet die Basis für gezielte Generierung von Strukturen prozeduraler Natur. 1982 von Ken Perlin für den Film Tron entwickelt, zielte Perlin Noise in Vergleich zu White Noise auf die Generierung von zusammenhängendem Rauschen ab, das zur Generierung natürlicher Strukturen verwendet werden kann [34, 35].

Zur Berechnung einer 2D-Textur mit Perlin Noise wird ein Bild in Regionen unterteilt. Jeder Ecke dieser Regionen wird ein Vektor zugeordnet, der in eine zufällige Richtung zeigt, die uniform aus dem Intervall $[0, 2\pi]$ gewählt wird, wie in Abbildung 2.2 dargestellt. Für jeden Pixel (x, y) im Bild wird die korrespondierende Region bestimmt und die Punktprodukte zwischen den Vektoren der Ecken der Region und den Verbindungsvektoren zu dem Punkt werden gebildet. Die Intensität des Pixels wird durch bilineare Interpolation dieser Werte auf Grundlage seiner Position in der Region bestimmt. Auf diese Weise entsteht ein zusammenhängendes Muster, das als Perlin Noise bezeichnet wird [34].

Der Detailgrad von Perlin Noise wird durch Überlagerung mehrerer Schichten erzielt, die sich in der Größe ihrer Regionen unterscheiden; je kleiner die Regionen, desto größer der Detailgrad. Durch Variation der Regionsgrößen und Überlagerung mehrerer Schichten kann ein beliebiger Detailgrad erzielt werden, der in einem natürlichen Rauschen resultiert.

2.1.10 Seeding

Die algorithmische Generierung von Zufallszahlen ist nicht ohne Verzerrung möglich. Daher ist bei von Computern generierten Zufallszahlen die Rede von pseudo-zufälligen Zahlen. Die Generierung dieser pseudo-zufälligen Zahlen beruht auf deterministischen Algorithmen, die üblicherweise einen Seed verwenden, um Zahlen zu generieren, die zufällig erscheinen. Durch die Verwendung des selben Seeds ist die Generierung von Zufallszahlen deterministisch wiederholbar, wodurch die Ableitung von Daten reproduzierbar und nachvollziehbar ist.

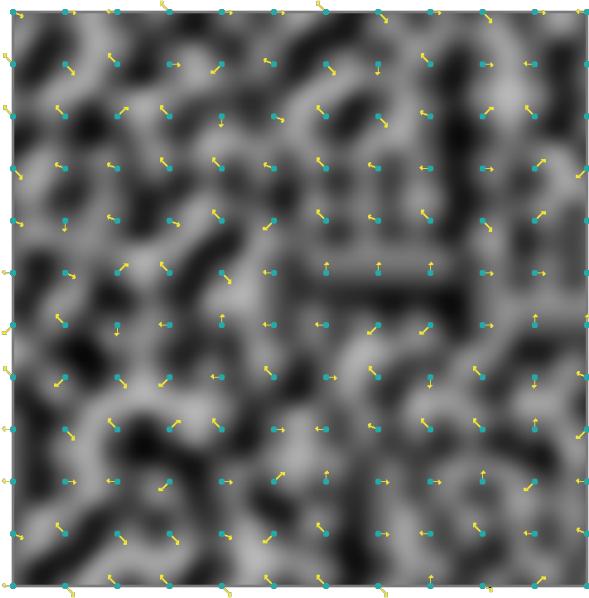


Abbildung 2.2: Generierung von Perlin Noise [36]. Zufällige Rotationsvektoren werden uniform über ein Bild verteilt. Durch diese wird die Intensität der Textur bestimmt.

2.1.11 Thresholding und Maskierung

Thresholding ist für einen Wert f und einen Threshold t definiert durch [37]:

$$\Phi(f, t) = \begin{cases} 1, & \text{wenn } f \geq t, \\ 0 & \text{sonst} \end{cases}$$

Im Rahmen der Datengenerierung dieser Thesis werden Thresholding und Maskierung verwendet, um Texturen gezielt miteinander zu kombinieren. Beim Thresholding wird ein Schwellenwert genutzt, um zu bestimmen, ob ein bestimmter Farbwert einer Textur berücksichtigt wird. Liegt die Pixelintensität über dem festgelegten Schwellenwert, wird der entsprechende Pixel einbezogen; andernfalls wird er verworfen. Die inverse Arbeitsweise ist ebenfalls möglich. Ein typischer Anwendungsfall des Thresholdings in dieser Arbeit besteht in der Begrenzung von Noise-Texturen auf Bereiche mit Pixelintensitäten oberhalb eines definierten Schwellenwerts.

Wird globales Thresholding aller Pixel lokalisiert, sodass jedem Pixel ein dedizierter Threshold zugeordnet wird, spricht man von Maskierung. Eine Maske definiert die Schwellenwerte, die für die Einblendung einer modulierten Textur erforderlich sind. Neben den binären Ein- oder Ausblendung von Pixeln ermöglichen Masken durch kontinuierliche Werte im Intervall $[0, 1]$ eine anteilige Darstellung der Quelltextur. Diese Methode findet in dieser Thesis Anwendung bei der Erstellung einer realistischen Dartscheibe, indem mehrere maskierte Texturen überlagert werden.

2.1.12 Prozedurale Texturen

Prozedurale Texturen sind der Grundbaustein der Datenerstellung in dieser Thesis. Sie dienen als Blaupause zur Generierung zufälliger Texturen, indem sie ein generelles Erscheinungsbild einer Textur definieren. Dieses Erscheinungsbild ist aufgebaut aus Bestandteilen, die Seeding integrieren, um ihr Aussehen zu bestimmen. Die Änderung des verwendeten Seeds führt folglich zu einer Änderung der Textur. Der Grad der Änderung ist vorhersehbar und kann durch Grenzwerte vorgegeben sein; die konkrete Änderung unterliegt jedoch weiterhin der Pseudo-Zufälligkeit. Unter Einbindung dieser Technik kann eine beliebige Anzahl unterschiedlicher Texturen generiert werden.

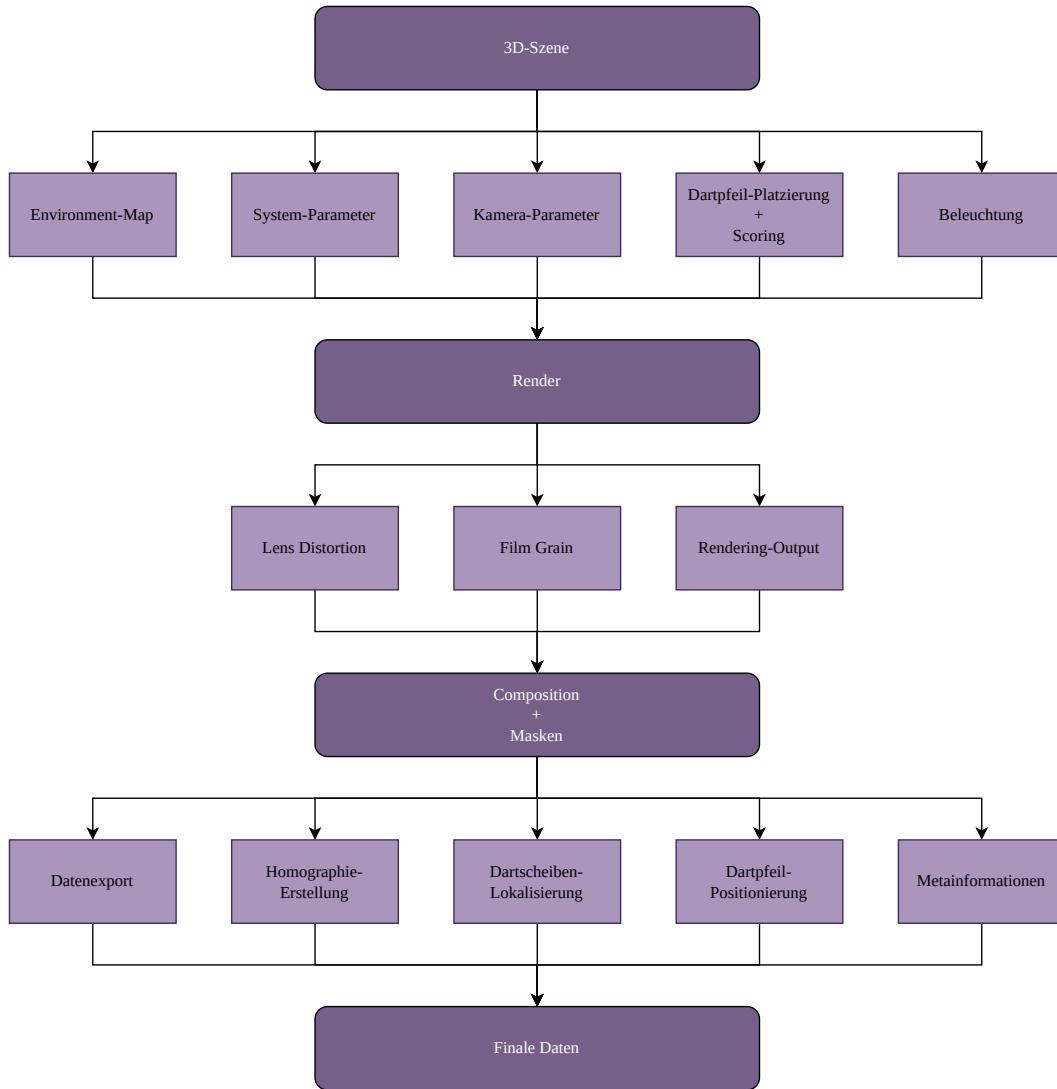


Abbildung 2.3: Schematische Darstellung der Rendering-Pipeline.

2.2 Methodik

In diesem Abschnitt wird die Methodik der automatisierten Datenerstellung thematisiert. Es werden die unterliegenden Konzepte erläutert und die Hintergründe dieser beschrieben. Bevor in die einzelnen Bereiche eingestiegen wird, ist es zunächst notwendig, einen Überblick über das Zusammenspiel der jeweiligen Komponenten zu gewinnen. In Abbildung 2.3 ist der Ablauf der Datenerstellung schematisch dargestellt.

Die Datenerstellung fußt auf einer 3D-Szene, beschrieben in Unterabschnitt 2.2.1, in der sich Objekte befinden, die für die Gestaltung der Trainingsdaten verantwortlich sind. Hintergründe zu ihrem Aufbau und der konkreten Auswahl der Objekte werden in Unterabschnitt 2.2.2 gegeben. Diese Objekte der Szene werden durch ein Skript hinsichtlich ihrer Existenz, ihres Aussehens, ihren Eigenschaften und ihrer Positionierung algorithmisch randomisiert, welche durch ein externes Skript geschieht, das in Unterabschnitt 2.2.3 näher betrachtet wird. Das Rendern der Szene geschieht unter Einbeziehung von Imperfektionen wie Rauschen und Verzerrungen, wie es in Kameras aus Mobiltelefonen vorkommt, wodurch ein weiterer Schritt des Realismus der erstellten Daten erzielt wird. Zusätzlich werden binäre Masken unterschiedlicher Objekte gerendert, welche zur Extraktion relevanter Informationen in den Daten verwendet werden. Zuletzt werden durch Nachverarbeitungsschritte, welche in Unterabschnitt 2.2.4 detailliert beschrieben werden, in den Daten enthaltene Metainformationen extrahiert. Im Wesentlichen umfasst dies die Lokalisierung der Dartpfeilspitzen in den gerenderten Bildern und die Ermittlung der Normalisierung.

2.2.1 Der Aufbau der 3D-Szene

Das Fundament der Datengenerierung ist die Simulation realistischer Dartscheiben. Diese Simulation fußt auf der virtuellen 3D-Szene, deren Grundzüge in den folgenden Unterkapiteln beschrieben werden. Es werden die zentralen Objekte der Szene beschrieben, die Dreh- und Angelpunkt der Datenerstellung darstellen und die für die Varianz und Komplexität der erstellten Daten ausschlaggebend sind. An Objekten werden die Dartscheibe, die Dartpfeile und die Beleuchtungsmöglichkeiten beschrieben, sowie die globalen Parameter der Szene.

2.2.1.1 Dartscheibe

Die Dartscheibe ist jenes Objekt, welches statisch in der Szene vertreten ist und in jedem gerenderten Bild vorhanden ist. Damit ist ihr Aussehen zentral für die Qualität der Daten. Sie ist gemäß der Richtlinien „Playing and Tournament Rules“ der World Dart Federation (WDF) erstellt [5]. Beschreibungen dieses Regelwerks werden in dieser Thesis als Quelle für Maße und Toleranzen genutzt; Dartscheiben mit esoterischen Maßen und Feldfarben, die nicht konform mit den Regeln sind, werden nicht explizit für diese Arbeit in Betracht gezogen. Zusätzlich zu diesen Regeln werden unterschiedliche reale Dartscheiben als Referenzen genutzt, die Gebrauchsspuren aufweisen und anhand derer zu erwartende Beschaffenheiten eingefangen werden.

2.2.1.2 Dartpfeile

Zusätzlich zentral für die Datenerstellung sind die Dartpfeile. Im Vergleich zur Dartscheibe ist das Aussehen der Dartpfeile nicht stark durch die Darts-Regulierungen vorgegeben [5, 6]. Lediglich die maximale Länge und das Gewicht sowie die grundlegende Zusammensetzung der Pfeile werden in den Regulierungen thematisiert. Dadurch ist die Spanne des Aussehens möglicher Dartpfeile sehr groß und muss dementsprechend behandelt werden, um systematische Fehler zu mindern.

2.2.1.3 Beleuchtung und Lichtobjekte

Für die Beleuchtung der Szene werden globale und lokale Beleuchtungen verwendet. Globale Beleuchtung wird durch die Verwendung von Environment Maps erzielt, während lokale Beleuchtung durch spezielle Lichtobjekte umgesetzt wird. Diese Lichtobjekte stellen unterschiedliche Beleuchtungsmöglichkeiten dar, die bei der Recherche zum Aussehen und den Umgebungen von Dartscheiben beobachtet wurden. Diese sorgen für eine vielseitige Beleuchtung der Szene.

2.2.1.4 Weitere Objekte

Eine weitere Beobachtung typischer Dartscheiben ist Anbringung von Dartscheiben in Dartschränken. Diese schützen die Wand vor an der Dartscheibe vorbei geworfenen Dartpfeilen und ermöglichen das Verschließen der Dartscheibe. In der Szene befindet sich zusätzlich ein Dartschrank aus Holz, dessen Farbe zufällig gesetzt wird. Die Existenz des Dartschranks ist verbunden mit der Abwesenheit eines Ringlichts, das in Unterabschnitt 2.2.2.3 beschrieben wird, da sich diese Objekte überschneiden.

2.2.1.5 Parametrisierung

Für Generierung von Zufallsvariablen stellt die Szene zwei Parameter zur Verfügung: Seed und Alter. Der Seed ist ein Wert in einem vorgegebenen Intervall, der zur deterministischen Generierung von Zufallsvariablen genutzt wird. Diese Zufallsvariablen werden in den Objekten genutzt, um Abnutzungen, Zusammensetzungen, Verschiebungen und Texturen zu beeinflussen (vgl. Unterabschnitt 2.3.1, Unterabschnitt 2.3.2). Auf diese Weise ist ein deterministisches Erstellen von Szenen möglich. Der Seed wird ebenfalls zur Generierung des Alters-Parameters genutzt. Dieser gibt das Alter der Objekte in der Szene an und wird verwendet, um den Grad der Abnutzung in Dartscheibe und Dartpfeilen zu bestimmen.

2.2.2 Material und Licht

Die Ausgestaltung der Objekte sowie die Arten der Beleuchtung sind essenziell für das Erstellen realistischer Daten und eine Abdeckung einer Vielzahl unterschiedlicher Szenarien. Für die Datenerstellung werden prozedurale Texturen verwendet, die in ihren Grundzügen reale Beobachtungen widerspiegeln. Die genaue Zusammensetzung und Ausgestaltung der Texturen wird in den folgenden Unterabschnitten genauer erläutert. Dazu werden insbesondere die Dartscheibe, die Dartpfeile und die Lichtquellen betrachtet.

2.2.2.1 Material der Dartscheibe

Die Dartscheibe besteht grundlegend aus den Darts-Feldern, der Spinne, dem Zahlenring und einer Beschriftung. Die Dartfelder sowie der Rahmen der Dartscheibe sind der Beschaffenheit von Sisal nachempfunden und werden mit unterschiedlichen Gebrauchsspuren versehen. Das Material von Sisal ist sehr diffus und wenig reflektiv. Die Oberfläche ist angerau und weist Unebenheiten auf.

Die Gebrauchsspuren des Sisals werden in Form von Abnutzung durch Kratzer, Einstichlöcher und Staubansammlung sowie durch Risse im Material dargestellt. Darüber hinaus ist zur Simulation von Alterung der Dartscheibe eine Verfärbung der Felder und Verstärkung der Abnutzungen eingebaut. Die Beschaffenheiten und Vorkommen dieser Abnutzungen sind realen Dartscheiben nachempfunden und zielen darauf ab, eine möglichst große Spanne unterschiedlicher Dartscheiben abzudecken.

Die Spinne der Dartscheibe ist als reflektives Metall modelliert. Sie ist ebenfalls von einem Alterungsprozess betroffen, da beobachtet wurde, dass die Dicke und Sichtbarkeit der Spinne bei Dartscheiben zunehmenden Alters stärker ausgeprägt sind. Historisch ist dies dadurch begründet, dass der Herstellungsprozess von Dartscheiben im Laufe der Zeit fortschrittlicher und die Integration weniger auffälliger Spinnen ermöglicht wurde.

Da alte Dartscheiben aktuell weiterhin in Verwendung sind, ist eine Abdeckung ihrer Geometrien in der Datenerstellung von Relevanz. Die Spinne sowie der Zahlenring unterliegen darüber hinaus einer Wahrscheinlichkeit, von Rostbildung betroffen zu sein, und werden mit steigendem Alter der Dartscheibe zunehmend stark verformt, sodass sie nicht auf den zu erwartenden Positionen liegen. Diese Verformungen wurden ebenfalls auf realen Dartscheiben beobachtet und sind potenziell für Dartpfeile, die nahe der Spinne landen, relevant.

Die Beschriftung der Dartscheibe beinhaltet typischerweise den Herstellernamen der Dartscheibe sowie Symbole und Logos. Diese werden mit zufälligen Texten approximiert, die entlang des Randes der Dartscheibe verlaufen und zufällig platziert werden. Konkrete Informationen zur Umsetzung der Texturierung der Dartscheibe und zum Aufbau des Materials sind in Unterabschnitt 2.3.1 zu finden.

2.2.2.2 Generierung der Dartpfeile

Obligatorische Bestandteile von Dartpfeilen beinhalten Tip, Barrel, Shaft und Flight. Aus jeweiligen Pools von Objekten werden Dartpfeile zufällig zusammengestellt, um randomisierte Dartpfeile zu generieren.

Die Tips der Dartpfeile sind wenig variabel und unterscheiden sich hauptsächlich in Länge und Farbe. Trotz ihrer geringen Größe ist eine realistische Modellierung der Tips bedeutend für die Daten, da sie ausschlaggebend für die erzielte Punktzahl sind. Neben silbernen Tips sind ebenfalls schwarze oder auch bronzenen Tips möglich, die in ihrer Reflektivität variieren.

Die Barrels sind im Vergleich zu den Tips wesentlich komplexer, sodass einige vordefinierte Barrels generiert sind, die zufällig hinter einer Tip angefügt werden. Darüber hinaus wird die Länge der Barrels zufällig variiert, um weitere Variationen einzubringen. Die Beschaffenheit von Barrels realer Dartpfeile ist sehr unterschiedlich, sodass eine sehr große Variabilität ihrer Erscheinungsbilder möglich ist. Um weitere Variabilitäten einzubinden, werden Materialien verwendet, deren Farbe zufällig gesetzt wird.

Auf die Barrel folgt der Shaft des Dartpfeils, der als Übergang zum Ende des Dartpfeils dient. Dieser wird ebenso wie die Barrel aus vorgefertigten Bauteilen ausgewählt, in seiner Länge modifiziert und mit zufälligen Farben versehen. Reale Dartpfeile folgen häufig einem kohärenten Farbschema während die Abstimmung von Barrels und Shafts in dieser Thesis zufällig ist.

Das Ende der Dartpfeile bilden die Flights. Diese sind die meist aus Plastik gefertigten Flügel des Dartpfeils und ihr Erscheinungsbild variiert von allen Bestandteilen am stärksten. Farben von Flights reichen von einzelnen Farben über Flaggen und Wappen bis hin zu abstrakten Bildern. Zusätzlich ist die Form von Flights nicht vorgegeben. Diese Gegebenheiten werden in dieser Thesis durch Projektion eines zufälligen Bereichs eines Texturatlats² auf eine Grundformen für Flights realisiert. Die Grundformen sind ebenfalls vorgefertigt und orientieren sich an realen Formen für Flights. Weiterhin ist eine Verformung der Flights als Gebrauchsspuren von Dartpfeilen identifiziert worden, die ebenfalls in die Generierung der Dartpfeile einbezogen ist. Die Materialien der Flights variieren in ihrer Reflektivität, sind jedoch sehr glatt und dem Erscheinungsbild von Plastik nachempfunden. Die Umsetzung dieser Methodiken für die Datenerstellung wird in Unterabschnitt 2.3.2 der Implementierung beschrieben.

2.2.2.3 Lichtquellen

Hinsichtlich der Beleuchtungsmöglichkeiten der Szene existieren unterschiedliche Objekte, die jeweils unterschiedliche Auswirkungen der Beleuchtung mit sich ziehen. Für die Datenerstellung sind fünf unterschiedliche Arten der Beleuchtung modelliert.

Environment Maps Environment Maps, auch als HDRIs bezeichnet, sind 360°-Aufnahmen von realen Umgebungen. Diese können bei dem Rendern von Szenen als Hintergrund genutzt werden, sodass diese zur Ausleuchtung der Szene dienen. Dadurch ist die Simulation realistischer Beleuchtungen möglich, ohne die jeweilige Szene nachzustellen. Die Intensität der Environment Maps bestimmt dabei die Ausprägung der Beleuchtung, sodass eine Intensität < 1 die Helligkeit der Environment Map reduziert, sodass eine Spanne unterschiedlicher Beleuchtungen unter der Verwendung der selben Environment Map möglich ist. Für diese Arbeit werden 208 Environment Maps von der Webseite PolyHaven bezogen [39].

Kamerablitz Wenige Zentimeter neben der Kamera befindet sich ein Punktlicht, das als Kamerablitz fungiert. Es kann ein- und ausgeschaltet werden und sorgt für eine helle Ausleuchtung der Szene. Die Farbe des Lichtes ist kaltweiß und sorgt durch seine Positionierung für harte Schatten entlang der Kanten von Objekten im Bild. Besonders stark ist dieser Effekt bei Dartpfeilen zu beobachten.

Spotlight Bei der Aufnahme realer Daten ist die Existenz von Spotlights aufgekommen. Bei Spotlights handelt es sich um ein oder mehrere Lichter, die auf die Dartscheibe gerichtet sind und den Feldbereich ausleuchten. Diese Art der Ausleuchtung sorgt ebenfalls für einen auffälligen Schattenwurf und wird in der 3D-Szene als Flächenlicht mit variierender Größe modelliert. Diese Art der Beleuchtung wird beispielsweise im Strongbows Pub³ in Kiel eingesetzt, aus welchem Teile der realen Daten stammen.

Ringlicht Ein typisches Accessoire für Dartscheiben sind Ringlichter. Diese bestehen aus einem Gestell, das an der Dartscheibe befestigt wird. An diesem werden LEDs in einem Ring vor der Dartscheibe angeordnet, um diese direkt zu beleuchten. Ringlichter sorgen für eine uniforme Ausleuchtung der Dartscheibe und wenig Schattenwurf der Pfeile. Modelliert sind Ringlichter nach dem Vorbild der Dartscheiben in Jess Bar in Kiel. Die LEDs sind bei dieser Art des Ringlichts an der Vorderkante eines zylindrischen Korpus angebracht, in dem die Dartscheibe befestigt ist. Dieser Korpus kann unterschiedliche Farben besitzen.

Deckenbeleuchtung Zusätzlich zu den bereits genannten Beleuchtungsmöglichkeiten existieren Deckenleuchten in der Szene, die für eine warmweiße Beleuchtung sorgen. Diese Lichter werden unter anderem als Rückfall-Beleuchtung verwendet, sofern – mit Ausnahme der Environment Maps – keine andere Beleuchtung aktiviert ist. Dadurch ist sichergestellt, dass keine unbeleuchtete Szene entsteht.

²Der Texturatlas beinhaltet Länderflaggen sowie geometrische Formen und zufällige Farben. Teile des Atlas sind durch ein Bilderstellungs-Tool von DeepAI [38] generiert.

³<https://www.strongbowspub.de>

2.2.3 Scripting

Neben der 3D-Szene wird ein externes Skript genutzt, mit dem auf die Szene zugegriffen wird und durch das Einstellungen gesetzt werden. Dieses Unterkapitel thematisiert den Hintergrund und die Arbeitsweise dieses Skriptes.

2.2.3.1 Notwendigkeit eines externen Skriptes

Obwohl die 3D-Szene der zentrale Punkt der Datenerstellung ist, geschieht das Setzen spezifischer Parameter und das Rendern der Szene durch ein Skript. Der Hintergrund dessen ist die Flexibilität einer programmatischen Herangehensweise im Vergleich zum strikten Modellieren. Darüber hinaus ist die automatisierbare Arbeitsweise und der Verzicht auf eine grafische Nutzeroberfläche prädestiniert für die Erstellung einer großen Menge an Daten.

2.2.3.2 Einfluss der Parametrisierung

Für jedes generierte Sample ist der erste zentrale Schritt des Skriptes das Setzen des Seeds. Dieser beeinflusst das Aussehen der Objekte in der Szene, die Beschaffenheit der Dartscheibe sowie die Zusammensetzung der Dartpfeile, jedoch nicht ihre Positionierung oder Existenz. Er hält einen zufälligen Wert, dem keine konkrete Bedeutung zugeschrieben ist. Dieser Wert wird als Seed zur Generierung von Zufallsvariablen genutzt, die wiederum in den Materialien und Geometrien der Objekte eingesetzt werden. Der tatsächliche Wert des Seeds wird einzig genutzt, um das Alter der Szene zu bestimmen. Kleine Werte werden als geringes Alter interpretiert, große Werte als hohes Alter. Das Alter schlägt sich in dem Erscheinungsbild der Dartscheibe und der Dartpfeile nieder.

2.2.3.3 Spezifisches Setzen von Parametern

Der Seed und der Altersparameter bestimmen weitestgehend das Aussehen der Szene, jedoch existiert die Ausnahme der Texte um die Dartscheibe. Diese Texte werden durch das Skript in ihrem Inhalt, der Schriftart und in ihrer Positionierung manipuliert.

Darüber hinaus ist das Setzen der Dartpfeilpositionen ein wichtiger Schritt des Skriptes. Diese werden anhand von Wahrscheinlichkeitsverteilungen auf der Dartscheibe verteilt und zufällig rotiert. Aus den Positionen im 3D-Raum kann die erzielte Punktzahl des simulierten Dartspiels abgeleitet werden.

Die Kamera wird von dem Skript in einem vorgegebenen Raum positioniert und ihre internen sowie externen Parameter werden in definierten Intervallen randomisiert, zu Teilen basierend auf ihrer Position relativ zur Dartscheibe. Es werden unter anderem Brennweite, Fokuspunkt, Auflösung und das Seitenverhältnis gesetzt, um einer Datenverzerrung hinsichtlich spezifischer Kameraeinstellungen vorzubeugen. Würden alle Sample die selben Kameraparameter nutzen, bestünde die Gefahr der Spezialisierung eines aus diesen Daten trainierten Systems auf diese Gegebenheiten. Dadurch besteht die Gefahr der fehlerhaften Vorhersage auf Daten, die nicht diese exakten Kameraparameter vorweisen. Durch Randomisierung der Parameter wird diese Einschränkung der Generalisierungsfähigkeit umgangen.

2.2.3.4 Statische und dynamische Objekte

Die Präsenz bzw. Absenz einiger Objekte in der Szene wird ebenfalls durch das Skript gesteuert. Somit ist eine Unterscheidung möglich zwischen statischen Objekten, die in jeder Szene vorhanden sind, und dynamischen Objekten, die nicht in jeder Szene vorhanden sind. Statische Objekte der Szene sind die Dartscheibe, die Kamera und die Environment Map.

Zu dynamischen Objekten zählen unter anderem die Dartpfeile, deren Existenz durch eine Wahrscheinlichkeitsverteilung beeinflusst wird. Um eine variable Anzahl an Würfen abzubilden, ist ein zufälliges Ausblenden der Dartpfeile möglich. Darüber hinaus zählen jegliche Lichtobjekte zu den dynamischen Objekten, da sie ebenfalls durch das Skript ein- und ausgeblendet werden können. Weiterhin ist der Dartschrank ein dynamisches Objekt, dessen Existenz von dem Skript gesteuert wird. Diese Existenz ist an die Abwesenheit gewisser Lichtobjekte geknüpft, sodass keine ungewünschte Überschneidung von Objekten in der Szene vorkommt.

2.2.3.5 Rendern von Bildern

Der Anstoß zum Rendern von Bildern aus der Szene geschieht ebenfalls durch das Skript. Nachdem die Szene vorbereitet ist, wird ein Bild aus der Sicht der Kamera gerendert. In diesem Bild ist eine randomisierte Dartscheibe mit einem zufälligen Dartwurf abgebildet, die aus einem zufälligen Winkel in einer variierenden Beleuchtung eingefangen wird. Zusätzlich zu diesem Bild werden weitere Masken von Objekten gerendert, die als binäre Schwarzweißbilder exportiert werden und lediglich einzelne Objekte zeigen. Unter anderem werden die Dartpfeile, die Fläche der Dartscheibe, die Schnittpunkte der Dartpfeile mit der Dartscheibe sowie Orientierungspunkte der Dartscheibe, wie sie im DeepDarts-System verwendet werden, generiert. Durch diese Masken wird die Ableitung unterschiedlicher Informationen aus dem Bild genutzt.

Zusätzlich zum Speichern dieser Bilder werden Metainformationen zu der Szene exportiert. Diese beinhalten u. a. Informationen zu Kameraparametern, Punktzahl, Existenz von Objekten und dem Kamerawinkel zur Dartscheibe. Diese Informationen dienen der Annotation der Daten sowie der Erhebung von Statistiken.

2.2.4 Nachverarbeitung und Fertigstellung der Daten

Nach dem Rendern von Bildern und Masken der Szene erfolgt eine Nachverarbeitung der Daten. Dabei werden weitere Informationen von den gespeicherten Informationen angeleitet und für das Training eines neuronalen Netzes vorbereitet.

2.2.4.1 Normalisierung der Dartscheibe

Das Training des neuronalen Netzes zum Scoring von Dartsrunden in dieser Thesis geschieht mit normalisierten Bildern. Da das Ziel der Datenerstellung eine Nachempfindung möglichst realistischer Daten ist, die in ihrem Umfang möglichst variabel sind, besteht eine Diskrepanz zwischen gerenderten Bildern und Netzwerk-Inputs, indem die gerenderten Bilder nicht normalisiert sind. In der Inferenz des in dieser Thesis entwickelten Gesamtsystems wird diese Diskrepanz durch die in Kapitel 3 dargestellte Normalisierung geschlossen. Zur Überführung gerenderter Bilder zu normalisierten Bildern wird ein Nachverarbeitungsschritt der Datenerstellung eingebunden.

Die Normalisierung der Dartscheibe basiert auf Orientierungspunkten, deren Positionen relativ zur Dartscheibe bekannt sind und deren Positionen in den gerenderten Bildern durch binäre Masken ermittelt werden. Da die Positionen aller Punkte in den normalisierten Bildern bekannt sind, ist ein Überführen der Orientierungspunkte in gerenderten Bildern auf ihre Zielpositionen in normalisierten Bildern trivial zu ermitteln. Diese Herangehensweise hat sich bereits im DeepDarts-System bewährt und wird daher analog in dieser Thesis angewendet [1].

Dazu werden vier Orientierungspunkte ermittelt, die in konstanten Abständen entlang der Außenkante der Dartfelder verteilt sind. Durch diese ist eine eindeutige Entzerrung der Dartscheibe möglich, indem eine Homographie gebildet wird. Dieser Ansatz wird ebenfalls in Kapitel 3 verfolgt, um eine Entzerrung zu ermitteln.

2.2.4.2 Identifizierung von Dartpfeilpositionen in Bildern

Beim Rendern der Bilder werden unter anderem Masken von Einstichstellen der Dartpfeile in die Dartscheibe sowie Masken der Dartpfeile erstellt. Durch Überlagerung dieser Masken ist eine eindeutige Zuordnung von Dartpfeilen und Einstichstellen möglich, die eine Korrelation zwischen den Positionen im Bild und der erzielten Punktzahlen ermöglicht.

Unter Verwendung der im vorherigen Unterabschnitt erläuterten Homographie zur Normalisierung der Dartscheibe lassen sich die Positionen der Dartpfeile im Ausgangsbild auf Positionen im normalisierten Bild überführen. Durch die Verwendung von Masken der Einstichstellen ist eine fehlerfreie Positionierung der Einstichstellen in den Bildern möglich, obgleich diese in den Bildern sichtbar sind oder von anderen Dartpfeilen verdeckt sind. Dieser Punkt ist ein wesentlicher Vorteil automatisierter Datenerstellung gegenüber manueller Annotation, da in jedem Fall davon auszugehen ist, dass keine Ungenauigkeiten oder fehlerhafte Annotationen in den Daten enthalten sind.

2.2.4.3 Erhebung von Statistiken

Zusätzlich zu den essenziellen Informationen zum Trainieren eines neuronalen Netzes werden Metainformationen zu den generierten Daten abgeleitet und gespeichert. Diese dienen der Erhebung von Statistiken und ermöglichen einen potenziellen Einblick in die Stärken und Schwächen eines Systems. Inbegriffen in diesen Daten sind unter anderem Informationen zur Orientierung der Kamera und ihren internen Parametern.

Durch Approximation von Ellipsen und Linien auf Grundlage der Orientierungspunkte und Dartscheiben-Maske ist eine geometrische Beschreibung der Dartscheibe möglich. Die Verwendung einer solchen geometrischen Beschreibung der Dartscheibe und ihrer Ausrichtung im Bild wurde im Verlauf der Thesis verworfen, jedoch existieren die notwendigen Daten weiterhin als Relikte in den Daten. Diese können potenziell genutzt werden, um unterschiedliche Herangehensweisen zur Lokalisierung der Dartscheibe zu implementieren, beispielsweise durch Approximation der Dartscheibengeometrie als Ellipse.

2.3 Implementierung

Nachdem die grundlegende Methodik zur Datenerstellung erläutert ist, widmet sich dieser Abschnitt einem Einblick in relevante Details der Implementierung. Diese dienen dem tiefgehenden Verständnis der Umsetzung der Datenerstellung. Für die Implementierung der Datenerstellung wird sich für die Verwendung von Blender entschieden [40]. Blender ist eine Open-Source-Software, die eine realistische Modellierung von 3D-Szenen ermöglicht. Zusätzlich bietet es die Möglichkeit, durch Python-Skripte auf diese Szenen zuzugreifen und sie programmatisch zu manipulieren. Diese Eigenschaften bilden die Grundlage, mit der die für diese Arbeit relevanten Aufgaben bewältigt werden. Wie diese bestimmte Aspekte dieser Umsetzung vorgenommen werden, wird in diesem Abschnitt erläutert.

Es wird in einem ersten Unterabschnitt betrachtet, wie die Implementierung der Parameter in der Dartscheibe und ihren Materialien stattfindet. Danach wird auf die Zusammensetzung der Dartpfeile und ihre Positionierung eingegangen. Zuletzt werden die konkreten Einstellungen für das Rendern erläutert und es wird beschrieben, wie die automatische Normalisierung der Dartscheibe geschieht.

2.3.1 Parametrisierung der Dartscheibe

Die Dartscheibe ist auf unterschiedliche Weisen parametrisiert. Sie integriert sowohl den globalen Seed als auch den von diesem abgeleiteten Altersfaktor, um ihr Aussehen parametrisiert zu steuern. Grundlegend basiert das Aussehen jeder Dartscheibe auf einer idealen, neuen Dartscheibe. Die Dartscheibe besitzt ein Grundmaterial, dessen Beschaffenheit von Sisalfasern inspiriert ist und farblich einer neuen Dartscheibe entspricht.

Der Einfluss des Seeds für diese Dartscheibe beläuft sich auf eine leichte Variation von Farben und Oberflächenstruktur. Mit zunehmendem Altersfaktor werden die Farben der Felder zwischen dem Grundfarbton und einer Variante für Farben alter Dartfelder interpoliert, sodass ein Altern der Dartscheibe mit einer Verfärbung der Felder einhergeht.

2.3.1.1 Parametrisierung von Spinne und Zahlen

Hinsichtlich ihrer Geometrie wird die Dartscheibe durch eine Verformung der Spinne und einer Verschiebung der Zahlen mit zunehmendem Alter beeinflusst. Die Verschiebung dieser Objekte wird durch eine auf Perlin Noise beruhenden Translation der Vertices im Mesh realisiert. Die Magnitude dieser Translation ist durch den Altersfaktor bestimmt, sodass neue Dartscheiben minimale und alte Dartscheibe starke Verschiebungen aufweisen. Zusätzlich wirkt sich der Altersfaktor auf Spinne und Zahlen aus, indem Rost mit steigendem Alter häufiger vertreten ist. Dieser ist als überlagerte Noise-Textur mit rost-rotem Farbton in das Material eingearbeitet.

2.3.1.2 Parametrisierung des Materials

Die Dartscheibe weist Gebrauchsspuren durch Einstichlöcher, Risse und Staubpartikel auf, die ebenso wie ihre Farbe von den Parametern des globalen Seeds und von dem Altersfaktor beeinflusst werden. Umgesetzt ist dies durch die Verwendung von Noise-Texturen, Maskierungen und Variation der Stärken dieser Techniken, um den Einfluss von Alter zu integrieren.

Einstichlöcher Die Einstichlöcher werden durch ein Zusammenspiel mehrerer Masken generiert. Zur Simulation dieser wird eine Maske verwendet, anhand welcher Einstichlöcher uniform über die gesamte Dartscheibe verteilt werden. Die Stärke und damit die Existenz der Einstichlöcher wird moduliert durch eine weitere Maske. An den Stellen, an welchen die Existenzmaske stark vorhanden ist, treten Einstichlöcher vermehrt auf im Vergleich zu Bereichen mit geringer Ausprägung der Existenzmaske. Der Altersfaktor bestimmt einerseits die Dichte der Einstichlöcher durch Manipulation der Skalierung der Einstichlockmaske, andererseits wird zudem die Stärke der Existenzmaske derart vom Altersfaktor beeinflusst, dass ein Zunehmen des Alters mit mehr Einstichlöchern einhergeht.

Risse Analog zu den Einstichlöchern werden die Risse im Material parametrisiert. Es existieren ebenfalls zwei Masken, die die Ausprägung der Risse und ihre Existenz bestimmen. Die Ausprägung der Risse ist durch eine verzerrte Voronoi-Textur gegeben, deren Verzerrung und Größe durch den Altersfaktor beeinflusst wird. Die Existenzmaske der Risse wird analog zur Existenz der Einstichlöcher gehandhabt. Die Existenzmasken der unterschiedlichen Charakteristiken werden zudem mit unterschiedlichen Variationen des globalen Seeds generiert, sodass eine Korrelation der Masken ausgeschlossen ist.

Staubpartikel Die Staubpartikel setzen sich aus Masken für kleine Haare und Staubpartikel selbst zusammen. Diese werden ebenso wie die zuvor beschriebenen Charakteristiken durch Masken der Existenz und Ausprägung erstellt. Staubpartikel sind modelliert als kleine Punkte während Haare als kurze Striche dargestellt sind.

Die erstellten Masken der Einstichlöcher, Risse und Staubpartikel werden jeweils mit eigenen Texturen versehen und beeinflussen teilweise die Oberflächenbeschaffenheit der Dartscheibe durch Beeinflussung der Normal Maps des Materials.

2.3.2 Zusammensetzung der Dartpfeile

Die Generierung von Dartpfeilen beruht auf der Nutzung von Geometry Nodes in Blender, welche die Möglichkeit der deskriptiven und Node-basierten Zusammensetzung von Objekten ermöglichen. Weiterhin ist die Einbindung von externen Parametern wie dem globalen Seed der Szene zur Steuerung von Zufallsvariablen durch sie ermöglicht. Aufgebaut werden die Dartpfeile aus einem Pool unterschiedlicher vordefinierter Objekte, die auf Grundlage des globalen Seeds zu einem zufälligen Dartpfeil zusammengesetzt werden. Die für die Generierung vordefinierten Objekte sind in Abbildung 2.4 aufgelistet.

Tips Der erste Schritt zur Generierung eines Dartpfeils ist die Wahl der Tip. Sie wird aus einem Pool von vier Objekten gewählt und ihre Spitze wird zum Ursprung des finalen Dartpfeils. Die Helligkeit und Reflektivität der Textur der Tip wird auf Grundlage des Seeds zufällig gesetzt, sodass ein möglichst großes Spektrum unterschiedlicher Dartpfeilspitzen abgedeckt wird.

Barrels Auf die Tip folgt die Platzierung der Barrel. Die Barrel wird aus einem Pool von sieben Objekten ausgewählt, welche lückenlos hinter der Tip platziert werden. Die unterschiedlichen Barrel-Objekte verwenden verschiedene Methoden der Texturierung, sodass die Materialien einiger Barrel statisch vorgegeben sind, wohingegen andere Barrel ebenso wie die Tips zufällig texturiert werden, jedoch über alle RBG-Kanäle. Darüber hinaus wird die Geometrie der Barrel bei ihrer Platzierung hinter der Tip um $\pm 20\%$ sowohl in ihrer Länge als auch im Durchmesser variiert.

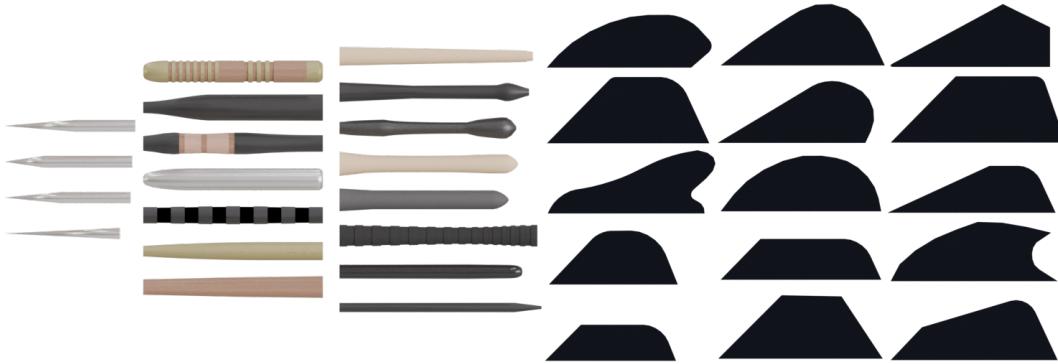


Abbildung 2.4: Bestandteile von Dartpfeilen. Von links nach rechts: Tips, Barrels, Shafts und Flights.

Shafts Im Anschluss an die Barrel wird der Shaft des Dartpfeils platziert. Dieser wird ebenfalls aus einem vordefinierten Pool von acht Objekten ausgewählt. Der Großteil dieser Objekte besitzt dynamische Texturen, die analog zu dynamischen Barrel-Texturen agieren. Ebenfalls wird die Geometrie der Shafts um $\pm 20\%$ in Länge und Durchmesser variiert.

Flights Die Flights sind die komplexesten Elemente der Dartpfeile, da sie die größte Spanne an Erscheinungsbildern besitzen. Ihr Aussehen variiert nicht nur durch ihre Farben, sondern auch durch ihre Form. Flights setzen sich aus vier gleichen Flügeln zusammen, die entlang des Dartpfeils in einem Abstand von 90° platziert sind. Um die Variation der Formen einzufangen, sind 15 unterschiedliche Formen für Flights modelliert, die sich an realen Formen von Flights orientieren. Ihre Textur wird aus einem Texturatlas mit einer Größe von 1.920×1.920 px gesampled. Dieser besteht aus neun unterschiedlichen Grundtexturen, bestehend aus Landesflaggen und abstrakten Formen unterschiedlicher Farbpaletten. Abhängig vom Altersfaktor wird die Verformung der Flights gesteuert, sodass neue Flights keine Deformierungen aufweisen, alte Flights jedoch stark deformiert werden, um Gebrauchsspuren zu simulieren.

Alle Dartpfeile einer Szene nutzen den selben Geometry Nodes, sodass alle Dartpfeile einer Szene gleich aufgebaut sind. Eine Variation der Dartpfeile innerhalb einer Szene ist möglich, es wird sich jedoch gegen diese Art der Umsetzung entschieden, da die Verwendung unterschiedlicher Dartpfeile innerhalb einer Runde unwahrscheinlich ist. Unterschiedliche zufällig generierte Dartpfeile sind in Abbildung 2.5 dargestellt. Hervorzuheben sind die unterschiedlichen Farben und Formen der Flights sowie die variierenden Bestandteile und ihre Texturierung.

2.3.3 Generierung von Dartpfeilpositionen

Die Positionierung der Dartpfeile auf der Dartscheibe ist ausschlaggebend für die Variabilität der Daten. In der Umsetzung der Platzierung werden unterschiedliche Techniken verwendet, um realistische Verteilungen und Erscheinungsbilder zu erzielen. In den folgenden Kapiteln wird die Positionierung der Dartpfeile auf der Dartscheibe beschrieben und die Bestimmung der Existenz von Dartpfeilen zur Variation der Anzahlen der Dartpfeile wird erklärt. Danach werden die Setzung der Rotation und zuletzt die Bestimmung der erzielten Punktzahl erläutert.

2.3.3.1 Positionierung der Dartpfeile auf der Dartscheibe

Eine uniforme Wahrscheinlichkeitsverteilung der Dartpfeilpositionen folgt weder Erwartungen realer Spiele noch wird es dem Anspruch dieser Arbeit gerecht. Zur realitätsnahen Simulation von Dartsrunden wurden daher reale Wahrscheinlichkeitsverteilungen analysiert und diese sind in Form von Heatmaps in die Szene eingearbeitet.

Die für die Datengenerierung dieses Thesis genutzten Heatmaps sind in Abbildung 2.6 dargestellt. Es werden zwei unterschiedliche Heatmaps genutzt: Eine realistische Heatmap



Abbildung 2.5: Auswahl zufällig erstellter Dartpfeile des Systems.

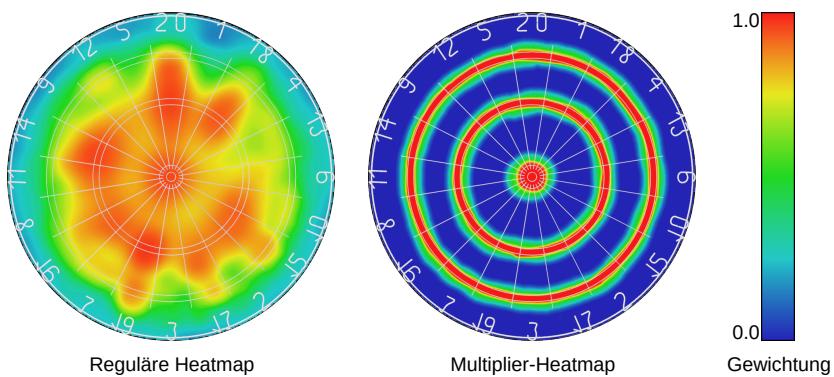


Abbildung 2.6: Heatmaps für die Datenerstellung; (links) Realistische Heatmap; (rechts) Multiplier-Heatmap für Oversampling der Daten.

und eine Heatmap zur gezielten Erstellung von Multiplier-Feldern und ihren Umgebungen. Tiefgehende Hintergründe für die Verwendung unterschiedlicher Wahrscheinlichkeitsverteilungen zur Positionierung von Dartpfeilen werden in Unterabschnitt 4.2.3.3 (Oversampling) erläutert. Die realistische Heatmap orientiert sich an den für DeepDarts gefundenen Wahrscheinlichkeitsverteilungen [1], Verteilungen aus Online-Recherchen [41] und eigenen Beobachtungen. Die Wahrscheinlichkeitsverteilungen dieser Heatmaps beziehen die gesamte Dartscheibe ein, sodass Treffer außerhalb der Dartfelder ebenfalls möglich sind.

Bei der Findung von Positionen der Dartpfeile geschieht die Platzierung auf Grundlage der aktiven Heatmap. Bereiche mit hohen Gewichten unterliegen einer höheren Wahrscheinlichkeit, als Position für einen Dartpfeil gewählt zu werden, als Bereiche mit geringen Gewichten. Durch eine Adaption der Heatmap, kann die Positionierung der Dartpfeile gezielt beeinflusst werden. So wird für die Datengenerierung dieser Arbeit eine weitere Heatmap erstellt, die sich auf die Multiplier-Felder und ihre Umgebungen fokussiert. Durch diese zweite Heatmap wird eine Erstellung von Daten ermöglicht, bei denen alle Dartpfeile entweder auf den Multiplier-Feldern liegen oder in ihrer Nähe. Treffer weit außerhalb der Dartscheibe sowie Treffer zentral in Einzelfeldern werden unter Verwendung der Multiplier-Heatmap nicht generiert.

Nach der Positionierung der Pfeile auf der Dartscheibe wird eine Nachverarbeitung vorgenommen, bei der Dartpfeile, die eine Überschneidung mit der Spinne aufweisen, von dieser entfernt werden. So wird sichergestellt, dass keine ambivalenten Dartpfeile existieren, die auf der Grenze zweier Felder eintreffen.

2.3.3.2 Bestimmung der Existenz von Dartpfeilen

Die Existenz von Dartpfeilen wird durch zwei Faktoren gesteuert. Vor der Positionierung eines Dartpfeils wird für jeden Pfeil entschieden, ob dieser existiert oder nicht. Anhand einer Wahrscheinlichkeitsverteilung werden die Dartpfeile zufällig ausgeblendet. Durch diese Zufallsentscheidung wird eine dynamische Anzahl an Dartpfeilen generiert.

Eine weitere Gegebenheit, unter der ein Dartpfeil ausgeblendet wird, ist die zu geringe Entfernung zu anderen Dartpfeilen. Liegt die Position eines Dartpfeils zu nahe an einem bereits platzierten Dartpfeil, wird dieser ausgeblendet. Es wird sich gegen eine Adaption der Position entschieden, um starke Abweichung von Heatmaps und erneute Überschneidung der Spinne zu vermeiden; eine neue Positionierung des Dartpfeils wird nicht eingesetzt, da die Möglichkeit besteht, dass die verwendete Heatmap nicht ausreichend Bereiche zur korrekten Platzierung aller Dartpfeile zur Verfügung stellt.

2.3.3.3 Rotation der Dartpfeile

Alle Dartpfeile, die nicht ausgeblendet sind, werden nach ihrer Positionierung rotiert. Die Rotation erfolgt unabhängig voneinander entlang ihrer x - y - und z -Achse. Die Rotation des Dartpfeils entlang der horizontalen x -Achse verläuft uniform im Intervall $[-5^\circ, 35^\circ]$. Diese Rotation bestimmt den Einschlagswinkel des Dartpfeils. Entlang der vertikalen y -Achse erfahren die Dartpfeile eine normalverteilte Rotation mit einer Standardabweichung von $\sigma = \frac{15^\circ}{3}$ um 0° mit einem Clipping einer maximalen Rotation von $\pm 15^\circ$. Die Rotation entlang ihrer z -Achse ist uniform im Intervall $[0^\circ, 360^\circ]$, um eine zufällige Drehung des Dartpfeils zu modellieren.

2.3.3.4 Ermittlung der Punktzahl

Nachdem die Dartpfeile positioniert und rotiert sind, wird das Scoring der Szene vorgenommen. An diesem Punkt sind die Position der Dartscheibe $p_{\text{Dartscheibe}} \in \mathbb{R}^3$ und die Positionen aller Dartpfeile $p_{\text{Pfeil},i} \in \mathbb{R}^3$ bekannt. Durch ihre Positionen zueinander lassen sich die Felder identifizieren, in denen die Dartpfeile eingetroffen sind. Auf diese Weise wird für jeden Dartpfeil ermittelt, in welchem Feld dieser eingetroffen ist und welche Punktzahl durch ihn erzielt ist. Das Scoring der Runde erfolgt durch Summation der Punktzahlen der einzelnen Dartpfeile.

2.3.4 Ermittlung von Kameraparametern

Die Kamera ist durch eine Vielzahl intrinsischer wie extrinsischer Parameter charakterisiert. Während einige Parameter statisch gesetzt oder durch einfache Wahrscheinlichkeitsverteilungen modelliert sind, zeigen andere Parameter wesentlich komplexere Verhalten auf. Dieser Unterabschnitt ist dafür vorgesehen, die Umsetzungen der komplexen Parameter genauer darzustellen. Es wird begonnen mit der Betrachtung der Kamerapositionierung, danach folgt das Setzen der Brennweite der Kamera und die Wahl von Seitenverhältnis und Auflösung des exportierten Bildes. Abschließend wird die Wahl des Fokuspunkts und die Umsetzung von verwackelten Bildern beschrieben.

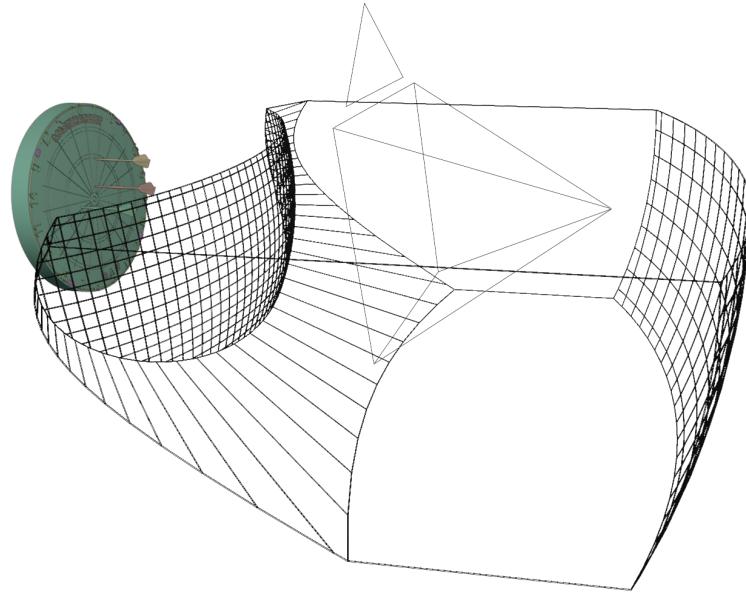


Abbildung 2.7: Darstellung des Kamerabereichs, der zur Erstellung der Daten verwendet wird. Zu sehen sind zusätzlich die Dartscheibe und die Kamera.

2.3.4.1 Kameraraum

Für die Positionierung der Kamera in der Szene existiert ein Objekt, das den Bereich umfasst, innerhalb dessen die Kamera platziert werden kann. Dieser kegelförmige Kameraraum ist ausgehend vom Mittelpunkt der Dartscheibe platziert und durch verschiedene Parameter definiert:

- **Horizontaler Seitenwinkel** ϕ_h : Öffnungswinkel des Kegels zu den Seiten der Dartscheibe,
- **Vertikaler Winkel** ϕ_v : Öffnungswinkel des Kegels in die Höhe,
- **Kameraabstand** (d_{\min}, d_{\max}): Minimaler und maximaler Abstand der Kamera von der Dartscheibe,
- **Kamerahöhe** (y_{\min}, y_{\max}): Minimale und maximale Höhe der Kamera im Raum⁴,
- **Maximaler Seitenabstand** dx_{\max} : Maximaler seitlicher Abstand der Kamera zum Dartscheibenmittelpunkt.

Die Generierung der Daten dieser Arbeit erfolgte mit den Parametern: $\phi_h = 110^\circ$, $\phi_v = 60^\circ$, $d_{\min} = 60 \text{ cm}$, $d_{\max} = 150 \text{ cm}$, $y_{\min} = 160 \text{ cm}$, $y_{\max} = 220 \text{ cm}$, $dx_{\max} = 60 \text{ cm}$. Der verwendete Kameraraum ist in Abbildung 2.7 dargestellt.

2.3.4.2 Brennweite

Das Setzen der internen Kameraparameter erfolgt nach der Positionierung der Kamera im Raum. So wird die Brennweite l_{Kamera} der Kamera in Abhängigkeit ihrer Distanz zur Dartscheibe gesetzt. Die Spanne der Brennweite reicht von 18 mm bis 60 mm. Die tatsächlichen Grenzwerte der Brennweiten in Abhängigkeit der Distanz werden wie folgt berechnet:

$$l_{\text{lower}} = l_{\min} + \frac{d_{\text{Kamera}}}{d_{\max} - d_{\min}} * \frac{l_{\max} - l_{\min}}{2}$$

$$l_{\text{upper}} = \frac{l_{\max} - l_{\min}}{2} + \frac{d_{\text{Kamera}}}{d_{\max} - d_{\min}} * \frac{l_{\max} - l_{\min}}{2}$$

Visualisiert sind diese Gleichungen in Abbildung 2.8. Werte zwischen Ober- und Untergrenze werden zufällig uniform gewählt. Auf diese Weise wird eine Korrelation zwischen verwendeter Brennweite und Abstand zur Dartscheibe gewonnen unter Beibehaltung von Variabilität der Daten und Einfluss von Zufallswerten.

⁴Es wird von einer standardisierten Anbringungshöhe der Dartscheibe von 2,07 m ausgegangen.

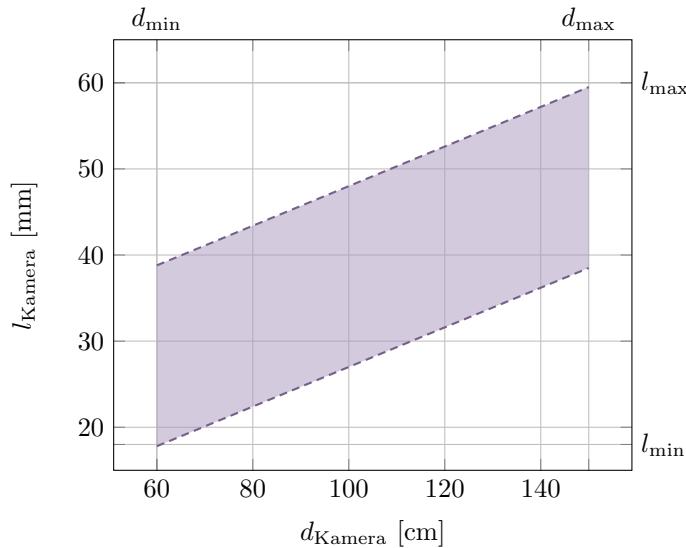


Abbildung 2.8: Abhängigkeit der Brennweite vom Abstand zur Dartscheibe. Ober- und Untergrenzen sind durch Strichlinien angegeben, der farblich hervorgehobene Bereich stellt die Spanne möglicher Brennweiten für jeweilige Entfernung dar.

2.3.4.3 Seitenverhältnis und Auflösung

Unterschiedliche Seitenverhältnisse der Kameraaufnahmen sind ebenfalls in dieser Arbeit berücksichtigt. So wird aus unterschiedlichen Seitenverhältnissen, die von Kameras in Mobiltelefonen verwendet werden, ausgewählt. Mögliche Seitenverhältnisse sind 4:3, 16:9, 1:1, 3:2, 2:1, 21:9 und 5:4. Die Ausrichtung der Kamera ist in $\frac{2}{3}$ der Aufnahmen vertikal und in $\frac{1}{3}$ der Aufnahmen horizontal. Die Auflösung der Kamera wird uniform im Intervall zwischen 1.000 px und 4.000 px gewählt, welches die Pixelzahl entlang der längeren Seite angibt.

2.3.4.4 Fokuspunkt

Der Fokuspunkt der Kamera liegt in der Szene auf einem spezifischen leeren Objekt, dessen Ursprung fokussiert wird. Dieses wird normalverteilt mit den Standardabweichungen $\sigma_x = \sigma_z = \frac{r_D}{3}$ und $\sigma_y = 2$ cm um den Dartscheibenmittelpunkt platziert. Die x - und z -Positionen liegen dabei auf der Dartscheibe, die y -Achse verläuft parallel zur Normalen der Dartscheibe und r_D ist der Gesamtdurchmesser der Dartscheibe. Der Kamerafokus ist damit grob auf den Mittelpunkt der Dartscheibe gerichtet.

2.3.4.5 Verwackelungen

Zuletzt werden verwackelte Kamerabilder simuliert. Mit einer Wahrscheinlichkeit von 10 % wird die Kamera während der Aufnahme bewegt, wodurch verschwommene Bilder aufgenommen werden. Die Kamera wird normalverteilt mit einer Standardabweichung von $\sigma = \frac{2\text{cm}}{3}$ in x -, y - und z -Position verschoben. Durch diese Verschiebung entstehen Aufnahmen, die teilweise verschwommen sind.

2.3.5 Render-Einstellungen

Zur Handhabung der Farbinformation bietet Blender eine Vielzahl unterschiedlicher Einstellungen. Die hohe Diversität gewünschter Erscheinungsbilder sorgt für viele Möglichkeiten zur Anpassung der Farbaufnahme der Kameras in den Szenen. Der für diese Thesis verwendete Farbraum ist derart gewählt, dass Farben möglichst realistisch dargestellt werden. Dazu wird als Darstellungsgerät und Sequencer sRGB mit AgX als Anzeigetransformation gewählt. Trotz häufiger Korrekturen von Handykameras hinsichtlich Kontrasterhöhung der Aufnahmen wird ein neutraler Basiskontrast verwendet. Eine Erhöhung des Kontrasts geschieht bei der Augmentierung der Trainingsdaten in Unterabschnitt 4.2.3.4.

2.3.6 Berechnung von Entzerrung

Die Erstellung der Entzerrungshomographie geschieht auf Grundlage exportierter Masken von dem Rendering. Eine der exportierten Masken zeigt die Orientierungspunkte, wie sie von DeepDarts verwendet werden [1]. Die Orientierungspunkte liegen auf der Außenseite des Double-Rings zwischen den Feldern 5 und 20 (oben), 13 und 6 (rechts), 17 und 3 (unten) und 8 und 11 (links). Durch die bekannten Positionen dieser Punkte im normalisierten ist die Berechnung der Verschiebungen dieser Punkte trivial und die Homographie zur Entzerrung des Bildes wird von diesen abgeleitet. Zur Erstellung dieser Maske befindet sich ein Objekt in der Szene, welches aus vier Punkten an den Positionen der Orientierungspunkte besteht. Dieses Objekt wird als Maske ausgehend von der ermittelten Kameraposition und den Kameraparametern gerendert. Aus dieser Maske lassen sich die Positionen durch Identifizierung von Mittelpunkten in Pixelclustern identifizieren und durch ihre Positionierung zueinander zu den jeweiligen Orientierungspunkten zuordnen.

2.4 Ergebnisse

Die Resultate der Datenerstellung sind neben den gerenderten Bildern ebenfalls die normalisierten Bilder sowie die in den Bildern enthaltenen Positionen. Diese Informationen umfassen neben den Positionen der Dartpfeile und ihre erzielten Punktzahlen weitere Metainformationen, die das Bild ausmachen. In einem ersten Schritt der Auswertung werden exemplarische Daten aufgezeigt. Anschließend wird ein Überblick über die Rahmenbedingungen der Datenerstellung gegeben, bevor mit einer qualitativen Auswertung der gerenderten Bilder fortgefahrene wird. Darauf folgen Einblicke in die Metainformationen der erstellten Daten. Anschließend wird auf die Korrektheit der Daten eingegangen und zuletzt werden Ungenauigkeiten bei der Erstellung der Daten aufgezeigt.

2.4.1 Beispiele gerenderter Bilder

In Abbildung 2.9 sind Bilder exemplarische Bilder dargestellt, die durch die vorgestellte Pipeline erstellt werden können. Die Auswahl der Bilder erfolgte durch subjektive Selektion exemplarischer Beispiele, die die Variation der Gesamtdaten einfangen. Eine Verzerrung der tatsächlichen Datenlage ist durch das Betrachten lediglich weniger Beispiele und die Art der Selektion nicht auszuschließen, jedoch wurde Acht gegeben, die Auswahl möglichst divers und repräsentativ für die Gesamtheit aller Daten zu halten.

Dargestellt sind sechs Bilder, die aus den für diese Thesis erstellten Trainingsdaten stammen. Diese Bilder zeigen die Spanne möglicher Bilder auf, die durch die Datenerstellung möglich sind. Die Variation der Kameraperspektiven ist in diesen Daten zu sehen, welche vollkommen unabhängig voneinander sind, im Vergleich zu den IEEE-Trainingsdaten von DeepDarts. Ebenfalls ist eine Variation der Hintergründe und Beleuchtungen zu erkennen. Die Wahl der Environment Maps, die den Hintergrund maßgeblich bestimmen, üben erheblichen Einfluss auf die Beleuchtung der Dartscheibe aus. Zusätzlich sind in zwei der dargestellten Bilder Ringlichter vorhanden, die darüber hinaus für eine Variation des Hintergrunds und der Beleuchtung sorgen. Die Dartscheiben der unterschiedlichen Bilder unterscheiden sich zusätzlich voneinander, wodurch eine große Spanne möglicher Dartscheiben simuliert werden kann. Hinsichtlich der Dartpfeile sind unterschiedliche Zusammensetzungen und Positionierungen in den Bildern vorhanden sowie variierende Anzahlen der Dartpfeile.

Die normalisierten Bilder befinden sich vertikal entlang der Mitte der Abbildung. Diese weisen allesamt die selbe Art der Entzerrung auf sowie die selben Abmessungen. Auffällig ist die Verzerrung von Dartpfeilen durch die Normalisierung der Dartscheiben: Je geringer der Winkel⁵ zwischen Dartscheibe und Kamera, desto stärker ist die Verzerrung der Dartpfeile durch den Effekt der Normalisierung.

⁵Ein großer Winkel steht für eine frontale Aufnahme der Dartscheibe während ein geringer Winkel eine stark seitliche Aufnahme der Dartscheibe zeigt.

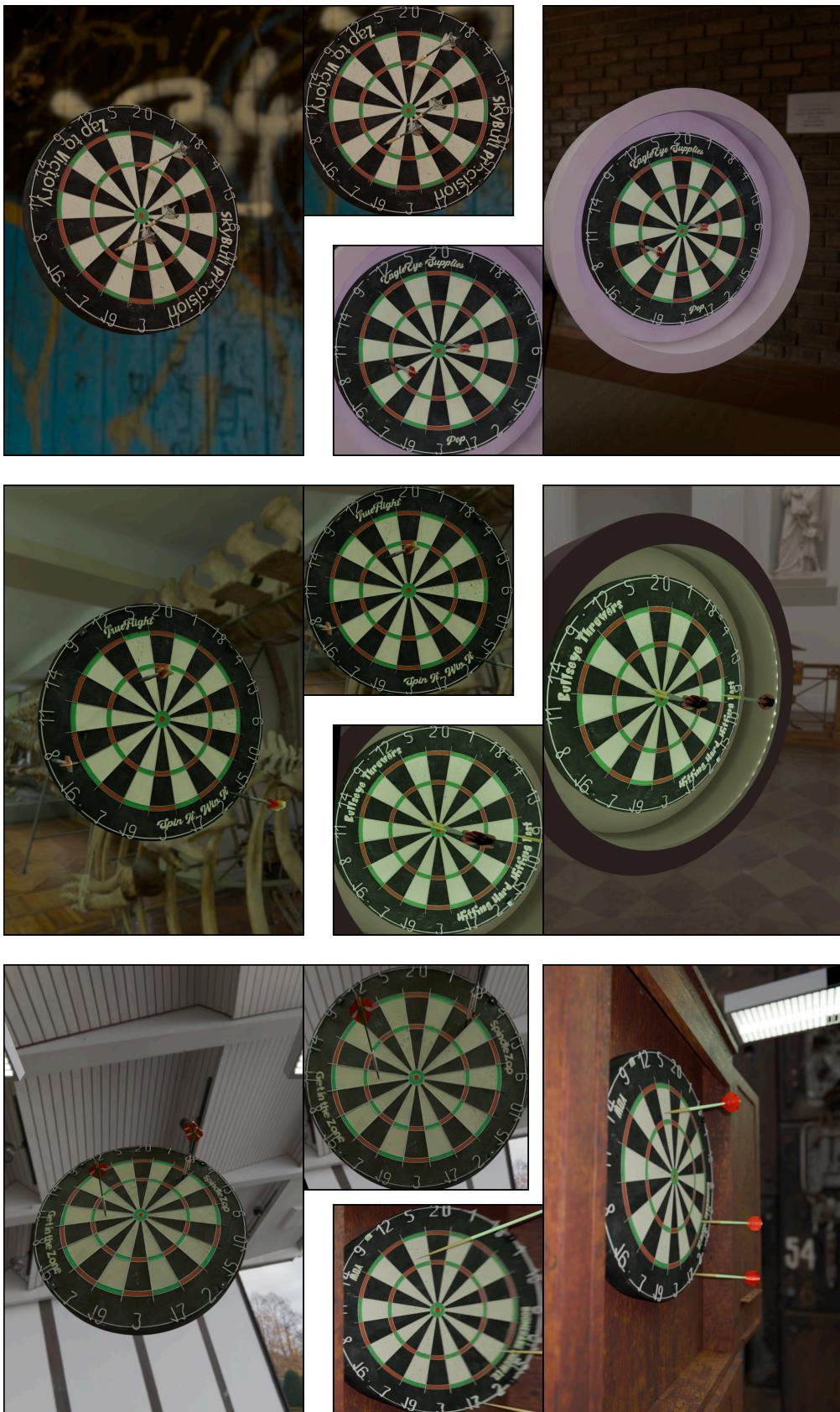


Abbildung 2.9: Gerenderte Bilder der Datenerstellung. Große Bilder an den Seiten sind Render-Outputs, kleine quadratische Bilder der mittleren Spalte sind dazugehörige normalisierte Bilder der Render.

2.4.2 Rahmenbedingungen der Erstellung

Die Datenerstellung wurde parallel auf mehreren Rechnern mit unterschiedlichen Grafikkarten ausgeführt. Als Mittelwert aller Daten wurde eine Erstellungszeit von 30 Sekunden je Sample ermittelt. Die Erstellung eines Samples beinhaltet das Einlesen der Szene, Setzen von Objekten und Parametern, Rendern des Output-Bildes sowie den Binärmasken und die Nachverarbeitung der Daten zur Anreicherung der Metadaten und Normalisierung für das Training neuronaler Netze. Insgesamt wurden 24.960 Trainingsdaten, 256 Validierungsdaten und 2.048 Testdaten erstellt. Die Speichernutzung je Sample beträgt durchschnittlich 8 MB, wobei die Datenmenge durch Entfernen von Masken, die nach der Erstellung der Daten nicht mehr verwendet werden, weiter reduziert werden kann. Insgesamt wurden für diese Arbeit Daten mit einem Umfang von etwa 216 GB erstellt. Das Blender-Projekt beläuft sich auf eine Größe von 220 MB, zu denen weitere 1,8 GB durch 208 Environment Maps hinzukommen, die extern dynamisch in die Blender-Szene eingebunden werden.

2.4.3 Qualitative Auswertung: Subjektiver Unterschied zwischen generierten und realen Bildern

Eine quantitative Auswertung der Bilddaten wird nicht vorgenommen. Die objektive Bewertung des Realismus von Bildern ist äußerst komplex und die Aussagekraft einer Metrik nicht eindeutig. Stattdessen wird eine qualitative Auswertung der Ergebnisse durchgeführt, bei der die Objektivität der Bewertung eine hohe Priorität hatte. Für die qualitative Analyse wird ein besonderer Fokus auf Merkmale gesetzt, die eine Unterscheidung realer und gerenderter Bilder ermöglichen. Die in Abbildung 2.9 dargestellten Render von Dartscheiben können als Referenz erstellter Bilder herangezogen werden, indem sie unter Vorbehalt von Verzerrungen einen Querschnitt möglicher generierter Daten darstellen.

Augenscheinlich ist den meisten Bildern eindeutig anzuerkennen, dass diese synthetisch erstellt sind und keine realen Dartscheiben abbilden. Ein zentraler Punkt dieser Beobachtung ist die Umgebung der Dartscheibe. Es wird sich bewusst gegen die Einbindung eines statischen Hintergrundes entschieden, um die unmittelbare Umgebung der Dartscheibe variabel zu halten. Diese Design-Entscheidung schlägt sich jedoch darin nieder, dass die Dartscheiben in Situationen dargestellt sind, die in realen Aufnahmen nicht existieren. Die Dartscheiben schweben in unüblichen Szenarien und ohne sichtliche Befestigung im Raum. Für den menschlichen Betrachter ist dieser Umstand sehr auffällig und sorgt für eine direkte Identifizierung als gerendertes Bild. Für die Einsatzbereiche der Normalisierung und des Trainings ist dieser Aspekt jedoch nicht von Relevanz. Die Semantik eines möglicherweise fehlenden thematischen Kontexts um die Dartscheibe liegt nicht im Interessenbereich der algorithmischen Normalisierung sowie des Trainings eines neuronalen Netzes. Für diese Systeme ist lediglich die Existenz variiender Informationen relevant, die strukturelle Ähnlichkeiten zu realen Daten aufweist. In Bildern, in denen unmittelbare Hintergründe der Dartscheibe beispielsweise durch Einblendung eines Dartschanks gegeben sind, wirken für den Betrachter erheblich realistischer. Beispiele von Bildern mit Hintergrundobjekten sind in Abbildung 2.9 in der rechten Spalte dargestellt. Insbesondere das unterste Bild wirkt durch die Existenz des Dartschanks weitaus realistischer als Bilder der linken Spalte.

Insgesamt ist jedoch trotz der erwähnten Schwachpunkte nach subjektivem Empfinden eine realistische Darstellung von Dartscheiben gelungen. Die Dartscheiben weisen realistische Gebrauchsspuren wie Risse, Verfärbungen, Einstichlöcher und Verformungen plastischer Elemente auf, die sich durch prozedurale Generierung in jeder Dartscheibe unterscheiden. Ebenfalls ist eine realistische Simulation möglicher Kameraparameter und -positionen abgedeckt, die eine zu erwartende Spanne von Parametern in realen Aufnahmen abdecken. Die Verteilung der Dartpfeile sowie ihre Orientierung bei Eintreffen auf der Dartscheibe wirken ebenfalls realistisch und kohärent zueinander.

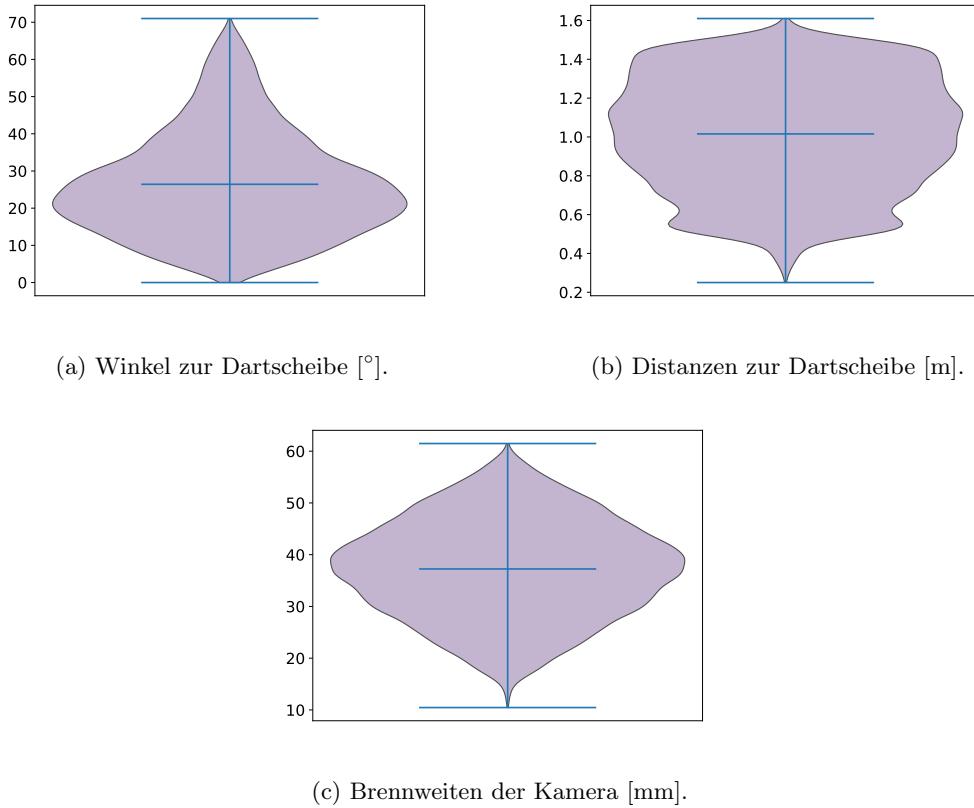


Abbildung 2.10: Verteilungen der Kameraparameter in den Trainingsdaten. Die Breite der Violinengraphen gibt relative Häufigkeiten an, Mittel- und Extremwerte sind dunkelblau gekennzeichnet.

2.4.4 Quantitative Metadatenauswertung

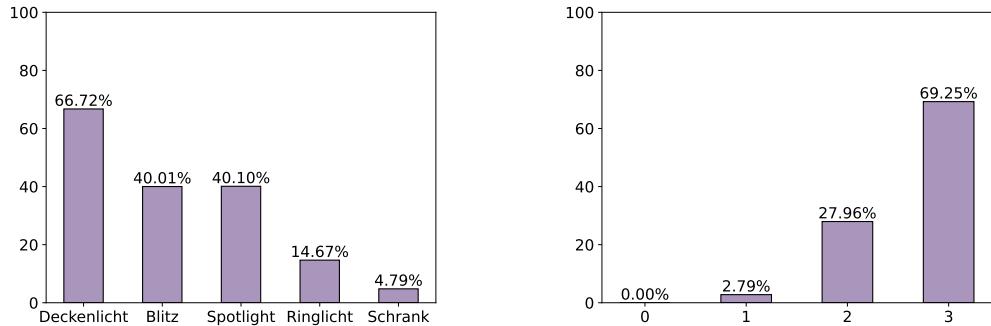
Eine quantitative Auswertung der Daten hinsichtlich ihres Erscheinungsbilds wird aus bereits genannten Gründen nicht vollzogen. Jedoch existieren vielerlei Metadaten, anhand derer die Durchführung einer quantitativen Auswertung möglich ist. Dieser Unterabschnitt liefert einen Einblick in die Aufstellung der generierten Daten und die Resultate der in Abschnitt 2.2 beschriebenen Techniken. Es wird begonnen mit einem Einblick in ausgewählte intrinsische und extrinsische Parameter der Kamera, gefolgt von einer Übersicht über die Beleuchtungen und dynamischen Objekten sowie die Anzahl der Dartpfeile je Bild. Zuletzt wird ein Blick auf die Verteilung der Dartpfeile über die Dartscheibe geworfen, indem die getroffenen Felder betrachtet werden.

2.4.4.1 Kamera-Auswertung

Die erste quantitative Auswertung dreht sich um ausgewählte und aussagekräftige Kameraparameter. Die Ergebnisse sind in Abbildung 2.10 in Form von Violinengraphen dargestellt. Die Graphen sind lediglich in ihrer y-Achse beschriftet, da die x-Achse relative Häufigkeiten ausdrückt.

Winkel zur Dartscheibe In Abbildung 2.10a ist der Winkel der Kamera zur Dartscheibe dargestellt. Zur Berechnung wird der Kamerawinkel ins Verhältnis zur Normalen der Dartscheibe gesetzt. Ein Winkel von 0° entspricht einer Frontalaufnahme der Dartscheibe während ein Winkel von 90° eine Aufnahme von der Seite der Dartscheibe darstellt. Es sind ausschließlich positive Werte der Winkel möglich.

Die häufigsten Winkel der Kamera befinden sich um 20° mit einem mittleren Winkel von $\approx 27^\circ$ und einem maximalen Winkel von $\approx 70^\circ$. Auffällig ist das verminderde Vorkommen frontaler Aufnahmen, die durch geringe Winkel charakterisiert sind.



(a) Relative Auftrittswahrscheinlichkeiten der Beleuchtungs-Objekte und dem Darts-Schrank.

(b) Relative Anzahl der Dartpfeile je gerendertem Bild.

Abbildung 2.11: Informationen zu Beleuchtung und Dartpfeilen. Abbildung 2.11a zeigt Beleuchtungs-Objekte in der Szene. Abbildung 2.11b zeigt die Anzahlen der Dartpfeile je Bild.

Distanz zur Dartscheibe Bei der Erstellung der Daten wird die Kamera zufällig in einem vordefinierten Kamerabereich platziert. Die resultierenden Distanzen der Kamera in den Daten sind in Abbildung 2.10b dargestellt. Es ist zu erkennen, dass ein geringes Aufkommen kurzer Distanzen unter 50 cm zu verzeichnen ist, da der Bereich, in dem sich die Kamera für derartige Distanzen befinden muss, verhältnismäßig klein ist. Die mittlere Distanz der Kamera beträgt $\approx 1\text{ m}$ und die maximale Distanz $\approx 160\text{ cm}$. Diese Verteilungen spiegeln nach subjektivem Empfinden Verteilungen wider, die in realen Szenarien zu erwarten sind. Aufnahmen geringeren Abstands sowie Aufnahmen mit größerer Entfernung als die in den Daten verzeichneten werden bereits bei der Parametrisierung des Kamerabereichs als unwahrscheinlich eingestuft und kategorisch ausgeschlossen, da die Dartscheibe in den Fällen unter Umständen nicht in einem Maße zu identifizieren ist, in welchem eine angemessene Verarbeitung der Daten zu erwarten ist.

Brennweite der Kamera Die Brennweiten der Kamera richten sich nach der in Abbildung 2.8 dargestellten Verteilung auf Grundlage der Kameradistanzen zur Dartscheibe. Die resultierenden Brennweiten in Abbildung 2.10c sind im Einklang mit einer zu erwartenden Verteilung indes sie approximativ die Breite der möglichen Distanzen je Brennweite darstellen, welche bei einer uniform verteilten Kameradistanz zu erwarten wäre. Ein Zuschnüren der geringwertigen Brennweiten durch die Unterrepräsentation geringer Distanzen, wie im vorherigen Abschnitt identifiziert, ist zu erkennen. Die minimale Brennweite in den Daten beträgt $\approx 10\text{ mm}$, die maximale Brennweite $\approx 60\text{ mm}$. Die mittlere Brennweite der Daten beträgt $\approx 38\text{ mm}$ und ist damit wertgleich zu dem oberen Brennweiten-Grenzwert minimaler Abstände sowie zu der unteren Grenze maximaler Abstände zur Dartscheibe.

2.4.4.2 Beleuchtung und Objekte in der Szene

In der Szene werden dynamische Objekte zufällig ein- und ausgeblendet, basierend auf unterschiedlichen Wahrscheinlichkeitsverteilung und Umgebungsbedingungen. Diese in den Daten aufgetretenen dynamischen Objekte sind in Abbildung 2.11a dargestellt. Hinsichtlich der Beleuchtungsobjekte ist das Deckenlicht mit $\approx 67\%$ am häufigsten vertreten. Diese Beobachtung beruht darauf, dass die Deckenleuchten als Rückfallbeleuchtung verwendet werden, sofern keine andere Lichtquelle vorhanden ist. Der Kamerablitz und das Spotlight sind in $\approx 40\%$ der Daten vorhanden. Ringlicht und Dartschrank sind Objekte, die sich gegenseitig ausschließen und daher nicht zusammen auftreten können. Ihre Existenz sorgt für wenig diverse Umgebungen um die Dartscheiben und wird daher gering gehalten. Mit $\approx 15\%$ Vorkommen des Ringlichts und $\approx 5\%$ Vorkommen des Dartschranks sind ihre Existenz von allen dynamischen Objekten am seltensten.

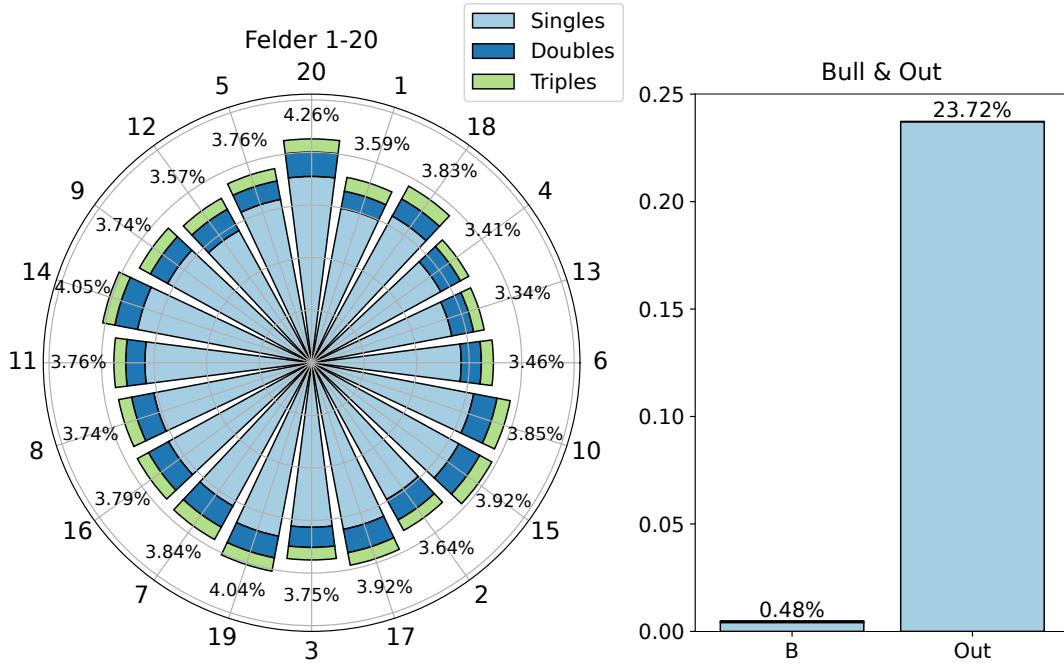


Abbildung 2.12: Relative Verteilung der Dartpfeile je Feld. Links: Felder 1-20; rechts: Bull und Outs.

Neben den dynamischen Objekten wird ebenfalls die Anzahl an Dartpfeilen in Abbildung 2.11b ausgewertet. Die Existenz jedes Dartpfeils wird während der Platzierung der Dartpfeile bestimmt und unterliegt ebenfalls einer vordefinierten Wahrscheinlichkeit. In $\approx 69\%$ der Daten sind alle drei Dartpfeile vorhanden, in $\approx 28\%$ sind zwei Dartpfeile auf der Dartscheibe verteilt und in $\approx 3\%$ aller Trainingsdaten ist lediglich ein Dartpfeil vorhanden. Es werden keine Bilder ohne Dartpfeile erstellt. Die Existenz von Dartpfeilen geht nicht automatisch mit einer Punktzahl > 0 einher, da Pfeile sowohl auf der Dartscheibe als auch außerhalb platziert werden können. Die Existenz bezieht sich lediglich auf das Vorhandensein von Dartpfeilen in den gerenderten Bildern.

2.4.4.3 Getroffene Felder

Zur Platzierung der Dartpfeile wird eine Heatmap verwendet, welche die zu erwartenden realistischen Verteilung der Dartpfeile auf der Dartscheibe nachempfunden ist. Zusätzlich werden spezifisch Daten erstellt, in denen die Pfeile ausschließlich auf die Bereiche der Double- und Triple-Felder und dem Bull sowie deren unmittelbare Umgebung fokussiert sind. In Abbildung 2.12 ist eine Auswertung der getroffenen Felder aller Pfeile in den Trainingsdaten visualisiert. Sie stellt die Verteilung aller in den Daten enthaltenen Pfeile dar. Die Grafik teilt sich auf in die Felder 1-20 (links) und Bull und Treffer außerhalb der Dartscheibe (rechts). Ausblendete Dartpfeile sind nicht in dieser Auswertung vertreten. Die Verteilung der Felder 1-20 entsprechen einer zu erwartenden Verteilung, in der Felder mit geringen Werten seltener getroffen sind als Felder mit hohen Zahlenwerten. Die Felder 20, 14 und 19 werden am häufigsten getroffen, was typische Spielweisen widerspiegelt⁶.

Das Bull ist in den Daten verhältnismäßig gering vertreten mit statistisch einem Treffer von 200 Dartpfeilen. Übermäßig häufig hingegen ist das Verfehlen der Dartscheibe vertreten mit etwa jedem vierten Dartpfeil. Der Bereich des Bulls und der Außenfläche der Dartscheibe sind jedoch analog sehr klein bzw. sehr groß im Verhältnis zu regulären Feldern. Daher sind diese Verteilungen hinsichtlich der Feldflächen nicht ungewöhnlich.

⁶Das Bespielen der 14 zieht bei Verfehlern höhere Zahlenwerte mit sich als das Verfehlen von 19 oder 20. Daher präferieren einige Spieler die 14 über 19 oder 20.

2.4.5 Korrekte Annotation der Daten

Bei der Erstellung von Daten werden die Positionen der Dartpfeile im 3D-Raum festgelegt und liegen für die Bestimmung der von den Dartpfeilen getroffenen Felder vor. Da Position sowie Transformation der Dartscheibe statisch sind, ist eine Rückrechnung der getroffenen Dartfelder sowie die korrekte Errechnung der erzielten Punktzahl fehlerfrei möglich. Dies spiegelt sich ebenfalls in den Daten wider: Die Dartpfeile sind allesamt korrekt annotiert und es existieren durch die in Unterabschnitt 2.3.3.1 erläuterten Methodiken des Umgangs mit ambivalenten Dartpfeilen keine Positionen, die auf mehrere Weisen gedeutet werden können. Anzumerken ist an dieser Stelle jedoch, dass eine Kollision der Dartpfeile mit der Spinne durch ihre Transformation als Resultat der Abnutzungssimulation nicht ausgeschlossen werden kann. Dieser Umstand tritt selten ein, jedoch wurden vereinzelte Daten mit dieser Anomalie identifiziert. Die Auftrittswahrscheinlichkeit dieses Umstands ist jedoch sehr gering und die Korrektheit der Annotation ist dadurch nicht beeinflusst⁷.

2.4.6 Ungenauigkeiten der Datenerstellung

Durch die Verwendung von 3D-Modellierung sind alle unterliegenden Informationen der Datenerstellung vorhanden und können zur korrekten Annotation der Daten verwendet werden. Trotz der Informationen über Kameraposition und -parameter sowie Dartpfeilpositionen geschieht die Bestimmung der Dartscheibenorientierung sowie die Lokalisierung von Dartpfeilpositionen im exportierten Bild nicht durch Berechnungen, sondern durch Nachverarbeitungsschritte. Diese gehen mit einem gewissen Grad Ungenauigkeit einher und sind resultierend nicht 100 % akkurat.

Alle Informationen hinsichtlich in- und extrinsischer Kameraparameter, Objektpositionen sowie Nachverarbeitungsschritten liegen während der Erstellung der Daten vor, jedoch ist die Verwendung dieser zur Rückrechnung der Positionen im gerenderten Bild sehr komplex. Stattdessen werden Positionen durch Überschneidungen von Objekten und Rendern dieser als Binärbilder mit den selben Exportparametern, die zur Erstellung des gerenderten Bildes verwendet wurden, exportiert. Durch Nachverarbeitungsschritte wie Clustering werden die dargestellten Positionen approximiert. Dieser Prozess unterliegt einem gewissen Grad der Ungenauigkeit, da mit diskretisierten Werten und Approximationen gearbeitet wird. Das Resultat dieser Erstellung ist eine minimale Variation der Orientierungspunkte hinsichtlich der Ausrichtung der Dartscheibe. Diese Abweichungen befinden sich in der Größenordnung weniger Pixel, jedoch ist diese Ungenauigkeit anzumerken. Eine Beeinträchtigung der Trainingserfolge durch diese Ungenauigkeiten wird jedoch durch die Anwendung starker Augmentierung überschattet (vgl. Unterabschnitt 4.2.3.4).

⁷Aufgrund der Komplexität einer algorithmischen Identifizierung dieser Kollisionen wird keine quantitative Auswertung über diesen Umstand vollzogen. Jedoch kann ein signifikanter struktureller Fehler der Annotation nach manueller Betrachtung einer Vielzahl erstellter Daten ausgeschlossen werden.

Kapitel 3

Vorverarbeitung von Bildern durch klassische Computer Vision

Sowohl für das Training als auch für die Inferenz werden normalisierte Bilder von dem neuronalen Netz erwartet. Diese Normalisierung kann auf unterschiedliche Arten geschehen, für DeepDarts wurde sich entschieden, die Normalisierung und Vorhersage der Dartpfeilpositionen in einem Schritt von dem neuronalen Netz vorhersagen zu lassen. Dieser Ansatz geht jedoch mit einigen Schwachpunkten einher. Wesentliche Nachteile werden offensichtlich in Hinsicht auf die Identifizierung spezifischer Fixpunkte, die möglicherweise verdeckt sein könnten, und Variabilität der Trainingsdaten. Das Angehen einer Aufgabe durch ein neuronales Netz stützt sich auf die Korrektheit und einen angemessenen Umfang der für das Training genutzten Daten. Durch gewollte oder ungewollte Einbindung einer verzerrten Datenlage erlernt ein neuronales Netz eben diese Eigenheiten und es besteht die Gefahr des Overfittings, sodass eine Verallgemeinerung der System-Performance auf unbekannte Daten nicht oder nur bedingt möglich ist.

Eine algorithmische Herangehensweise ist im Gegensatz zu einem neuronalen Netz keine Blackbox und die innere Arbeitsweise ist bekannt und klar definiert. Es kann strikt nachvollzogen werden, an welcher Stelle und aus welchem Grund eine fehlerhafte Vorhersage auftritt, und Problemquellen können gezielt angegangen werden. Darüber hinaus kann eine algorithmische Normalisierung von Daten ohne den Mehraufwand eines neuen Trainings eines neuronalen Netzes auf neue Daten erweitert werden. Zur Anpassung des Systems an neue Gegebenheiten sind lediglich wenige exemplarische Bilddaten notwendig, anhand derer charakteristische Eigenschaften abzuleiten sind und durch die die Arbeitsweise des Algorithmus angepasst werden kann.

Zusätzlich ist die zu lösende Aufgabe der Identifizierung von Dartscheiben durch ihre Geometrie und ihren Aufbau mit kontrastreichen und farblich markanten Feldern prädestiniert für eine algorithmische Verarbeitung. Diese Charakteristiken werden von Algorithmen und Techniken der herkömmlichen CV genutzt, um relevante Informationen zu extrahieren.

Aus diesen Gründen wird sich in dieser Thesis für eine algorithmische Normalisierung der Daten entschieden, die nach der Datenerstellung den zweiten Themenbereich ausmacht. In diesem Kapitel werden in einem ersten Schritt die für das Verständnis des Algorithmus notwendigen Grundlagen in Abschnitt 3.1 erläutert. Darauf folgend wird in Abschnitt 3.2 auf die Methodik der Normalisierung eingegangen und anschließend wird auf einige relevante Themen der Implementierungen eingegangen; Abschnitt 3.3. Zuletzt werden die Ergebnisse dieser Normalisierung anhand idealer Entzerrungen ausgewertet und mit dem Ansatz des DeepDarts-Systems verglichen.

3.1 Grundlagen

Bevor in die Thematik der Normalisierung eingegangen wird, ist die Klärung von Grundbegriffen und -konzepten relevant. Diese legen den Grundbaustein für die Nachvollziehbarkeit der Algorithmen und Techniken, die für die Verarbeitung der Bilddaten relevant sind. Diese Grundlagen setzen mathematische Kenntnisse voraus, wie sie in einem Studium der Informatik erlernt und verstanden werden. Die Grundlagen werden in einem Detailgrad erklärt, der ein Verständnis der eingesetzten Techniken ermöglicht.

3.1.1 Polarlinien

Die polare Darstellung von Linien ist für die Identifizierung der Dartscheibe dahingehend relevant, dass sie es ermöglicht, einer Linie einen Winkel zuzuordnen, in dem diese verläuft. Eine Polarlinie ist definiert als ein Tupel (ρ, θ) , in dem ρ der minimale Abstand der Linie zum Koordinatenursprung ist und θ der Winkel der Liniennormalen zur x-Achse. Die Charakterisierung einer Linie durch einen Winkel in Unterabschnitt 3.2.4 (Linienverarbeitung) verwendet.

Zur Umrechnung einer Linie, gegeben durch zwei Punkte $P_1 = (x_1, y_1)$ und $P_2 = (x_2, y_2)$, in Polarform werden folgende Gleichungen genutzt [42]:

$$\rho = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\theta = \arctan \frac{y_2 - y_1}{x_2 - x_1}$$

Festzuhalten ist, dass Start- und Endpunkte der Linie durch diese Art der polaren Beschreibung nicht berücksichtigt werden. Dies beruht auf der Gegebenheit, dass mathematische Beschreibungen von Linien infinite Längen haben. Diese Eigenschaft in dieser Darstellung von Polarlinien wird in der Methodik dieser Arbeit zur Identifizierung von Charakteristiken von Vorteil genutzt und wird gezielt in Unterabschnitt 3.2.4.2 eingesetzt.

3.1.2 Thresholding

Als Thresholding wird in der Datenverarbeitung die Einteilung von Werten eines Definitionsbereichs D in zwei Kategorien verstanden. Dabei wird ein Grenzwert $T \in D$, der Threshold, definiert, anhand dessen die Kategorie eines Wertes festgelegt wird. Üblich sind in der Bildverarbeitung Definitionsbereiche $D \in \{\mathbb{R}, \mathbb{N}\}$ und die Einteilung $v \leq T$ bzw. $v > T$ mit $v \in D$.

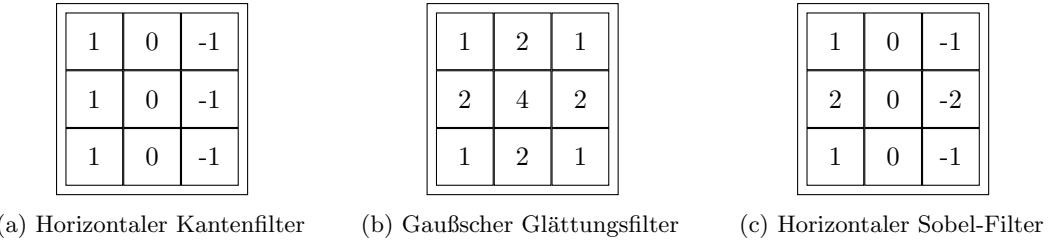
Thresholding findet seine Verwendung im Kontext dieser Masterarbeit in der Extraktion relevanter Informationen auf Bildern durch die Betrachtung von Pixeln innerhalb bestimmter Grenzwerte in Bildern. Diese Technik ermöglicht unter anderem das Identifizieren von Merkmalen anhand von Farben, die in bestimmten Farträumen besonders hervorgehoben sind und durch Thresholding eindeutig identifiziert werden können. Durch Kombination mehrerer Thresholds können komplexe Sachverhalte und Charakteristiken aus Bildern extrahiert werden.

In dieser Thesis wird Thresholding in vielerlei Hinsicht verwendet, unter anderem in Unterabschnitt 3.2.3.1 zur Differenzierung zwischen Kanten und Hintergrund oder in Unterabschnitt 3.2.4.3 zur Filterung von Linien anhand einer Abstandsmeetrik.

3.1.3 Binning

Binning ist als Erweiterung von Thresholding zu verstehen. Es bezeichnet die Diskretisierung von Werten in definierte Intervalle, sogenannte Bins oder Buckets. Durch Binning wird es ermöglicht, Spannen von Werten in definierte Bereiche zu unterteilen und somit in diskrete Kategorien einzufügen. Dabei wird zwischen Hard Binning und Soft Binning unterschieden.

Beim Hard Binning werden Intervallgrenzen $I = \{i_0, i_1, \dots, i_n\} \subseteq D$ definiert, die einen Definitionsbereich D von Daten in halboffene Intervalle $I_{k \in [0, n-1]} = [i_k, i_{k+1})$ unterteilen. Diese Intervalle korrespondieren mit Bins $B_{i \in [0, n]}$, in welche diejenigen Werte akkumuliert werden, die in das dementsprechende Intervall fallen. Es existieren harte Grenzen der Bins, die unter anderem dafür sorgen, dass nahe beieinander liegende Werte $v_0 \in D$ und $v_1 = v_0 - \epsilon > 0$

Abbildung 3.1: Unterschiedliche 2D-Kernel der Größe 3×3 .

in unterschiedlichen Bins zugeordnet werden, sofern $x_0 \in I$. Um dieses Artefakt zu umgehen, gibt es das Soft Binning, in dem Werte anteilig in Bins eingeordnet werden, sodass Werte im Umkreis um Intervallgrenzen in beide Bins einsortiert werden, gewichtet mit der Distanz zu der Intervallgrenze.

Binning wird in dieser Thesis unter anderem in Unterabschnitt 3.2.4.2 (Mittelpunktextraktion) genutzt, um Polarlinien anhand ihrer Winkel zu kategorisieren.

3.1.4 Faltung

Die Faltung, auch Convolution genannt, ist eine mathematische Operation, die ihren Ursprung in der Signalverarbeitung hat. Bei der Faltung werden Funktionen f und g kombiniert, um eine resultierende Funktion $h = (f * g)$ zu errechnen. Die Definition einer Faltung lautet [43, 37]:

$$(f * g)(x) = \int_{-\infty}^{-\infty} f(t)g(x-t)dt$$

Hinsichtlich der Bildverarbeitung wird eine diskrete 2D-Faltung genutzt, die die Extraktion von Merkmalen aus Bildern ermöglicht. Die diskrete 2D-Faltung funktioniert, indem ein Kernel auf jede Position eines Bildes angewandt wird. Mathematisch ist sie wie folgt beschrieben [44]:

$$I[x, y] = \sum_{i=0}^{x_k} \sum_{j=0}^{y_k} I[x - \lfloor \frac{x_k}{2} \rfloor + i - 1, y - \lfloor \frac{y_k}{2} \rfloor + j - 1] \times k[i, j]$$

Dabei ist $I \in \mathbb{N}^2$ ein Eingabebild mit einem Farbkanal, x und y sind Positionen im Bild in $k \in \mathbb{N}^2$ ein Kernel der Größe $x_k \times y_k$.

Diese allgemeine Formel lässt sich auf die Verwendung mehrerer Kanäle erweitern. Für eine Faltung eines Bildes mit c_0 Kanälen, einer Kernel-Größe von $k_x \times k_y$ und einer Ausgabe von c_1 Kanälen wird ein Kernel der Größe $c_0 \times k_x \times k_y \times c_1$ erstellt. Dabei werden für jeden Pixel in jedem Kanal alle Eingabekanäle betrachtet und gefiltert.

3.1.5 Kantenerkennung

Nachdem die Prinzipien der Faltung bekannt sind, wird in diesem Abschnitt auf die konkrete Anwendung der Faltung in der Bildverarbeitung eingegangen.

Anhängig von den Werten eines Kernels (auch Filter genannt) werden unterschiedliche Charakteristiken eines Bildes hervorgehoben. So können mit einem Kantenerkennungsfilter hochfrequente Bestandteile in einem Bild hervorgehoben werden während ein Bild mit mit einem Glättungsfilter weichgezeichnet wird und hochfrequente Anteile herausgefiltert werden. Der Sobel-Filter ist ein in der CV etablierter Filter zur robusten Kantenerkennung [45, 37]. Er kombiniert die Prinzipien der Glättung und Kantenerkennung, indem das Bild durch diesen in einer Richtung geglättet wird und in der anderen eine Kantenerkennung durchgeführt wird. Die Kantenerkennung ist durch diesen Filter robust hinsichtlich Rauschen im Bild. Exemplarische Filter für Kantenerkennung und Weichzeichnung sowie ein Sobel-Filter sind in Abbildung 3.1 dargestellt.

Filterung und Kantenerkennung werden in dieser Thesis ausgiebig genutzt. So wird in Unterabschnitt 3.2.3 Kantenerkennung als erster Schritt der Normalisierung angewandt und Filterung an sich ist in Kapitel 4 ein wichtiger Bestandteil von Schichten neuronaler Netze.

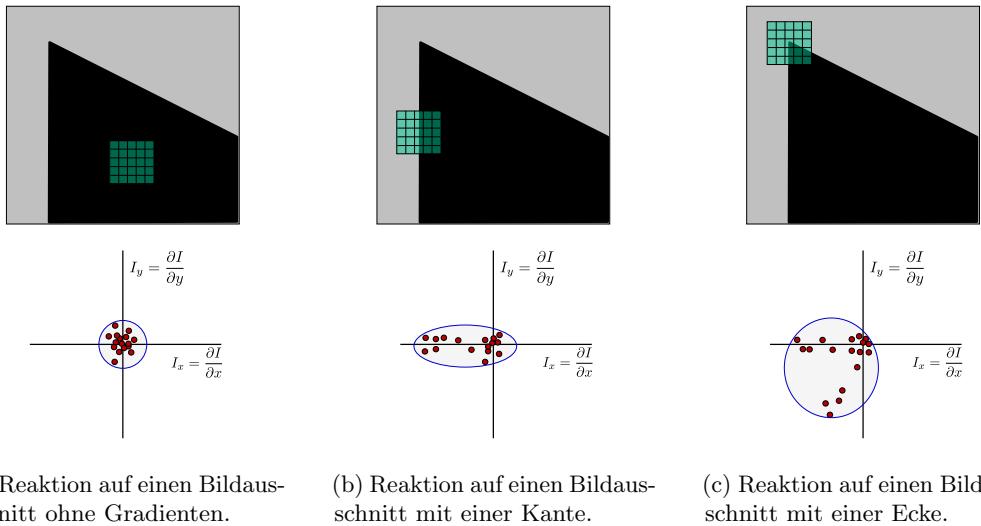


Abbildung 3.2: Visualisierung der Reaktionen von Kantenfiltern auf unterschiedliche Bildstrukturen und die Repräsentation ihrer Gradienten [47]. Die Haupt- und Nebenachsen sind in Abbildung 3.2a jeweils kurz, in Abbildung 3.2b ist die Hauptachse lang, während die Nebenachse kurz ist, und in Abbildung 3.2b sind sowohl Haupt- als auch Nebenachse lang.

3.1.6 Harris Corner Detection

Die Harris Corner Detection ist ein etablierter Algorithmus in der CV, der 1988 von C. Harris und M. Stephens vorgestellt wurde [46, 37]. Ziel des Algorithmus ist es, Ecken in einem Eingabebild zu identifizieren. Der Kerngedanke hinter der Harris Corner Detection basiert auf der Beobachtung, dass eine Ecke dadurch spezifiziert ist, dass sie der Endpunkt zweier aufeinandertreffender Kanten ist. Durch Kombination von Kanteninformationen und Thresholding werden Ecken in einem Bild identifiziert.

Als erster Schritt der Harris Corner Detection wird das Bild mit Kantenerkennungsfiltern verarbeitet, die, wie in Unterabschnitt 3.1.5 beschrieben, Kanten im Bild hervorheben. Diese Kantenerkennung erfolgt in horizontaler und vertikaler Richtung, woraus zwei Kantenreaktionen hervorgehen. Kantenreaktionen benachbarter Pixelregionen werden je in einem Koordinatensystem dargestellt, die die Magnitude der Kantenreaktionen von Pixeln in horizontaler und vertikaler Richtung darstellen. In diesen Koordinatensystemen werden umliegende Ellipsen um die resultierenden Cluster gelegt, wie in Abbildung 3.2 visualisiert.

Anhand der Exzentrizität und Größe einer Ellipse wird beurteilt, ob es sich bei einem Cluster von Pixeln um eine Fläche, Kante oder Ecke handelt. Benachbarte Pixel um einen Pixel auf einer Ecke werden dabei ebenfalls als Ecken identifiziert. Um diese zu unterdrücken, wird Non-Maximum-Suppression (NMS) verwendet, bei der lediglich diejenigen Pixel mit der stärksten Antwort auf die Eckenerkennung hervorgehoben werden. Alle benachbarten Pixel in einem vordefinierten Fenster um diesen maximalen Pixel werden entweder in ihrer Intensität gedämpft oder unterdrückt und auf Null gesetzt.

3.1.7 Hough-Transformation

Die Hough-Transformation, veröffentlicht 1962 von Paul Hough, ist ein Algorithmus zur Identifizierung von simplen Strukturen in Kantenbildern [48, 48]. In diesem Abschnitt wird die Erkennung von Linien mit der Hough-Transformation beschrieben, wie sie in Unterabschnitt 3.2.4.1 (Linienerkennung) verwendet wird.

Hauptaspekt des Algorithmus ist die Transformation von Punkten im Image-Space zu Linien im Parameter-Space. Der Image-Space ist das Koordinatensystem des Bildes mit den Achsen x und y ; der Parameter-Space ist ein Koordinatensystem mit den Achsen ρ und θ . Linien im Image-Space sind mathematisch beschrieben als Geraden der polaren Form:

$$0 = x \sin \theta - y \cos \theta + \rho$$

Dabei sind ρ und θ – wie in Unterabschnitt 3.1.1 eingeführt – Winkel und Abstand einer Linie zum Koordinatenursprung, welcher in Bildern in der oberen linken Ecke liegt. Ein Punkt (x, y) im Image-Space ist im Parameter-Space als Sinuswelle dargestellt, welche alle Linien beschreibt, die durch den Punkt (x, y) im Image-Space verlaufen. Ein Punkt (ρ, θ) im Parameter-Space beschreibt eine Linie in Image-Space.

Eingabedaten der Hough-Transformation sind durch Thresholding vorverarbeitete Bilder, in denen typischerweise Kanten oder Ecken extrahiert sind. Die extrahierten Pixel werden im Parameter-Space akkumuliert, in dem sie sich in Form von Sinuswellen überschneiden. Liegen demnach Punkten im Image-Space in einer Linie, so überschneiden sich ihre korrespondierende Sinus-Darstellungen im Parameter-Space in einem Punkt. Dieser Punkt liegt an den Koordinaten (ρ, θ) und beschreibt die Parameter der Geraden, die durch die Punkte verläuft. Das Identifizieren von Peaks im Parameter-Space führt analog zur Identifizierung von Linien im Image-Space.

3.1.8 Transformationsmatrizen

Transformationen von Bildern in der CV basieren auf der Grundlage von Transformationsmatrizen [49, 50, 37]. Die unterschiedlichen Arten von Transformationsmatrizen und ihre Arbeitsweisen werden in den folgenden Unterabschnitten erläutert. Es wird dabei begonnen mit Grundlagen homogener Koordinaten, gefolgt von unterschiedlichen Arten von Transformationsmatrizen.

3.1.8.1 Homogene Koordinaten

Punkte im 2D-Raum besitzen eine x - und eine y -Koordinate, sodass ein Punkt definiert ist durch $P = (x, y)$. Eine Erweiterung dieser Koordinatendarstellung sind homogene Koordinaten. Um einen 2D-Punkt P in einen homogenen Punkt $\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z})$ umzuwandeln, wird eine Koordinate $\tilde{z} \neq 0$ hinzugefügt, die zur Normalisierung der Koordinaten genutzt wird [37]. Zur Umwandlung von P in homogene Koordinaten wird $z = 1$ gesetzt; die Rücktransformation geschieht durch $P = (\frac{\tilde{x}}{\tilde{z}}, \frac{\tilde{y}}{\tilde{z}})$. Homogene Koordinaten sind eine dreidimensionale Einbettung des zweidimensionalen Raumes und ermöglichen Transformationen von Koordinaten und als Erweiterung dessen auch die Transformation von Bildern durch Transformationsmatrizen M . Diese sind 3×3 Matrizen, die durch Multiplikation mit homogenen Punkten Transformationen auf diese anwenden:

$$\tilde{P}' = M \times \tilde{P}$$

Die unterschiedlichen Einträge der Transformationsmatrizen bestimmen die Art der Transformation. Durch Verkettung von Transformationsmatrizen können mehrere Transformationen in einer einzelnen Transformation zusammengefasst werden. Dabei ist die rechts-Assoziativität der Matrixmultiplikation zu beachten, durch die eine Anwendung der Matrizen von rechts nach links geschieht. Darüber hinaus ist die Kommutativität bei Matrixmultiplikationen nicht gegeben, sodass die Reihenfolge der Anwendungen relevant ist:

$$\begin{aligned}\tilde{P}' &= M_n \times \cdots \times M_2 \times M_1 \times \tilde{P} \\ &= (M_n \times \cdots \times M_2 \times M_1) \times \tilde{P} \\ &= M_{n,\dots,2,1} \times \tilde{P}\end{aligned}$$

Die Anwendung der Matrix $M_{n,\dots,2,1}$ auf ein Bild hat den selben Effekt wie die aufeinanderfolgende Anwendung der Matrizen M_1 bis M_n .

3.1.8.2 Affine Transformationsmatrizen

Affine Transformationen zeichnen sich durch die Aufrechterhaltung von Punkten, geraden Linien und Flächen aus. Nach einer affinen Transformation verbleiben parallele Linien weiterhin parallel, Winkel zwischen Geraden können sich jedoch ändern [37]. Es gibt unterschiedliche Arten affiner Transformationen, die in diesem Unterabschnitt vorgestellt werden.

Translationsmatrix Die Translationsmatrix verschiebt Punkte um gegebene Distanzen. x - und y -Verschiebungen werden mit t_x und t_y beschrieben und sind wie folgt aufgebaut:

$$\begin{aligned} M_{\text{trans}} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \end{aligned}$$

Skalierungsmatrix Die Skalierungsmatrix skaliert ein Bild, unterteilt in eine horizontale Skalierung s_x und eine vertikale Skalierung s_y . Skalierungen < 1 resultieren in einer Stauchung, Skalierungen > 1 in Streckungen. Skalierungsmatrizen sind wie folgt aufgebaut:

$$\begin{aligned} M_{\text{scl}} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} s_x \cdot x \\ s_y \cdot y \\ 1 \end{bmatrix} \end{aligned}$$

Scherungsmatrix Bei der Scherung werden Punkte parallel zur x -Achse mit a_x und parallel zur y -Achse mit a_y geschert; der neutrale Wert einer Scherung beträgt 0. Scherungsmatrizen haben die Form:

$$\begin{aligned} M_{\text{shr}} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} 1 & a_x & 0 \\ a_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} x + a_x \cdot y \\ a_y \cdot x + y \\ 1 \end{bmatrix} \end{aligned}$$

Rotation Die Rotation erfolgt anhand eines Winkels α , der Punkte in mathematisch positiver Richtung um den Koordinatenursprung rotiert. Die Zusammensetzung einer Rotationsmatrix kann aufgeteilt werden in einen Skalierungs- und einen Scherungsanteil. Die Skalierung erfolgt uniform in horizontaler und vertikaler Richtung um den Faktor $s_{x,y} = \cos(\alpha)$; die Anteile der Scherung sind gegeben mit $a_x = -\sin(\alpha)$ und $a_y = \sin(\alpha)$. Aus diesen Voraussetzungen ergibt sich die folgende Definition der Rotationsmatrix:

$$\begin{aligned} M_{\text{rot}} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos(\alpha) \cdot x - \sin(\alpha) \cdot y \\ \sin(\alpha) \cdot x + \cos(\alpha) \cdot y \\ 1 \end{bmatrix} \end{aligned}$$

3.1.8.3 Homographien

Im Gegensatz zu affinen Transformationen können Matrixeinträge in Homographien beliebig sein, sodass eine allgemeine Homographie die folgende Form besitzt:

$$H = \begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} \\ h_{1,0} & h_{1,1} & h_{1,2} \\ h_{2,0} & h_{2,1} & h_{2,2} \end{bmatrix}$$

Durch Fixierung der Skalierung mit $\sqrt{\sum_{ij} h_{ij}^2} = 1$ sind in einer allgemeinen Homographie H 8 freie Parameter vorhanden. Diese können z. B. durch vier Punktverschiebungen mit je zwei Koordinaten gegeben sein. Dadurch ist eine beliebige Transformation eines Bildes ermöglicht, in dem vier Punkte eines Quellbildes auf vier Punkte eines Zielbildes transformiert werden. Diese Eigenschaft wird bei der Normalisierung der Dartscheibe in Unterabschnitt 3.2.5.4 (Entzerrung der Dartscheibe) genutzt, um Orientierungspunkte der Eingabebilder auf bekannte Positionen in normalisierten Bildern zu transformieren.

3.1.9 Log-polare Entzerrung

Die log-polare Darstellung eines Bildes wird durch eine Transformation des Koordinatensystems erlangt. Während Koordinaten in einem kartesischen Koordinatensystem durch x - und y -Koordinaten angegeben sind, werden Punkte im log-polaren Koordinatensystem durch ein Tupel (ρ, θ) beschrieben. Sie sind definiert als logarithmischer Abstand und Winkel zu einem spezifischen Punkt (c_x, c_y) [51]. Die Umwandlung der Koordinaten ist definiert durch:

$$\begin{aligned} \rho(x, y) &= \ln \sqrt{(x - c_x)^2 + (y - c_y)^2} \\ \theta(x, y) &= \arctan2((y - c_y), (x - c_x)) \end{aligned}$$

In dieser Thesis wird die log-polare Darstellung eines Bildes in Unterabschnitt 3.2.5.1 genutzt, um Dartscheiben um ihren Mittelpunkt abzuwickeln. Der Effekt dieser Entzerrung ist die Transformation der Dartfelder von Kreisabschnitten zu Rechtecken.

3.1.10 Farbräume

Farbinformationen in Bildern werden auf unterschiedliche Arten gespeichert. Diese unterschiedlichen Arten der Darstellungen von Farben werden als Farbräume bezeichnet [52, 37]. Zu den am weitesten verbreiteten Farbräumen zählen unter anderem der RGB- und der HSV-Farbraum. Im RGB-Farbraum werden Farbinformationen nach dem Vorbild des menschlichen Auges in designierten Kanälen für rote, grüne und blaue Bestandteile des Bildes unterteilt; im HSV-Farbraum stehen die Kanäle H, S und V für Färbung („hue“), Sättigung („saturation“) und Helligkeit („value“). Visuell näherte Verbundenheit zwischen der menschlichen Farbwahrnehmung wird durch die Farbräume YCbCr und Lab erzielt. Diese Farbräume nutzen ebenfalls drei Farbkanäle, die jedoch abstrakter gestaltet sind und Farbinformationen auf eine Art encodieren, dass eine Interpolation von Farben zueinander mit visuellem Einklang ermöglichen, der in RGB und HSV nicht trivial erzielbar ist.

Die Verwendung unterschiedlicher Farbräume steht mit verschiedenen Vor- und Nachteilen in Verbindung. In dieser Arbeit werden die aufgezählten Farbräume zur Identifizierung von Farben und Hervorhebung von Merkmalen in Bildern genutzt. Die Verwendung von Farbraumtransformationen in dieser Thesis geschieht insbesondere in Unterabschnitt 3.2.5.2 und Unterabschnitt 3.3.2.

3.1.11 Structural Similarity

Für die Aufgabe der Ähnlichkeitsbestimmung von Bildern wurde 2004 von Wang et al. die Structural Similarity (SSIM) entwickelt, die über das triviale Vergleichen von Farbinformationen hinaus geht [53, 37]. Das Grundprinzip der SSIM-Metrik ist das Einfangen und Vergleichen visuell auffälliger Änderungen im Bild und Unterdrückung von Effekten wie Weichzeichnung und lokaler Verschiebungen von Pixeln. Es werden Informationen zu Luminanz, Kontrast und Struktur auf Eingabebildern identifiziert und jeweils miteinander verglichen. Eine gewichtetes Produkt der jeweiligen Ähnlichkeitsfaktoren bestimmt schließlich den SSIM-Wert.

Diese Handhabung der Informationsverarbeitung sorgt für Robustheit gegenüber geringen augenscheinlichen Änderungen, die jedoch starke Auswirkungen auf die zugrundeliegenden Daten besitzen. So ist das Weichzeichnen eines Bildes oder das Hinzufügen von Rauschen eine solche Änderung, bei der die Pixelwerte der Bilder stark verändert werden, die Aussage hinter den Daten jedoch nicht.

3.1.12 RANSAC

Bei der Verarbeitung von Daten ist nicht in jedem Fall davon auszugehen, dass alle vorhandenen Datenpunkte Teil relevante Informationen halten. Outlier sind Datenpunkte, die nicht der zu erwartenden oder gewünschten Aussage der Daten folgen, sondern als fehlerhafte Aufnahmen in den Daten vorhanden sind. Triviale Least-Squares-Approximationen unter Einbezug aller Datenpunkte ist für diese Anomalien in Daten anfällig, wodurch die Ergebnisse dieser Methoden beeinflusst werden. Um mit dieser Art der Anomalien umzugehen, stellten Fischler und Bolles 1981 den RANSAC-Algorithmus vor [54, 37].

RANSAC steht für „Random Sample Consensus“ und beruht auf dem zufälligen Auswählen eines Subsets von Datenpunkten zur Approximation einer akkurate Beschreibung der Daten. Nach dem Auswählen zufälliger Punkte und Approximieren einer Zielverteilung wird jeder Punkt anhand von Grenzbedingungen als Inlier oder Outlier klassifiziert. Das Verhältnis von Inliern zu Outliern wird als Metrik zur Bewertung der Approximation genutzt. Durch wiederholtes Auswählen zufälliger Datenpunkte und Klassifizierung ist eine Identifizierung einer Approximation möglich, auf die Outlier in den Datenpunkten wenig Einfluss nehmen.

In dieser Arbeit wird dieser Ansatz in Unterabschnitt 3.2.5.4 (Entzerrung der Dartscheibe) genutzt, um eine Normalisierung der Dartscheibe auf Grundlage von Orientierungspunkten trotz möglicher Outlier robust zu identifizieren.

3.2 Methodik

In diesem Unterkapitel wird die Methodik des Algorithmus zur Lokalisierung und Normalisierung von Dartscheiben in Bildern beliebiger Dimensionen beschrieben. Dieses Kapitel ist dazu in weitere Unterkapitel unterteilt, in denen thematische Abschnitte des Algorithmus auf fortlaufend abstrakteren Daten beschrieben werden. Bevor jedoch in mit der Beschreibung der Arbeitsweise begonnen wird, wird in Unterabschnitt 3.2.1 die Frage der Notwendigkeit von der Verwendung herkömmlicher CV im Gegensatz zur Verwendung neuronaler Netze für diese Aufgabe geklärt. Danach wird mit Unterabschnitt 3.2.2 die Vorverarbeitung der Bilder für den Algorithmus beschrieben. Darauf folgen die Schritte der Kantenerkennung in Unterabschnitt 3.2.3, in welcher die relevanten Informationen des Bildes extrahiert werden. Nach der Kantenerkennung folgt die Linienverarbeitung in Unterabschnitt 3.2.4, in der die Kanteninformationen zu Linieninformationen überführt und weiter verarbeitet werden. Anschließend wird in Unterabschnitt 3.2.5 der Schritt der Orientierung beschrieben, in welchem anhand von bekannten Punkten eine Entzerrung der Dartscheibe errechnet wird. Abgeschlossen wird das Kapitel der Methodik mit der Zusammenführung aller Komponenten in Unterabschnitt 3.2.6.

Ein Überblick über die Schritte des Algorithmus ist in Abbildung 3.3 in Form eines Flussdiagramms dargestellt. Es sind die verschiedenen Stufen der Verarbeitung zu erkennen, welche in diesem Abschnitt genauer erläutert werden.

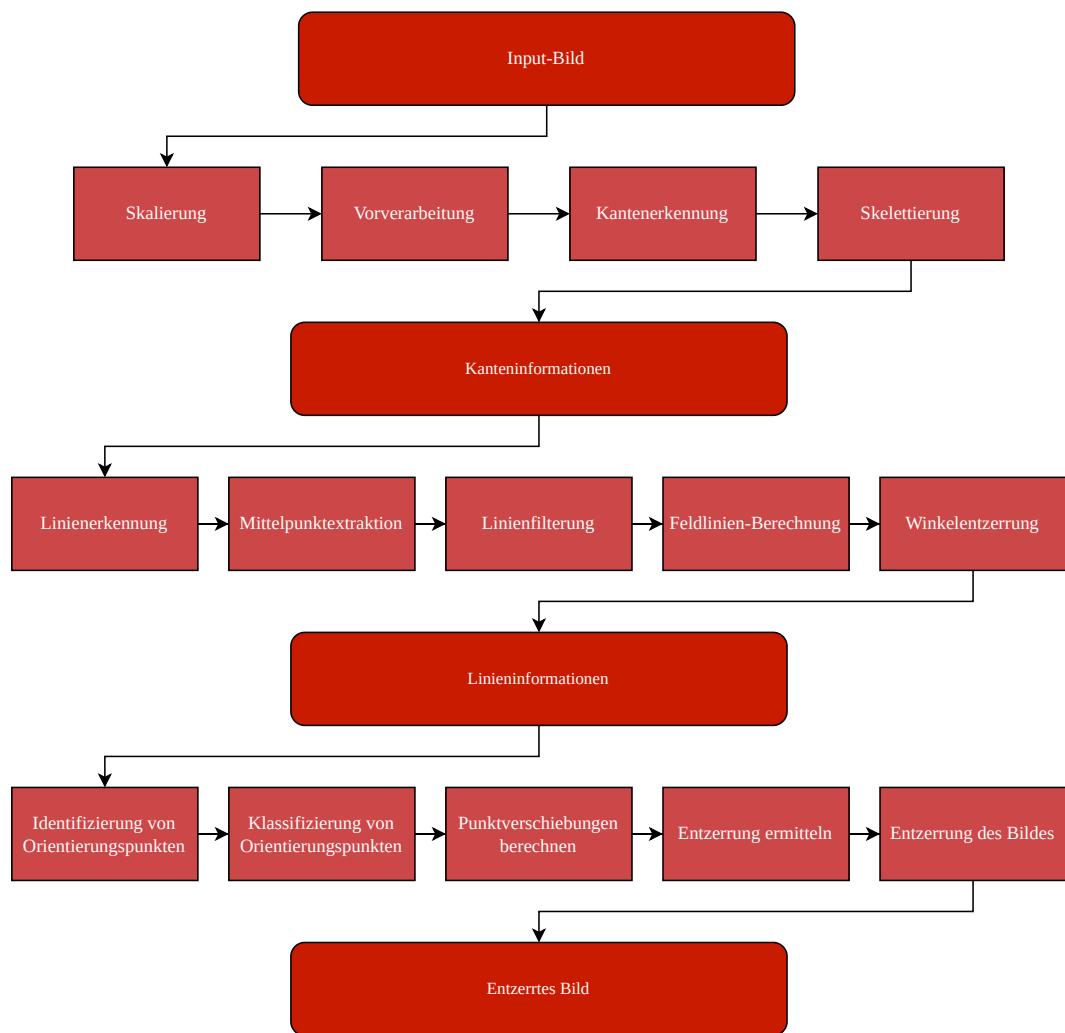


Abbildung 3.3: Schematische Darstellung der CV-Pipeline.

3.2.1 Motivation der Verwendung herkömmlicher Computer Vision

In dieser Arbeit wird eine strikte Trennung von Normalisierung der Bilddaten und Lokalisierung von Dartpfeilen auf normalisierten Dartscheiben vorgenommen. In der Herangehensweise von McNally et al. wird eine Netzwerkarchitektur verwendet, die unabhängig von der Eingabegröße der Bilddaten Aufgaben erzeugt, indem ein Fully Convolutional Neural Network (FCNN) verwendet wird. Die Identifizierung der Geometrie der Dartscheibe und Einstichpunkte der Dartpfeile werden in einem gemeinsamen Durchlauf ermittelt und die erzielte Punktzahl wird aus diesen ermittelt. In dieser Thesis ist dieser Prozess aufgeteilt in Normalisierung und Lokalisierung, um normalisierte Bilddaten als Eingaben in das neuronale Netz zur Vorhersage der Dartpfeilspitzen zu verwenden. Durch diese Bedingung an Eingabedaten kann die Spanne möglicher Eingabedaten reduziert werden und das Training zielgerichteter ablaufen.

Ein wesentlicher Vorteil der Auslagerung der Dartscheibenfindung in einen algorithmischen Vorverarbeitungsschritt ist neben der thematischen Kapselung der Herangehensweise das Verständnis des Systems. Während neuronale Netze in der Lage sind, beliebige Funktionen zu approximieren, ist ihre Arbeitsweise nicht bekannt und folglich nicht wartbar. Bei einer algorithmischen Lösung ist die Arbeitsweise bekannt und es kann im Fehlerfall nachvollzogen werden, aus welchen Gründen eine Ausführung fehlgeschlagen ist. Dadurch ist eine gezielte Wartung und Erweiterung eines Algorithmus ohne erneutes Training möglich.

Ebenfalls war die Verwendung von CV aufgrund der auffälligen Geometrie einer Dartscheibe naheliegend, da sie Ähnlichkeiten mit Schachbrettern aufweist, welche in der CV zur Identifizierung von Kameraparametern verwendet werden [37]. Da die Nutzung bekannter Geometrien eine zentrale Arbeitsweise der CV darstellt, war die Intuition gegeben, dass auch eine Erkennung eines ähnlich markanten Objekts – konkret: einer Dartscheibe – in einem Bild möglich ist. Aus dieser Intuition heraus ist der in diesem Abschnitt beschriebene Algorithmus entwickelt worden.

3.2.2 Vorverarbeitung

Die Algorithmen der CV arbeiten auf Bildern beliebiger Größe. Da die Dauer der Verarbeitung mit der Größe der Eingabebilder skaliert, ist eine angemessene Skalierung der Eingaben ein relevanter Bestandteil der Laufzeitoptimierung. Damit einher geht jedoch der Verlust von Informationen im Bild, wodurch eine Abwägung zwischen Geschwindigkeit und Genauigkeit notwendig ist. In dieser Arbeit wird eine schrittweise Verkleinerung der Eingabebilder mit Abmessungen (w, h) , entsprechend Breite und Höhe, unternommen, bis $\max(w, h) < s_{\max}$. Dabei werden Eingabebilder jeweils um den Faktor 2 verkleinert, um Artefakte durch Interpolation zu minimieren. Der Wert von $s_{\max} = 1.600 \text{ px}$ ist heuristisch ermittelt als geeignetes Mittel zwischen Geschwindigkeit und Genauigkeit.

Der Schritt der Vorverarbeitung kann übersprungen werden, indem $s_{\max} = \infty$ gesetzt wird. Die Laufzeit der Normalisierung kann dadurch jedoch stark beeinträchtigt werden, da die Anzahl der Pixel annähernd quadratisch mit der Größe des Bildes skaliert.

3.2.3 Kantenverarbeitung

Nachdem die Eingabebilder vorverarbeitet sind, werden die wichtigen Kanten im Bild extrahiert. Eingabebilder enthalten neben den für die Normalisierung relevanten Informationen der Dartscheibe Rauschen, das nicht für die Normalisierung notwendig ist. Mit der Kantenverarbeitung wird der Umfang an Informationen stark reduziert und auf die wichtigen Charakteristiken des Bildes limitiert.

Die Schritte der Kantenerkennung sind in Abbildung 3.4 dargestellt. Das verwendete Bild stammt aus dem für DeepDarts verwendeten Datensatz und wird ebenfalls im Paper des Systems zur Veranschaulichung von dessen Arbeitsweise genutzt. Im Sinne der Vergleichbarkeit der Systeme wird die Arbeitsweise dieses Algorithmus anhand des selben Bildes veranschaulicht.

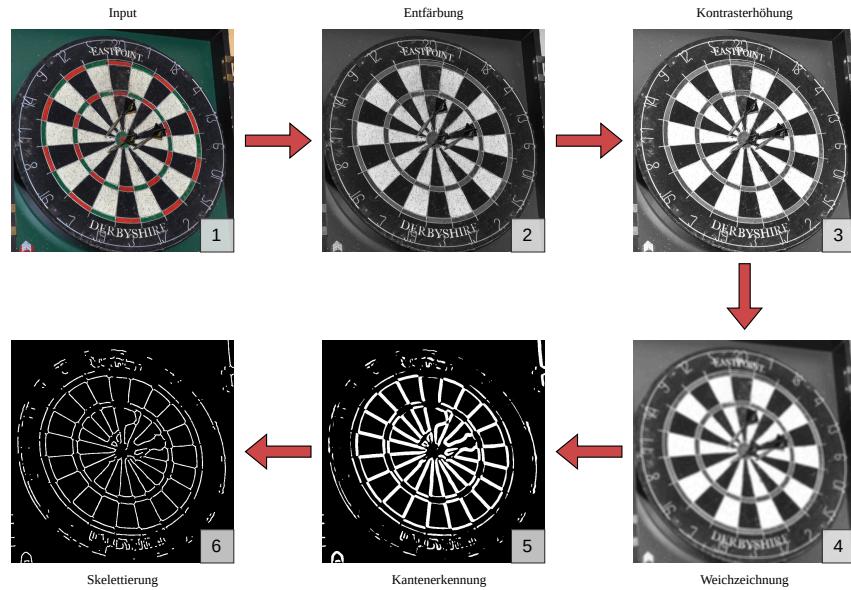


Abbildung 3.4: Schritte der Kantenverarbeitung. (1) Input-Bild aus dem DeepDarts-Datensatz [4]. (2) Umwandlung des Bildes in Graustufen. (3) Kontrasterhöhung des Bildes zur Hervorhebung der Unterschiede schwarzer und weißer Felder. (4) Weichzeichnung zur Verminderung von Störungen. (5) Filterung durch Sobel-Filter, gefolgt von Thresholding. (6) Skelettiertes Kantenbild.

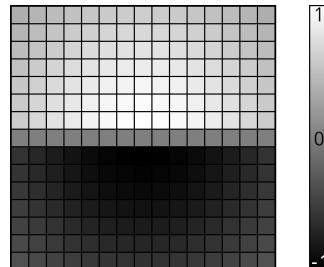


Abbildung 3.5: Vertikaler Sobel-Kernel der Größe 15×15 zur Identifizierung großer und ununiformer Kanten in einem Bild. Helle Pixel stehen für positive, dunkle Pixel für negative Werte.

3.2.3.1 Filterung

Für eine universelle Extraktion von Kanten in Bildern existieren Algorithmen und Filter, wie sie bereits in Unterabschnitt 3.1.5 beschrieben sind. Diese Filter sind für allgemeine Fälle geeignet, in denen das Ziel eine generelle Kantenerkennung ist oder wenig Annahmen über die Kanteninformationen in Eingabebildern getroffen werden können. In dem hier betrachteten Fall liegt der Fokus der Kantenerkennung nicht auf generischen Kanten im Bild, sondern spezifisch auf den Kanten zwischen den Flächen der Dartscheibe. Diese sind charakteristisch für die Dartscheibe und durch ihr festgelegtes Design vorgegeben. Durch die Erkennung dieser Kanten wird darauf abgezielt, den Mittelpunkt und die grobe Orientierung der Dartscheibe zu ermitteln.

Geometrie und Farbgebung der Felder einer Dartscheibe sorgen für starke Gradienten der Pixelintensitäten entlang der Kanten zwischen benachbarten Feldern. Zudem ist bekannt, dass diese Kanten geradlinig verlaufen und weitgehend uniforme Farbbereiche im Bild voneinander trennen, in denen zudem wenige Kanten zu erwarten sind. Auf Grundlage dieser Beobachtungen wird ein untypisch großer Sobel-Kernel mit einer Größe von 15×15 Pixeln verwendet, dargestellt in Abbildung 3.5. Dieser Kernel sorgt für eine gezielte Erkennung der beschriebenen Eigenschaften in Bildern von Dartscheiben.

Um die gewünschten Charakteristiken hervorzuheben, wird das Eingabebild vor der Kantenerkennung in Graustufen umgewandelt und der Kontrast wird erhöht, um den Unterschied zwischen hellen und dunklen Bereichen zu betonen. Um Rauschen vor der Filterung zu entfernen, wird das Bild weichgezeichnet. Hochfrequente Informationen werden dadurch verworfen und etwaige Unterbrechungen oder Störungen der Kanten zwischen den Feldern verringert. Auf das resultierende Bild wird der beschriebene Sobel-Kernel in horizontaler und vertikaler Richtung angewendet, um Filterreaktionen von Intensitätsänderungen entlang beider Richtungen zu erlangen. Diese werden miteinander kombiniert und durch Thresholding binarisiert. Die Ausgabe ist eine binäre Maske, in denen Pixel des Wertes 1 Kanten im Eingabebild darstellen.

3.2.3.2 Skelettierung

Das gefilterte Kantenbild der Dartscheibe enthält aufgrund der Verwendung eines großen Kernels redundante Kanteninformationen durch Kanten, die mehrere Pixel breit sind. Diese breiten Kanten werden mittels Skelettierung auf ihre zentrale Kante reduziert [55]. Bei der Skelettierung werden die existierenden Kanten iterativ verringert, bis eine zentrale Kante ermittelt ist. Dazu wird das Konzept der Erosion verwendet, bei der Cluster von Pixeln in Binärbildern entlang ihrer Kontur verkleinert werden. Nach der Skelettierung des Kantenbildes verbleibt eine minimale Darstellung der extrahierten Kanten. Der verbliebene Informationsgehalt des Bildes wird dadurch auf das für die kommenden Schritte wesentliche reduziert.

3.2.4 Linienvverarbeitung

An diesem Punkt in der CV-Pipeline sind relevante Kanteninformationen aus dem Bild extrahiert und als minimale binäre Maske vorhanden. Der nächste Schritt zur Normalisierung der Dartscheibe ist das Identifizieren von Linien in der Kantenmaske. Ziel der Linienvverarbeitung ist es, eine mathematische Darstellung der radial angeordneten Kanten zu erlangen, die die Felder der Dartscheibe voneinander trennen. Über diese Darstellung wird mittels Transformationen eine erste Stufe der Entzerrung vorgenommen, indem die Winkel dieser Linien aneinander angeglichen werden.

Die Schritte der Linienvverarbeitung sind in Abbildung 3.6 dargestellt und auf die jeweiligen Schritte wird in den folgenden Unterabschnitten genauer eingegangen.

3.2.4.1 Linienerkennung

Um die Dartscheibe anhand von Linien zu entzerren, müssen in einem ersten Schritt Linien ermittelt werden. Für diesen Prozess wird die Hough-Transformation genutzt. Diese ermöglicht die Identifizierung von Liniensegmenten in Bildern und gibt diese als Liste von Start- und Endpunkten zurück: $L_{\text{points}} = \{(p_{i,\text{start}}, p_{i,\text{stop}}) \mid i \in [0, n_{\text{lines}} - 1]\}$, wobei n_{lines} die Anzahl der gefundenen Liniensegmente ist. In Abbildung 3.6 (1) werden erkannte Linien anhand eines Beispielbildes dargestellt. Jeder Linie ist zur Visualisierung eine zufällige Farbe zugeordnet. Zu erkennen ist, dass neben gewünschten langen Linien viele kurze Linien erkannt werden. Der Grund für eine Häufung vieler kurzer Linien liegt in der diskretisierten Darstellung von Pixeln und Ungenauigkeiten durch Verwackelungen, ungerade Feldgrenzen oder Verzerrungen der Kameralinse. Bei dem Prozess der Linienerkennung kann nicht davon ausgegangen werden, dass Linien ideal erkannt werden. Trotz dessen tragen kurze Linien mit hoher Wahrscheinlichkeit wenig relevante Informationen, sodass Linien, die eine geringere Länge als 5 px aufweisen, herausgefiltert werden.

Aus den Start- und Endpunkten der Liniensegmente lassen sich unter Verwendung der in Unterabschnitt 3.1.1 eingeführten Gleichungen die polaren Darstellungen $L_{\text{polar}} = \{(\rho_i, \theta_i) \mid i \in [0, n_{\text{lines}} - 1]\}$ errechnen mit $\rho_i \in [0, \text{diag}(w, h)]$ und $\theta \in [0^\circ, 180^\circ]$. Wie bereits bei der Einführung der Gleichung erwähnt, sind in dieser Darstellungsform keine Informationen zu Längen der Linien enthalten. Dieser Aspekt wird im folgenden Verarbeitungsschritt zum Vorteil genutzt.

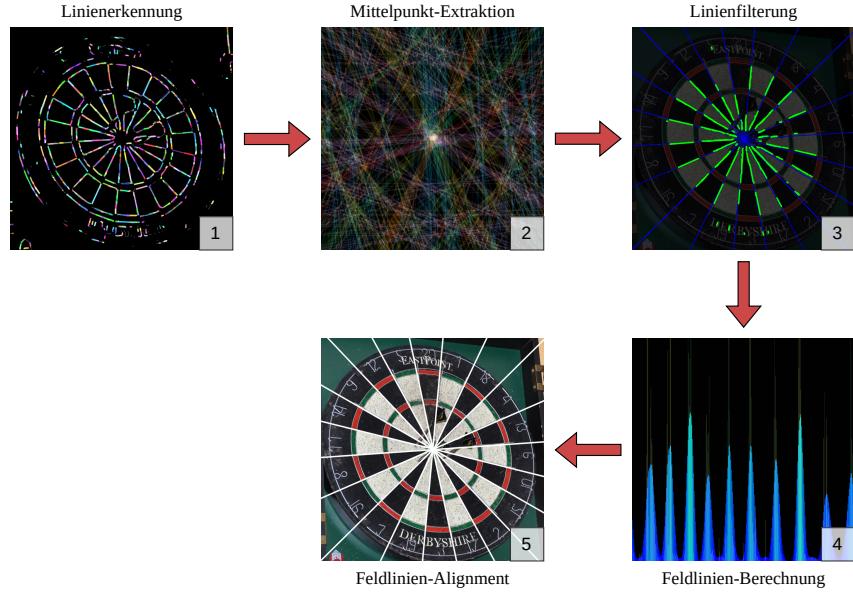


Abbildung 3.6: Veranschaulichung der Schritte der Linienverarbeitung. (1) Identifizierung von Linien im Kantenbild; jede Linie ist zur Visualisierung in einer zufälligen Farbe dargestellt. (2) Extraktion des Mittelpunktes anhand unterschiedlicher Linienwinkel; jede Klasse von Winkeln ist in einer zufälligen Farbe dargestellt. (3) Filterung der Linien anhand des Mittelpunktes; verbleibende Linien sind grün hervorgehoben, Feldlinienwinkel ϕ_i in blau. (4) Akkumulation der Winkel von Pixeln in gefilterten Linien. (5) Entzerrte Feldlinien; alle Winkel $\phi_i = 18^\circ$ sind weiß hervorgehoben.

3.2.4.2 Mittelpunktextraktion

Anhand der polaren Gleichungen L_{polar} wird in diesem Schritt der Mittelpunkt der Dartscheibe ermittelt. Der Mittelpunkt zeichnet sich dadurch aus, dass alle Linien, die zwischen Dartfeldern liegen – folglich als Feldlinien bezeichnet – auf diesen gerichtet sind. Unter der Annahme, dass alle Feldlinien in L_{polar} vorhanden sind, überschneiden sich eine Vielzahl dieser Linien im Mittelpunkt der Dartscheibe. Insbesondere ist bekannt, dass diese Linien in jeweils unterschiedlichen Winkeln auftreten, deren ungefähren Werte bekannt sind.

Unter Berücksichtigung dieser Beobachtung geschieht ein Binning von L_{polar} anhand der Winkel θ_i in $b = 10$ uniforme Bins B der Größe $\frac{180^\circ}{b} = 18^\circ$ mit den Intervallen $B_i = [i \times \frac{180^\circ}{b}, (i + 1) \times \frac{180^\circ}{b}]$. Für jeden dieser Bins wird eine binäre Maske erstellt, auf der die jeweiligen Polarlinien mit einheitlicher Intensität gezeichnet werden. Diese Masken werden anschließend überlagert und weichgezeichnet, um den Einfluss von Ungenauigkeiten zu minimieren. In dem resultierenden Bild zeichnet sich der Punkt P_{max} mit dem höchsten Wert dadurch aus, dass durch ihn die meisten Linien verschiedener Richtungen verlaufen. Diese Eigenschaft ist dadurch verfeinert, dass statt beliebiger Kanten gezielt Kanten mit bestimmten Eigenschaften als Grundlage für die Linien dienen. Durch diese Wahl an Eigenschaften ist mit hoher Wahrscheinlichkeit davon auszugehen, dass mit dem Punkt P_{max} der Mittelpunkt der Dartscheibe $m_{\text{Dart}} = (c_x, c_y)$ identifiziert ist.

Hinsichtlich der Robustheit dieses Algorithmus ist der Fall hervorzuheben, dass Feldlinien durch perspektivische Verzerrungen von den zu erwartenden Winkelintervallen abweichen können. Es kann dadurch zur Einordnung mehrerer Feldlinien in gleiche Bins oder dem Auslassen von Bins führen. Da bei der Ermittlung des Mittelpunktes jedoch nach einem globalen Maximum statt einem bestimmten Zahlenwert gesucht wird, ist ein gewisser Grad an Robustheit gegen nicht oder nicht korrekt gefüllte Bins gegeben.

Visualisiert ist die Extraktion des Mittelpunkts in Abbildung 3.6 (2). Linien gleicher Bins sind in der Visualisierung mit gleichen Farben dargestellt. Zu erkennen ist ein Highlight im Bulls Eye der Dartscheibe, in der sich die Linien der unterschiedlichen Bins überschneiden. Dieses Highlight ist der Mittelpunkt der Dartscheibe.

3.2.4.3 Linienfilterung

Die Mengen L_{points} und L_{polar} umfassen neben den für die Entzerrung relevanten Feldlinien weitere Linien, die nicht relevant für die Geometrie der abgebildeten Dartscheibe sind. Diese werden in diesem Schritt unter Verwendung des Mittelpunktes der Dartscheibe herausgefiltert. Zur Differenzierung zwischen möglichen Feldlinien und Linien, die mit Sicherheit keine Feldlinien sind, wird die Lotfuß-Distanz der Polarlinien zum Mittelpunkt genutzt. Ist eine Linie nicht auf den Mittelpunkt gerichtet, ist sie mit Sicherheit keine Feldlinie.

Die minimale Lotfuß-Distanz zwischen einem Punkt (\hat{x}, \hat{y}) und einer Linie in impliziter Form $(ax + by + c = 0)$ ist definiert durch [56]:

$$\text{dist}(ax + by + c = 0, (\hat{x}, \hat{y})) = \frac{|a\hat{x} + b\hat{y} + c|}{\sqrt{a^2 + b^2}}$$

Die implizite Form der Geraden lässt sich mit folgenden Gleichungen aus der Polarform berechnen:

$$\begin{aligned} \rho &= x \cos \theta + y \sin \theta \\ \iff 0 &= x \cos \theta + y \sin \theta - \rho \\ \Rightarrow a &= \cos \theta \\ \Rightarrow b &= \sin \theta \\ \Rightarrow c &= -\rho \end{aligned}$$

Durch Einsetzen dieser ermittelten Variablen in die Distanzberechnung folgt:

$$\begin{aligned} \text{dist}(ax + by + c = 0, (\hat{x}, \hat{y})) &= \frac{|a\hat{x} + b\hat{y} + c|}{\sqrt{a^2 + b^2}} \\ &= \frac{|\cos \theta \hat{x} + \sin \theta \hat{y} - \rho|}{\sqrt{\cos^2 \theta + \sin^2 \theta}} \\ &= |\cos \theta \hat{x} + \sin \theta \hat{y} - \rho| \end{aligned}$$

Mit dieser Gleichung lässt sich für jede ermittelte Polarlinie $(\rho_i, \theta_i) \in L_{\text{polar}}$ der Abstand zum Mittelpunkt der Dartscheibe m_{Dart} ermitteln. Anhand dieses Abstands werden die Linien gefiltert, sodass Linien, die mehr als 10 px von dem Mittelpunkt entfernt verlaufen, herausgefiltert werden.

Auf diese Weise werden diejenigen Linien \tilde{L}_{polar} und $\tilde{L}_{\text{points}}$ ermittelt, die auf den Mittelpunkt der Dartscheibe gerichtet sind und mögliche Teile der Feldlinien sind. Es kann an diesem Punkt jedoch nicht sicher ausgeschlossen werden, dass sich keine Outlier unter den gefilterten Linien befinden. Zu erkennen ist die Existenz von Outliern in den gefilterten Linien in Abbildung 3.6 (3). In dem Beispiel liegen Liniensegmente in den Schriftzügen auf der Dartscheibe, die auf den Mittelpunkt gerichtet sind und kein Teil von Feldlinien sind.

3.2.4.4 Feldlinien-Berechnung

Zur Identifizierung der Winkel ϕ_i der Feldlinien wird eine adaptierte Hough-Transformation auf die gefilterten Linien \tilde{L}_{polar} und $\tilde{L}_{\text{points}}$ verwendet. In dieser wird für jeden Pixel p aller Linien je Winkel $\theta_{i,p}$ und Abstand $d_{i,p}$ zum Mittelpunkt ermittelt. In einem Akkumulator-Array A^{360} werden die Winkel in 360 Bins mit einer Granularität von $0,5^\circ$ aufsummiert, gewichtet invers proportional zu $d_{i,p}$. Dadurch wird Pixeln, die weit von dem Mittelpunkt entfernt liegen, ein geringes Gewicht zugeordnet, da diese einer größeren Wahrscheinlichkeit unterliegen, kein Bestandteil einer Feldlinie zu sein. Ziel der Verwendung von A^{360} ist das Identifizieren von Clustern der Winkel.

Zur Minderung von Outliern und zur Festigung der mittleren Winkel wird A^{360} zweifach radial mit einem Fenster von 5° – entsprechend 10 Bins – geglättet. In dem resultierenden Akkumulator werden die 10 größten Peaks ϕ_i durch NMS identifiziert; diese Peaks sind die häufigsten Winkel von Liniensegmenten zum Mittelpunkt der Dartscheibe. Durch die getroffenen Annahmen ist davon auszugehen, dass diese Werte die Winkel der Feldlinien angeben. Eine Darstellung eines Akkumulators A^{360} ist mit Abbildung 3.6 (4) gegeben.

3.2.4.5 Winkelentzerrung

An dieser Stelle sind Mittelpunkt und Winkel der Feldlinien der Dartscheibe bekannt. Ziel dieses Schrittes ist es, die Winkel der Feldlinien zu normalisieren, sodass die Feldlinienwinkel uniforme Werte besitzen.

Um diese Entzerrung zu erzielen, wird eine Minimierung vorgenommen, in der eine affine Transformation gesucht wird, die diese Winkel bestmöglich aneinander anpasst und auf einen Wert von 18° angleicht. Diese Optimierung beginnt ausgehend von einer Startlinie und entzerrt alle restlichen Linien iterativ. Die Durchführung geschieht für jede der zehn Feldlinien als Startlinie und die finale Transformation wird berechnet durch den Mittelwert aller ermittelten Transformationen. Im Folgenden wird die allgemeine Transformationssequenz angegeben.

Die erste Teiltransformation ist die Translation des Mittelpunktes m_{Dart} in den Koordinatenursprung $O = (0, 0)$. Dieser Schritt ist relevant, da atomare affine Transformationen um O zentriert sind. Darauf folgt die vertikale Ausrichtung der Startlinie L_s mit Winkel ϕ_s durch eine Rotation um $-\phi_s$, sodass $\phi'_s = 0^\circ$ erzielt wird. Gefolgt wird diese Rotation von der horizontalen Ausrichtung der Orthogonalen $L_o = L_{(s+5) \bmod 10}$ durch eine Scherung entlang der Vertikalen. Wichtig bei diesem Schritt ist, dass die vertikale Scherung den Winkel von ϕ'_s nicht beeinflusst während eine Ausrichtung $\phi'_o = 90^\circ$ erreicht wird. An diesem Punkt sind $2/10$ Winkel entzerrt; die restlichen Winkel werden mit einer vertikalen Skalierung derart ausgerichtet, dass ein minimaler Abstand zwischen Zielwinkeln und Feldlinienwinkeln resultiert. Sind alle Feldlinienwinkel perfekt erkannt, ist eine optimale Skalierung möglich, sodass dieser Fehler gleich Null ist. Jedoch ist dies durch u. a. Diskretisierung und Artefakte in Linienerkennungen nicht gegeben und ein mittleres Minimum aller Winkeldifferenzen zu ihren Zielpositionen muss gebildet werden. Im Anschluss wird die vertikale Ausrichtung der Startlinie L_s rückgängig gemacht, sodass die ϕ'_s seinen Zielwinkel besitzt. Zuletzt wird eine Translation von O auf m_{Dart} durchgeführt und die Transformationssequenz ist abgeschlossen.

Diese Schritte werden für alle Startindizes $s \in [0, 9]$ ausgeführt und die finale Transformation wird durch Mittelwertbildung errechnet. Dadurch wird eine optimale Entzerrung aller Winkel ϕ_i erlangt, die nicht durch die Wahl der Startlinie beeinflusst ist. In Abbildung 3.6 (5) ist eine Dartscheibe nach der Entzerrung der Feldlinienwinkel dargestellt. Zu erkennen ist, dass trotz Angleichung der Winkel ϕ_i keine Normalisierung der Dartscheibe erreicht ist. Um die Normalisierung zu vollenden, muss die elliptische Dartscheibe in eine runde Form überführt und skaliert werden.

3.2.5 Orientierung

An dieser Stelle in dem Algorithmus ist der Mittelpunkt der Dartscheibe identifiziert und die Winkel der Feldlinien sind normalisiert, jedoch ist an diesem Punkt noch keine korrekte Normalisierung der Dartscheibe gegeben. Durch perspektivische Verzerrungen ist es möglich, dass die Dartscheibe nicht die Form eines Kreises, sondern einer Ellipse besitzt, wie in Abbildung 3.7 anhand eines Beispiels dargestellt ist.

Das Vorgehen der Orientierung basiert auf der Grundidee des DeepDarts-Systems, in welchem Orientierungspunkte identifiziert werden, deren Positionen in einem normalisierten Bild bekannt sind. Anhand dieser Punktverschiebungen wird eine Homographie abgeleitet, die eine Transformation des Bildes vornimmt, um diese Punkte auf ihre Zielpositionen zu überführen und die Dartscheibe zu entzerrn. Im Gegensatz zu wenigen fest definierten Orientierungspunkten, wie sie in DeepDarts verwendet werden, wird für dieses System eine Vielzahl an Orientierungspunkten identifiziert, um eine wesentlich robustere Entzerrung zu ermöglichen. Die wesentlichen Schritte der Orientierung sind in Abbildung 3.8 dargestellt und werden in den folgenden Unterkapiteln genauer erläutert.



Abbildung 3.7: Ellipsoide Darstellung einer Dartscheibe mit uniformen Feldlinienwinkeln.

3.2.5.1 Identifizierung von Orientierungspunkten

Die konkreten Orientierungspunkte werden nach dem Vorbild von DeepDarts ausgewählt und befinden sich an Eckpunkten zwischen Dartfeldern. Diese Punkte sind besonders markant und durch ihre Positionen klar zu identifizieren. Es werden die Kreuzpunkte auf der Innenseite des Triple-Rings sowie entlang der Innenseite des Double-Rings identifiziert, was eine Gesamtzahl von bis zu 60 Orientierungspunkten ergibt. Diese Punkte sind dadurch charakterisiert, dass sie an jede existierende Feldfarbe – schwarz, weiß, rot und grün – grenzen, was für die Identifizierung genutzt wird.

Log-polare Darstellung Für die Identifizierung der Orientierungspunkte wird das Bild in das log-polare Koordinatensystem überführt. Dazu wird das Bild um den Mittelpunkt der Dartscheibe abgewickelt und ausgerollt, sodass sich der Mittelpunkt der Dartscheibe entlang der Bildkante erstreckt. Dartfelder werden dadurch von Teilstücken eines Kreises zu Rechtecken transformiert und es erfolgt eine Parallelisierung der Feldlinien. Durch die Entzerrung der Winkel sind die Feldlinien in der log-polaren Darstellung äquidistant und die Koordinaten der Feldlinien sind bekannt. Auf Grundlage dieses Wissens können die Farben der schwarzen und weißen Felder extrahiert werden. Diese werden genutzt, um die Bestimmung der Lage der Orientierungspunkte zu unterstützen.

Identifizierung von Ecken Orientierungspunkte befinden sich an Eckpunkten von Feldern. Dadurch ergibt sich, dass sie bei einer Eckenerkennung stark ausschlagen. Mithilfe der Harris Corner Detection werden Ecken in dem Bild identifiziert und auf Grundlage ihrer Position gefiltert, sodass eine Liste an Eckpunkten entlang der Feldlinien identifiziert wird. Diese Eckpunkte sind potenzielle Orientierungspunkte, sofern ihre Umgebung wie erwartet gestaltet ist.

3.2.5.2 Klassifizierung von Orientierungspunkten

Für jeden potenziellen Orientierungspunkt in dem log-polaren Bild der Dartscheibe wird die Surrounding betrachtet. Als Surrounding wird die unmittelbare, quadratische Umgebung um einen potenziellen Orientierungspunkt bezeichnet. Durch sie ist eine Einordnung möglich, ob es sich um einen tatsächlichen Orientierungspunkt handelt oder nicht. Für jede Surrounding wird eine Farbraumtransformation in einen Farbraum mit dem Namen CrYV vorgenommen. Dieser Farbraum setzt sich aus Kanälen unterschiedlicher Farbräume zusammen und ist darauf ausgelegt, die Unterschiede zwischen schwarzen, weißen und bunten Feldern zu verstärken. Dabei wird nicht zwischen rot und grün unterschieden, da sich diese Farben aus den Positionen der schwarzen und weißen Felder ableiten lassen.

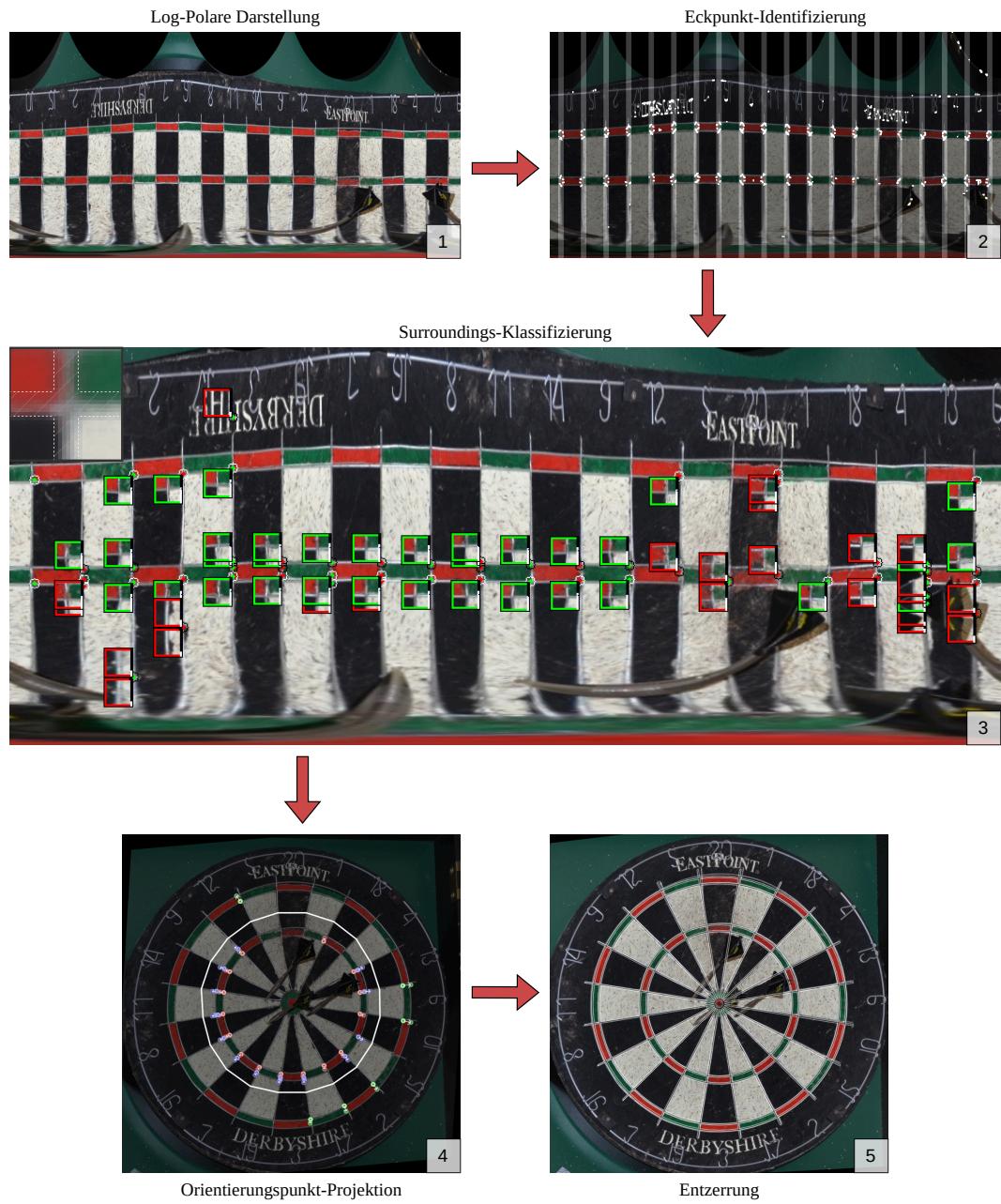


Abbildung 3.8: Schritte zur Orientierung der Dartscheibe. (1) Log-polare Darstellung der Dartscheibe. (2) Identifizierte Eckpunkte. Hervorgehobene Streifen veranschaulichen die erwarteten Bereiche der Feldlinien. (3) Identifizierung und Klassifizierung von Surroundings. Positiv klassifizierte Surroundings sind grün umrandet, negative rot. Weiße Balken an der Seite der Surroundings geben den Score an, der zur Klassifizierung errechnet wird; der Threshold befindet sich auf halber Höhe der Balken. In der oberen linken Ecke befindet sich die mittlere Surrounding. (4) Projektion der identifizierten Orientierungspunkte. Pfeile deuten die Richtung der Verschiebungen an; der graue Ring ist die Trennlinie zwischen inneren und äußeren Punkten. (5) Entzerrte Dartscheibe. Die ideale Entzerrung ist über das Bild gelegt.

Im CrYV-Farbraum werden die mittleren Farbwerte der Ecken aller Surroundings klassifiziert. Durch diese lässt sich einerseits herausfinden, ob ein potenzieller Orientierungspunkt ein Outlier ist, und andererseits, in welcher Orientierung sich dieser befindet. Dazu werden die Ecken anhand heuristisch erarbeiteter Thresholds in die Kategorien schwarz, weiß und farbig eingeordnet. Entspricht eine Surrounding nicht der Erwartung, dass ein schwarzes, ein weißes und zwei farbige Bereiche in diesem liegen, wird der jeweilige Punkt nicht weiter betrachtet. Durch diese Einordnung werden diejenigen Punkte entfernt, die mit großer Wahrscheinlichkeit keine Orientierungspunkte sind.

In einem folgenden Schritt wird eine mittlere Surrounding aller verbliebenen Eckpunkte als Median aller normalisierten Surroundings errechnet. Dieses mittlere Surrounding wird auf zweierlei Arten gegen jede Surrounding verglichen. Die erste Metrik ist der Abstand der Farbwerte im Lab-Farbraum, die zweite Metrik ist der SSIM-Index. Durch Gewichtung dieser Metriken wird eine pessimistische Kategorisierung der Surroundings in valide und nicht valide unternommen. Dieser Threshold wird dabei sehr strikt gesetzt, um die Wahrscheinlichkeit von Outliern möglichst gering zu halten. Da für eine Homographiefindung lediglich vier Punkte notwendig sind, ist der Verlust einzelner korrekter Orientierungspunkte vertretbar. Nach diesem Schritt verbleiben diejenigen Punkte im Bild, die auf den Eckpunkten der Felder liegen und deren Orientierung durch ihre Surroundings bekannt sind.

3.2.5.3 Berechnung von Punktverschiebungen

Durch die Position und die Farbgebung der Surroundings der klassifizierten Eckpunkte ist eine Rückrechnung auf die Position der Punkte im Ursprungsbild nach der Entzerrung der Feldlinienwinkel möglich. Darüber hinaus ist durch die Orientierung der Surroundings bekannt, ob sich Orientierungspunkte auf der Innen- oder Außenseite eines Rings befinden. Durch diese Informationen lässt sich für jeden Punkt eindeutig zuordnen, auf welcher Position dieser in einem ideal entzerrten Bild liegen muss. Die Unterscheidung zwischen Innenseite des Triple- und Innenseite des Double-Rings lässt sich durch Bildung des Mittelwerts der Orientierungspunkte auf der Außenseite des Triple-Rings ermitteln. Jegliche Punkte, deren Abstand geringer als das 1,2-fache des entsprechenden Outer-Triple-Orientierungspunkts derselben Feldlinie besitzen, werden als Inner-Triple-Orientierungspunkt klassifiziert, alle anderen als Inner-Double-Orientierungspunkt. Nicht erkannte Outer-Triple-Orientierungspunkte werden durch Interpolation identifiziert.

Nach diesem Prozess sind Start- und Zielpunkte bekannt und damit einhergehend die Verschiebungen aller Orientierungspunkte. Sofern mindestens 3/60 möglichen Punkten identifiziert sind, ist eine Entzerrung der Dartscheibe möglich, da der Mittelpunkt als vierter Punkt fungiert, um eine Homographie vollständig zu parametrisieren.

3.2.5.4 Entzerrung der Dartscheibe

Die finale Entzerrung der Dartscheibe geschieht durch Anwendung des RANSAC-Algorithmus. Hintergrund ist die Möglichkeit der Existenz von Outliern in den identifizierten Orientierungspunkten. Als Outlier werden fehlerhaft erkannte Orientierungspunkte bezeichnet, deren Positionen entweder falsch zugeordnet sind oder die sich nicht an Positionen von Orientierungspunkten befinden. Bei der Ableitung einer Homographie auf der Grundlage aller identifizierter Orientierungspunkte sorgen Outlier für Artefakte in der Entzerrung.

Für die Implementierung von RANSAC werden $3N_{OP}$ Homographiefindungen durchgeführt, wobei N_{OP} die Anzahl der identifizierten Orientierungspunkte angibt. Für jeden Durchlauf werden zufällig fünf Orientierungspunkte ausgewählt, zu denen der Dartscheibenmittelpunkt hinzugefügt wird. Anhand dieser Punkte wird eine Entzerrungshomographie identifiziert. Nach der Anwendung dieser Homographie werden die Distanzen aller identifizierten Orientierungspunkte zu ihren Zielpositionen bestimmt. Als finale Homographie wird diejenige gewählt, die die geringste Distanzsumme aller Homographien aufweist.

Es bleibt festzuhalten, dass der Determinismus der Ausgaben durch die Verwendung von RANSAC nicht mehr gegeben ist. Mehrfaches Ausführen des Algorithmus auf den selben Eingaben kann zu unterschiedlichen Ergebnissen führen.

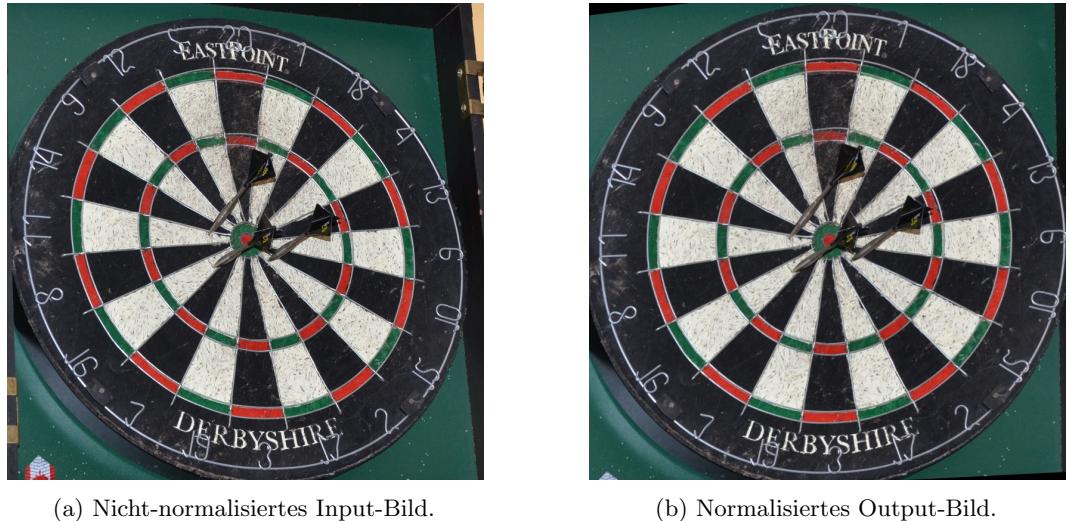


Abbildung 3.9: Entzerrung eines Beispielbildes aus dem DeepDarts-Datensatz [4]. Abbildung 3.9a zeigt das Input-Bild, Abbildung 3.9b zeigt das normalisierte Output-Bild nach der Verarbeitung durch die CV-Pipeline.

3.2.6 Zusammenführen aller Komponenten

An diesem Punkt ist die gesamte CV-Pipeline durchgelaufen und es wurden mehrere Transformationsmatrizen für verschiedene Zwischenschritte der Normalisierung ermittelt. Die Reihenfolge der angewandten Transformationen, um die Dartscheibe zu normalisieren, lautet:

1. Skalierung der Dartscheibe auf die Berechnungsgröße nach Unterabschnitt 3.2.2 (Vorverarbeitung): M_{scale}
2. Angleichung der Feldlinienwinkel nach Unterabschnitt 3.2.4 (Linienverarbeitung): M_{align}
3. Projektion der Orientierungspunkte auf Ziel-Positionen nach Unterabschnitt 3.2.5 (Orientierung): M_{project}

Die Aneinanderreihung dieser Transformationen führt zu der finalen Transformation M_{final} , die wie folgt berechnet wird:

$$M_{\text{final}} = M_{\text{project}} \times M_{\text{align}} \times M_{\text{scale}}$$

Nach der Anwendung dieser Transformation auf das Eingabebild wird das Bild hinsichtlich seine Abmessungen auf die Eingabegröße des neuronalen Netzes zugeschnitten, womit die Normalisierung des Bildes abgeschlossen ist. Der Effekt der Entzerrung anhand des Beispielbildes ist in Abbildung 3.9 dargestellt. Es sind Eingabe- und Ausgabebild dargestellt, um den Effekt der Normalisierung zu veranschaulichen.

3.3 Implementierung

Nach der Erläuterung der Methodik zur Normalisierung der Bilddaten widmet sich dieser Abschnitt Details zur Implementierung. Es wird auf ausgewählte Bereiche eingegangen, in denen die Implementierung wesentliche Einblicke in die Arbeitsweise des vorgestellten Algorithmus liefert. In diesem Abschnitt werden drei Unterabschnitte zu jeweils einem bestimmten Aspekt des vorgestellten Algorithmus erklärt. In Unterabschnitt 3.3.1 wird die Entzerrung der Feldlinienwinkel betrachtet, sodass die Winkel der Felder auf der Dartscheibe uniform verlaufen. Anschließend wird in Unterabschnitt 3.3.2 auf die Identifizierung relevanter Farben eingegangen. Dabei spielt der für diese Thesis konzipierte CrYV-Farbraum eine wesentliche Rolle. In Unterabschnitt 3.3.3 wird zuletzt betrachtet, wie die Klassifizierung von Surroundings stattfindet, anhand welcher Kandidaten von Orientierungspunkten einer Filterung unterzogen werden.

3.3.1 Winkelfindung aus gefilterten Linien

Die Aufgabe der Winkelfindung gefilterter Linien ist die Identifizierung von Winkel-Clustern. Die Winkel für diese Berechnung stammen aus Linien, deren unendliche Verlängerung nahe dem Dartscheibenmittelpunkt m_{Dart} verläuft.

Für die Bewältigung dieser Aufgabe wird eine Adaption der Hough-Transformation verwendet. Eingabe in diesen Teilalgorithmus ist ein binäres Bild, auf dem die gefilterten Linien eingezeichnet sind. Für jegliche Pixel, die in diesem Bild eingezeichnet sind, wird der Winkel φ_i zum Mittelpunkt $m_{\text{Dart}} = (c_x, c_y)$ bestimmt:

$$\varphi_i = \arctan2(y_i - c_y, x_i - c_x)$$

Für die Implementierung dieser Berechnung werden die von NumPy zur Verfügung gestellte vektorisierte Funktion `np.arctan2` verwendet. Diese Funktion, wie auch weitere Funktionen der Bibliothek, zeichnet sich durch eine effiziente und vektorisierte Berechnung von einer Eingabeliste aus. Unter der Verwendung dieser vektorisierten und zusätzlich kompilierten Funktionen ist eine schnelle Ausführung komplexer Berechnungen trotz der Ausführung mit Python möglich.

Aus dieser Berechnung gehen Winkel der Pixel im Wertebereich $[0^\circ, 180^\circ]$ hervor, die auf eine Granularität von 0,5 px quantisiert und in den Akkumulator A^{360} mit 360 Bins eingetragen werden. Dieser Akkumulator wird einer doppelten Filterung durch einen Box-Kernel der mit einer Breite von zehn Bins unterzogen, welche keinen radialen Verzerrungen unterliegen darf. Um dies zu erzielen, werden die zehn beginnenden und endenden Bins aus A^{360} an das jeweils gegenüberliegende Ende kopiert. Anschließend wird der erweiterte Akkumulator gefiltert und die angefügten Enden werden verworfen. Durch diese Technik ist eine Glättung unter Beibehaltung radialem Einflusses ermöglicht.

3.3.2 Farben-Identifizierung

Für die Identifizierung von Orientierungspunkten werden die Farben der Umgebungen potenzieller Orientierungspunkte klassifiziert. Der Kontext, in welchem diese Verarbeitung stattfindet, ist in Abbildung 3.8 (3) dargestellt: Die Dartscheibe ist log-polar um ihren Mittelpunkts abgerollt und die Kandidaten sind identifiziert. Durch diese Form der Darstellung sind korrekte Orientierungspunkte derart positioniert, dass ihre Surrounding die vier unterschiedlichen Farben der Dartscheibenfelder in je einer Ecke enthält. Zur Klassifizierung werden Funktionen verwendet, die die mittleren Farben dieser Eckbereiche der Surroundings in schwarz, weiß und farbig klassifizieren.

CrYV-Farbraum Die Farben der Surroundings werden zur Einordnung in den CrYV-Farbraum umgewandelt. Der CrYV-Farbraum ist derart gestaltet, dass eine gezielte Isolation für die Unterscheidung relevanter Farbcharakteristiken durch gezieltes Thresholding durchgeführt werden kann. Die Farbkanäle der CrYV-Bilder werden separatem Thresholding unterzogen, um Farbeinflüsse spezifisch zu untersuchen und um auszumachen, ob eine gewisse Farbgebung vorhanden ist.

Klassifizierung schwarzer und weißer Bereiche Die durchschnittlichen Farben schwarzer und weißer Felder der Dartscheibe sind durch bereits vollzogene Vorverarbeitungsschritte bekannt. Zur Einordnung, ob ein Feld schwarz oder weiß ist, wird der zu überprüfende Bereich der Surrounding – bezeichnet als Patch – mit der schwarzen bzw. weißen Farbe analysiert. In einem ersten Schritt wird die mittlere Farbe des Patches bestimmt. Die absoluten Differenzen der jeweiligen Farbkanäle des Patches sowie der Ziel-Farbe werden berechnet und die Summe dieser Kanäle wird berechnet. Anhand eines empirisch ermittelten Schwellwerts wird ein Thresholding durchgeführt, durch welches eine Klassifizierung in „schwarz“ oder „nicht schwarz“ (und analog für weiß) geschieht:

$$\text{is_bw}(C_p, C_r, T_C) = \begin{cases} \text{true}, & \text{wenn } \sum_{i=0}^2 |C_r[i] - C_p[i]| < T_C \\ \text{false} & \text{sonst} \end{cases}$$

In dieser Berechnung stehen $C_p \in \mathbb{R}^3$ und $C_r \in \mathbb{R}^3$ für 3-Kanal CrYV-Farben von Patch und Referenzfarbe. $T_C \in \mathbb{R}$ ist der Farb-Threshold, der unterschritten werden muss, um als die Referenzfarbe klassifiziert zu werden.

Klassifizierung roter und grüner Bereiche Im Gegensatz zur Einordnung schwarzer und weißer Farben stehen für die Einordnung roter und grüner Farben aus technischen Gründen keine Referenzfarben zur Verfügung. Vor der Hintergrund dieser Herausforderung ist der CrYV-Farbraum derart konzipiert, dass rote und grüne Farben entsprechend markant sind und durch Thresholding gezielt identifiziert werden können. Die Farbinformationen C_p eines Patches werden anhand ihres Cr-Kanals analysiert und mit Referenzwerten typischer roter und grüner Kanalwerte verglichen:

$$\text{is_color}(C_p, T_C, t_{\text{red}}, t_{\text{green}}) = \begin{cases} \text{true}, & \text{wenn } \min(|t_{\text{red}} - C_p[0]|, |t_{\text{green}} - C_p[0]|) < T_C, \\ \text{false} & \text{sonst} \end{cases}$$

In dieser Gleichung stehen t_{red} und t_{green} für zu erwartende Referenzwerte roter und grüner Felder.

3.3.3 Klassifizierung von Surroundings

Unter Verwendung des CrYV-Farbraums ist die Einordnung von Feldfarben effizient ermöglicht und kann durch die zuvor beschriebenen Techniken implementiert werden. Diese Klassifizierung von Feldfarben ermöglicht fortlaufend die Klassifizierung von Surroundings hinsichtlich der Filterung von Orientierungspunkten.

Orientierungspunkte sind als diejenigen Punkte definiert, die an den Ecken von Feldern liegen. Damit weisen sie in ihren Surroundings zwei farbige sowie eine schwarze und eine weiße Ecke vor. Diese Eigenschaft wird genutzt, um Kandidaten möglicher Orientierungspunkte zu filtern und zugleich ihre Orientierung zu bestimmen.

Für jede Surrounding wird ermittelt, welche Farbgebung in welcher Ecke der Surrounding im Mittel vorliegt. Die Farbinformationen der Ecken werden vermerkt und die Kombination dieser Informationen wird genutzt, um die Art des Orientierungspunkts zu identifizieren. Orientierungspunkte entlang der inneren Seite des Double- oder Triple-Rings zeichnen sich durch farbige außen liegende Ecken in Kombination mit einer schwarzen und einer weißen innen liegenden Ecke aus. Für Orientierungspunkte entlang der Außenseite des Triple-Rings gelten die gespiegelten Bedingungen bezüglich der Farbgebung. Die Positionierung der schwarzen und weißen Ecken liefert Auskunft über die Parität der Orientierungslinie.

Mit dieser Unterscheidung können diejenigen Kandidaten der Orientierungspunkte herausgefiltert werden, deren Surroundings nicht im Einklang mit der zu erwartenden Farbgebung sind. Die gefilterte Liste der Orientierungspunkte wird zur Bestimmung einer mittleren Surrounding verwendet, indem die Surroundings der verbliebenen Orientierungspunkte hinsichtlich ihrer Ausrichtung normalisiert werden und ein Mittelwert aller Surroundings ermittelt wird.

Anhand dieser mittleren Surrounding werden die potenziellen Orientierungspunkte klassifiziert. Die Klassifizierung verläuft auf Grundlage zweier Metriken: Lab-Abgleich und SSIM. Bei dem Lab-Abgleich wird eine Farbraumtransformation aller Surroundings in den Lab-Farbraum vorgenommen. In diesem werden die Kanaldifferenzen der roten und grünen Ecken betrachtet und es wird anhand der folgenden Ähnlichkeits-Metrik bestimmt, zu welchem Grad sich die Farben ähneln:

$$\text{sim}(S_1, S_2) = \exp^{-0.01/2(d_r + d_g)}$$

Hinter d_r und d_g stehen in dieser Formel die mittleren Differenzen der roten und grünen Bereiche der Surroundings S_1 und S_2 . Diese Metrik liefert einen Wert, anhand dessen ein Ähnlichkeitsgrad der gegebenen Surroundings abgelesen werden kann. Zusätzlich zu dieser Metrik wird der SSIM-Index genutzt, um die strukturelle Ähnlichkeit aller Surroundings zu der mittleren Surrounding zu ermitteln. Durch diese werden Outlier herausgefiltert, die nicht der zu erwarteten Struktur folgen und lediglich zufällig die korrekten mittleren Farben in den Ecken aufweisen.

Durch Abgleich des Mittelwerts der sim-Metrik und des SSIM-Indexes mit einem vordefinierten Threshold wird klassifiziert, ob ein potenzieller Orientierungspunkt als dieser klassifiziert wird oder ob dieser von dem System abgelehnt wird. Durch diese Klassifikation werden die finalen Orientierungspunkte, die für die Findung einer Homographie zur Normalisierung der Dartsscheibe genutzt werden, ermittelt.

3.4 Ergebnisse

In diesem Abschnitt werden die Ergebnisse der algorithmischen Normalisierung von Dartscheiben aufgezeigt. Dazu werden die verwendeten Metriken in Unterabschnitt 3.4.1 vorgestellt. Anschließend werden die Datenquellen in Unterabschnitt 3.4.2 aufgezeigt, anhand derer die Auswertung stattfindet. In Unterabschnitt 3.4.3 wird eine quantitative Auswertung vorgenommen. Es werden die Ergebnisse, die mit dem in dieser Arbeit vorgestellten Algorithmus erzielt werden konnten, hinsichtlich der aufgezeigten Metriken sowie weiteren Merkmalen analysiert und mit den durch DeepDarts auf den selben Daten erzielten Ergebnissen verglichen.

3.4.1 Metriken

Für die Messung der Entzerrungsgenauigkeit des entwickelten Algorithmus werden zwei Metriken verwendet. Die erste Metrik χ misst die Fähigkeit eines Systems, eine Homographie zur Entzerrung einer Dartscheibe zu ermitteln, unabhängig von ihrer Genauigkeit:

$$\chi(S, I) = \begin{cases} 1, & \text{wenn System } S \text{ zu Bild } I \text{ eine Homographie ermitteln kann} \\ 0 & \text{sonst} \end{cases}$$

Die zweite Metrik $\Lambda(\tilde{H}, \hat{H})$ bestimmt die Genauigkeit der ermittelten Entzerrungs-Homographie \hat{H} , gegeben einer Ziel-Homographie \tilde{H} . Da ein trivialer Vergleich der Zahlenwerte der ermittelten Homographien wenig Aufschluss über die konkrete Genauigkeit der Entzerrung liefert, ist eine komplexere Metrik notwendig. Für die verwendete Metrik Λ werden $N_{\text{OP, max}} = 61$ unterschiedliche Orientierungspunkte verwendet. Diese befinden sich entlang der Feldlinien radial verteilt in den Ringen der äußeren Bulls, des äußeren Triple-Rings und des äußeren Double-Rings. Zusätzlich ist der Mittelpunkt als weiterer Orientierungspunkt mit aufgenommen. Die Positionen $P_{i \in [N_{\text{OP, max}}]}$ aller Orientierungspunkte im Zielsbild sind durch die Definition der Entzerrung festgelegt. Diese Punkte werden durch die inverse Ziel-Homographie an ihre Ursprungspositionen $\tilde{P}_i = \text{inv}(\tilde{H}) \times P_i$ transformiert und von dort durch die ermittelte Homographie zu den vorhergesagten Zielpositionen $\hat{P}_i = \hat{H} \times \tilde{P}_i$ rücktransformiert. Der Wert der Metrik ist definiert durch:

$$\Lambda(\tilde{H}, \hat{H}) = \frac{1}{N_{\text{OP, max}}} \sum_{i=1}^{N_{\text{OP, max}}} \|P_i - \hat{H} \times \text{inv}(\tilde{H}) \times P_i\|_2$$

Durch diese Metrik ist eine Quantifizierung der Ähnlichkeit zweier Homographien zur Entzerrung einer Dartscheibe möglich. Zu sehen ist bei dieser Definition, dass $\Lambda(\tilde{H}, \hat{H}) = 0 \text{ px}$, wenn $\hat{H} = \tilde{H}$, da $\hat{H} \times \text{inv}(\tilde{H}) = \text{Id}$, die Identitätsmatrix, ist.

3.4.2 Verwendete Daten

Die Daten dieser Auswertung stammen aus fünf unterschiedlichen Quellen und werden strikt voneinander getrennt gehalten. Dies dient der Identifizierung von einerseits Verzerrungen auf Daten und andererseits Schwachstellen in dem getesteten System. Die Datenquellen sind in Tabelle 3.1 aufgelistet und stellen sich zusammen aus synthetischen Daten sowie Daten des DeepDarts-Systems.

Die Daten beinhalten lediglich positive Datensätze, in welchen ausschließlich Bilder vorhanden sind, die eine Dartscheibe abbilden. Da die Aufgabe der Systems nicht die Identifizierung von Dartscheiben ist, sondern die Entzerrung dieser, kann die Existenz von Dartscheiben in den Bildern vorausgesetzt werden. Die Ausführung des Algorithmus auf Bildern ohne Dartscheibe ist nicht von Relevanz.

Datenquelle	Gerenderte Bilder	DeepDarts- d_1 Validierung	DeepDarts- d_1 Test	DeepDarts- d_2 Validierung	DeepDarts- d_2 Test
Anzahl Bilder	2.048	1.000	2.000	70	150
Automatische Annotation	✓	✗	✗	✗	✗

Tabelle 3.1: Datenquellen für die Auswertung der Dartscheibenentzerrungen.

3.4.3 Quantitative Auswertung

Die quantitative Auswertung ist unterteilt in die Auswertung der Geschwindigkeit, der zuvor beschriebenen Metriken und einer zusammengefassten Auswertung. Die Aufteilung der Ergebnisse in Render-Daten und DeepDarts-Daten ergibt sich aus den Unterschieden der Daten. Während die DeepDarts-Daten derart vorverarbeitet sind, dass sie die Dartscheibe weitestgehend zentriert in Bildern fester Dimensionen zeigt, sind die synthetischen Daten wesentlich variabler hinsichtlich der Darstellung der Dartscheiben.

Die Auswertungen sind jeweils unterteilt in das in dieser Thesis erarbeitete System und die DeepDarts-Systeme DeepDarts- d_1 und DeepDarts- d_2 , um die Unterschiede der Genauigkeiten darzustellen. Da die DeepDarts-Systeme Single-Shot-Neural-Networks sind, in welchem die Normalisierung und die Lokalisierung der Dartpfeile nicht voneinander getrennt betrachtet werden können, wird die Geschwindigkeit der Normalisierung gleichgesetzt mit der Gesamtdauer der Vorhersage.

Ausgeführt wurde die Auswertung auf einer AMD Ryzen 3 3100 CPU. Es wurde trotz der Möglichkeit einer GPU-Ausführung der DeepDarts-Systeme keine Grafikkarte verwendet, um die Vergleichbarkeit der Systeme unter der Verwendung der selben Hardware zu gewährleisten. Aufgrund des Nichtdeterminismus des Algorithmus durch die Verwendung von RANSAC wurden fünf Durchläufe der Auswertung vorgenommen, deren Mittelwerte zur Evaluation verwendet wird.

3.4.3.1 Geschwindigkeit der Vorhersagen

Die Ausführungszeiten der jeweiligen Systeme sind in Abbildung 3.10 dargestellt. Die Ausführungszeiten von DeepDarts liegen mit durchschnittlich 131 ms auf den DeepDarts-Datensätzen und 239 ms auf den gerenderten Daten weitaus unter den Ausführungszeiten des Systems dieser Thesis. Diese liegen bei durchschnittlich 333 ms für die DeepDarts-Daten und 433 ms für die Render-Daten. Die Ausführungszeiten des in dieser Thesis erarbeiteten Algorithmus belaufen sich im Mittel etwa auf die doppelte Dauer der DeepDarts-Systeme, jedoch variiert dieser Faktor stark basierend auf der Datenquelle.

Die Unterschiede der Inferenzzeiten der Datenquellen ergeben sich aus den Abmessungen der Bilder. Die DeepDarts-Daten sind bereits vorverarbeitet, sodass sie ein quadratisches Seitenverhältnis mit einer Auflösung von 800×800 px aufweisen. Die gerenderten Daten hingegen sind für diese Auswertung nicht vorverarbeitet und werden den Systemen in den originalen Auflösungen präsentiert. Die Seitenlängen von Bildern der Render-Daten betragen mindestens 434 px, maximal 3.998 px und die durchschnittliche Seitenlänge beträgt 2.147 px. Verteilungen der Seitenverhältnisse sind in Abbildung 3.11 dargestellt.

Die tatsächlichen Ausführungszeiten der Systeme sind stark abhängig von der Infrastruktur, auf der die Systeme ausgeführt werden. Daher ist ihnen keine zu starke Bedeutung zuzusprechen. Die relativen Ausführungszeiten lassen sich jedoch miteinander vergleichen, um eine Einschätzung der Performance der Systeme zueinander zu erlangen. Die Inferenz des DeepDarts-Systems ist auf den DeepDarts-Daten um einen Faktor 4 schneller und bei den gerenderten Daten um den Faktor 2,6. Die Unterschiede liegen in den Arbeitsweisen der Systeme: DeepDarts verwendet ein neuronales Netz, dessen Ausführungszeit proportional zu den Eingabedaten skaliert, während die Bilddaten in dieser Thesis in einem Vorverarbeitungsschritt skaliert werden, um nahezu unabhängig von der Eingabegröße der Bilder zu sein. Die unterschiedlichen Ausführungszeiten zwischen DeepDarts-Daten und Render-Daten dieses Systems ergeben sich aus der minimalen Bildgröße dieses Vorverarbeitungsschritts, in welchem die Bilder zwischen 800 px und 1.600 px skaliert werden und damit über den Abmessungen der DeepDarts-Daten liegen.

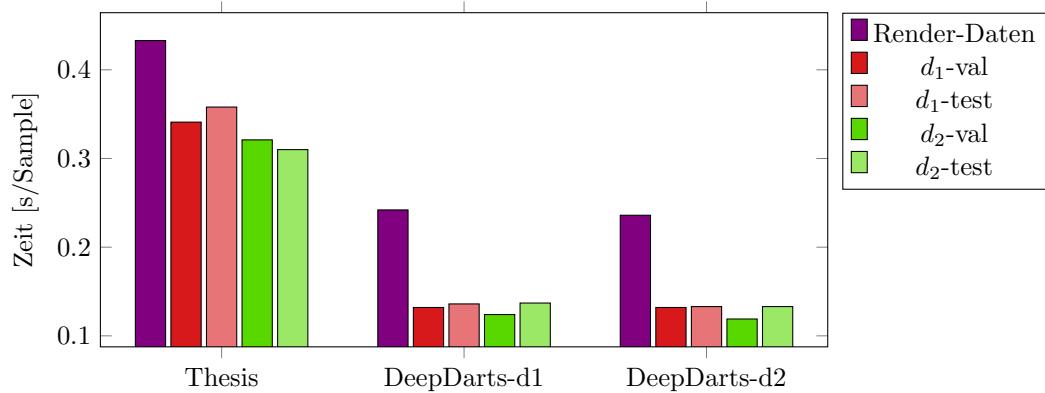


Abbildung 3.10: Dauer der Normalisierung auf unterschiedlichen Datensätzen, gruppiert nach Systemen.

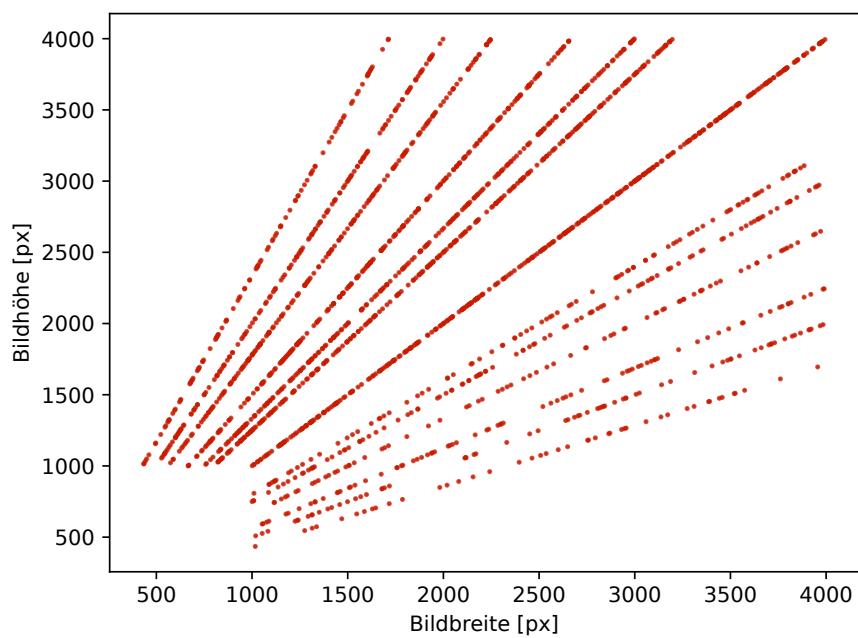


Abbildung 3.11: Verteilung der Seitenverhältnisse der gerenderten Bilder.

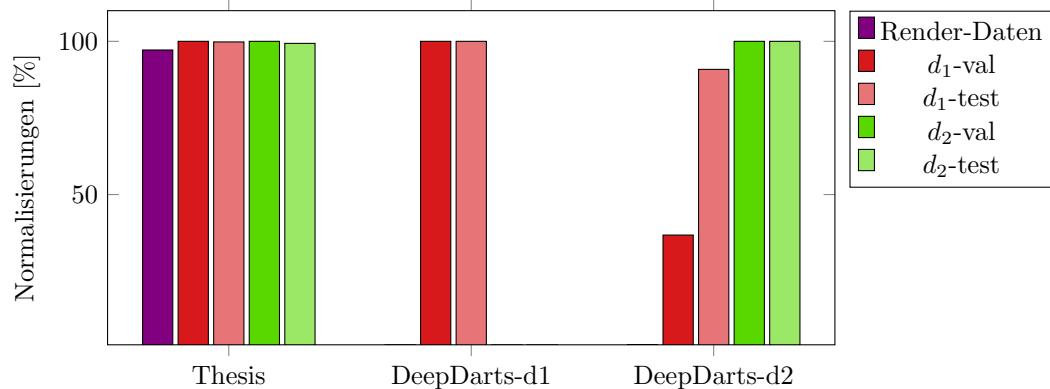


Abbildung 3.12: Auswertung der Fähigkeit unterschiedlicher Systeme, Normalisierungen für Daten zu finden.

3.4.3.2 Findung einer Normalisierung

In diesem Teil der Auswertung wird die Fähigkeit der Systeme betrachtet, eine Normalisierung der Bilder durchzuführen. Eine erfolgreiche Normalisierung bezieht sich für diese Auswertung lediglich darauf, ob ausreichend Orientierungspunkte für eine Normalisierung identifiziert werden konnten. Das DeepDarts-System muss dafür in der Lage sein, drei Orientierungspunkte zu identifizieren, da das System einem fehlenden Orientierungspunkt durch Interpolation ergänzt. Für das System dieser Thesis beinhaltet diese Anforderung die Lokalisierung des Mittelpunkts und mindestens drei weiterer Orientierungspunkte. Die Wahl der Orientierungspunkte ist dabei bei dem DeepDarts-System auf vier vordefinierte Punkte festgelegt, während das in dieser Thesis erarbeitete System 60 mögliche Punkte erkennen kann.

Die Ergebnisse dieser Auswertung sind in Abbildung 3.12 dargestellt. Die Auswertung ist sowohl hinsichtlich der Systeme als auch hinsichtlich der Datensätze aufgeteilt. Der Algorithmus dieser Thesis ist auf allen Datensätzen in der Lage, in mindestens 97 % der Bilder eine Normalisierung zu ermitteln. Die Performance ist dabei weitestgehend unabhängig von dem Ursprung der Daten. Dem gegenüber steht die Performance der DeepDarts-Systeme. Während DeepDarts- d_1 Auswertungen von 100 % auf den eigenen Validierungs- und Testdaten erzielt, ist es nicht in der Lage, positive Ergebnisse auf anderen Daten zu erzielen. Die Auswertung von DeepDarts- d_2 auf den eigenen Daten liegt ebenfalls bei 100 %. Die Auswertung auf den Validierungs- und Testdaten von DeepDarts- d_1 liegt bei 37 % und 91 %. Auf den gerenderten Daten werden lediglich für 2 % der Daten Normalisierungen identifiziert.

Diese Auswertung stärkt die Erkenntnis des Overfittings von DeepDarts und zeigt zugleich die Fähigkeit dieses Systems, ausreichend Orientierungspunkte zu finden, um eine Normalisierung zu ermöglichen.

3.4.3.3 Genauigkeit gefundener Normalisierungen

Die Genauigkeit der Normalisierungen wird mit μ_{xst} durchgeführt und die Ergebnisse sind in Abbildung 3.13 dargestellt. Es ist zu erkennen, dass signifikante Unterschiede zwischen den Systemen vorherrschen. DeepDarts- d_1 erzielt auf den ihm zugewiesenen Validierungs- und Testdaten gute Ergebnisse mit durchschnittlich 0,4 px Abweichung während es auf anderen Daten keine Ergebnisse erzielen kann. DeepDarts- d_2 hingegen kann auf allen Datensätzen Ergebnisse erzielen, jedoch sind deutliche Unterschiede zwischen den Quellen der Datensätzen zu erkennen. Auf DeepDarts-Daten können sehr gute Ergebnisse mit durchschnittlich 1,5 px Abweichung auf den d_1 -Daten und 1,1 px auf den d_2 -Daten erzielt werden. Auf den gerenderten Daten kann lediglich eine mittlere Abweichung von 1.568,7 px erzielt werden. Diese Beobachtung lässt darauf schließen, dass keine zuverlässige Normalisierung mit diesem System möglich ist, da die Bilder lediglich eine Größe von 800 × 800 px besaßen.

Das in dieser Thesis erarbeitete System ist in der Lage, auf allen Datensätzen Normalisierungen zu finden. Dariüber hinaus bewegen sich die mittleren Verschiebungen über die Datensätze in etwa ähnlichen Wertebereichen: Die Render-Daten konnten mit einer

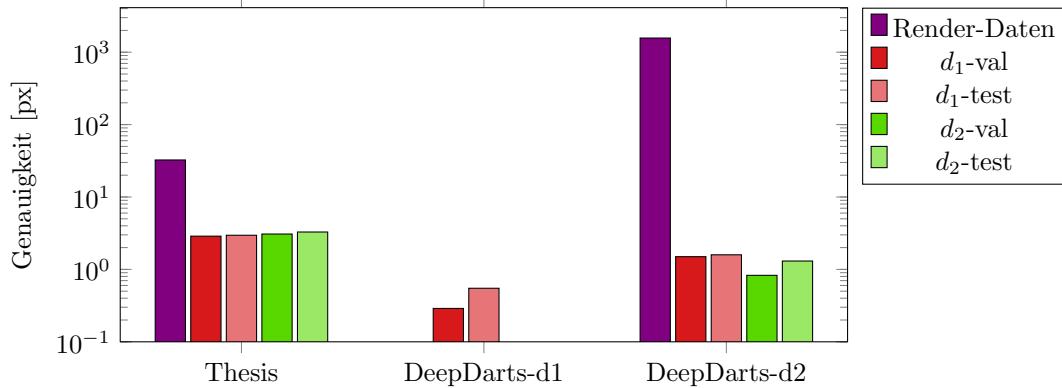


Abbildung 3.13: Genauigkeiten der Normalisierungen auf unterschiedlichen Datensätzen, gruppiert nach Systemen. Sofern keine Normalisierung möglich ist, existiert kein Balken.

mittleren Verschiebung von 17,2 px normalisiert werden und die DeepDarts-Daten mit 3,3 px. Die höhere Genauigkeit auf den DeepDarts-Datensätzen stammt von der Vorverarbeitung der Daten, sodass diese eine feste Größe besitzen. Da Skalierungen mit einem Informationsverlust einhergehen, steht die Anwendung dieser im Zusammenhang mit größeren Abweichungen der Auswertung.

3.4.3.4 Zusammenfassung der Auswertung

Die dargestellten Auswertungen zeichnen ein deutliches Bild der Arbeitsweisen und Genauigkeiten der unterschiedlichen Systeme. Während die DeepDarts-Systeme sehr gute Auswertungen auf den ihnen zugeschriebenen Daten erzielen, sind sie nicht in der Lage, auf ihnen unbekannte Daten zu generalisieren und ähnliche Ergebnisse zu erzielen. Das bereits in Abschnitt 1.2 erwähnte Overfitting wird durch diese Auswertung bereits verdeutlicht.

Die Inferenzzeit von DeepDarts ist geringer als die des in dieser Thesis erarbeiteten Systems. Hintergründe dafür können in den Implementierungen der jeweiligen Systeme gefunden werden. Während in dieser Thesis ein Algorithmus vorgestellt ist, dessen Implementierung in Python erfolgt, verwendet DeepDarts ein neuronales Netz, welches nahezu vollständig kompiliert ist und keinerlei Kontrollfluss wie Verzweigungen und Schleifen verwendet. Dadurch profitiert DeepDarts von starker Parallelität und effizienter Ausführung. Dieser Unterschied ist bei der Interpretation der Ergebnisse nicht außer Acht zu lassen.

Mit der Auswertung der Fähigkeit, Normalisierungen auf Daten zu finden, in Kombination mit der Genauigkeit dieser gefundenen Normalisierungen, ist hingegen ein wesentlicher Unterschied der Systeme erkennbar. Die DeepDarts-Systeme sind nicht in der Lage, Bilder sinngemäß zu normalisieren, welche nicht aus den für das Training verwendeten Daten stammen.

Kapitel 4

Identifizierung von Dartpfeilen mit neuronalen Netzen

An diesem Punkt in der Arbeit sind die Datenerstellung, bei der synthetische Bilddaten generiert werden, sowie die Vorverarbeitung der Bilder, durch die normalisierte Aufnahmen entstehen, erklärt und ausgewertet worden. Der dritte und letzte inhaltliche Themenbereich dieser Thesis umfasst die Lokalisierung der Dartpfeile in normalisierten Aufnahmen von Dartscheiben und das daraus resultierende Scoring der gespielten Runde. Diese Aufgabe wird durch die Verwendung neuronaler Netze angegangen, die auf Grundlage der in Kapitel 2 synthetisch erstellten Daten trainiert werden.

Ziel des Trainings ist es, ein System zu konzipieren, welches in der Lage ist, Dartpfeilspitzen exakt in Bildern zu lokalisieren und die getroffene Feldfarbe auszugeben. Anhand dieser Informationen können die getroffenen Felder und damit einhergehend die erzielte Punktzahl robust abgeleitet werden. In DeepDarts, dem Referenzsystem dieser Thesis, werden die getroffenen Felder ausschließlich anhand der relativen Positionen der Dartpfeile zu den Orientierungspunkten ermittelt. Diese Herangehensweise beruht darauf, dass Orientierungspunkte exakt lokalisiert werden können. Ist das nicht der Fall, werden Dartpfeile möglicherweise fehlerhaft identifiziert. Durch Ermittlung der getroffenen Feldfarbe ist ein gewisser Grad der Robustheit gegen Ungenauigkeiten in der Identifizierung der Orientierungspunkte bei der Normalisierung gegeben.

Die Unterteilung dieses Kapitels folgt dem bereits in Kapitel 2 und Kapitel 3 verwendeten Schema. Begonnen wird mit einer Erklärung von Grundlagen für das Verständnis der weiteren Unterabschnitte. Darauf folgt die Vorstellung der verwendeten Methodik hinsichtlich der Auswahl des Netzwerks und des Trainingsprozesses. Details zur Implementierung des neuronalen Netzes und des Trainings werden im Anschluss daran erläutert. Abschließend werden die Ergebnisse des Trainings und des resultierenden Gesamtsystems anhand unterschiedlicher Metriken ausgewertet.

4.1 Grundlagen

Zum Verständnis dieses Kapitels wird mit Grundlagen zu Konzepten und Begrifflichkeiten begonnen. Diese ermöglichen ein grundlegendes Verständnis, um die in diesem Kapitel eingesetzten Techniken nachzuvollziehen und den weiteren Unterkapiteln folgen zu können.

Begonnen wird mit der Klärung, was neuronale Netze sind und wie sie technisch funktionieren in Unterabschnitt 4.1.1. Dabei wird spezifisch auf bestimmte Arten neuronaler Netze und Arten von Vorhersagen, die in dieser Arbeit genutzt werden, eingegangen. Danach folgt ein Überblick über das Training neuronaler Netze in Unterabschnitt 4.1.2 und grundlegende Terminologie in Unterabschnitt 4.1.3. Im Anschluss darauf wird der Begriff der Augmentierung in Unterabschnitt 4.1.4 erklärt und es wird eine für diese Arbeit relevante Netzwerkarchitektur in Unterabschnitt 4.1.5 erläutert.

4.1.1 Was sind neuronale Netze?

Neuronale Netze sind Gegenstand eines spezifischen Bereichs des Machine Learnings, in dem sich auf das Erlernen von Verteilungen auf Grundlage von Daten fokussiert wird. Durch sie wird die Tür zur Approximation beliebiger Funktionen geöffnet, indem Resultate der zu erlernenden Funktionen gegeben werden [37]. Die Komplexität der Funktionen ist dabei beliebig, sodass die Spanne möglicher Einsatzbereiche von Sinuswellenapproximation bis zur Generierung natürlicher Sprache und Interaktion mit Menschen im Einsatzbereich von neuronalen Netzen liegt. Der für diese Arbeit relevante Teilbereich des Einsatzes neuronaler Netze ist die Extraktion von Informationen aus Bilddaten.

Hauptsächlich ausschlaggebend für den Erfolg des neuronalen Netzes ist seine Architektur, ihr innerer Aufbau. Herkömmliche neuronale Netze werden aus Aneinanderreihung von Schichten aufgebaut, die eingehende Daten transformieren und für die Verarbeitung subsequenter Schichten zur Verfügung stellen. Die Art der Schicht gibt die Spezifikation der Transformationen an, sodass unterschiedliche Schichten die Daten unterschiedlich verarbeiten. Innerhalb dieser Schichten existieren Parameter, die die Arbeitsweise der Transformation steuern. Durch Training des neuronalen Netzes werden die Parameter der Schichten derart angepasst, dass durch die schichtweise Verarbeitung der Eingabedaten die gewünschten Ergebnisse erzielt werden.

Der Name der neuronalen Netze leitet sich von im Gehirn vorzufindenden Neuronen ab, die für den Gedankenfluss verantwortlich sind. Das Erlernen von Parametern zur Steuerung von Ausgaben ist der Arbeitsweise von Neuronen nachempfunden. Der ursprüngliche Hintergedanke neuronaler Netze ist die technische Replikation der Arbeitsweise von Gehirnen.

4.1.1.1 Convolutional Neural Networks (CNNs)

Die Art der verarbeiteten Daten in einem neuronalen Netz kann viele Formen annehmen. Insbesondere die Verarbeitung von Bilddaten ist ein großer Themenbereich neuronaler Netze und essenziell für diese Thesis. Bereits in Unterabschnitt 3.1.4 wurde die Faltung auf Bilddaten eingeführt, die unter Verwendung von Kernen geschieht. Auf dieser Arbeitsweise fußen die Convolutional-Schichten. Die Parameter dieser Schichten bestimmen die Ausprägung eines Kernels, welcher auf eingehende Bilddaten angewandt wird. Durch vielfache Hintereinanderreihung von Convolutional-Schichten werden inkrementell komplexere Strukturen in Bildern identifiziert und abstrakt festgehalten [57]. Neuronale Netze, die auf Convolutional-Schichten aufbauen, werden als Convolutional Neural Networks (CNNs) bezeichnet [37].

4.1.1.2 Klassifizierung und Regression

Ebenso komplex wie Eingabedaten neuronaler Netze können ihre Ausgaben sein. Allgemein lassen sich Ausgaben von neuronalen Netzen in zwei Kategorien einteilen: Klassifikation und Regression [58, 37]. Bei der Klassifikation werden Datenpunkte in Form von Klassen vorhergesagt. Bei der Klassifizierung von Bildern sind Netzwerkausgaben der Klassen unterschiedlicher Objekte, Lebewesen oder Eigenschaften möglich (Beispielsweise die Beantwortung der Frage „Welches Tier ist in diesem Bild zu sehen?“). Ebenso ist eine binäre Klassifikation hinsichtlich der Existenz bestimmter Sachverhalte üblich (Beispielsweise die Beantwortung der Frage „Existiert eine Katze in diesem Bild?“). Die Ausgabe von neuronalen Netzen geschieht für diese Arten der Fragen typischerweise in Form von Vektoren, die diese Kategorien durch One-Hot-Encoding (oder 1-of-n-Encoding) darstellen [59]. Dabei ist jeder Kategorie ein Eintrag im Vektor zugeordnet; die Größe der Zahlenwerte geben die Sicherheit des Netzes für die jeweiligen Kategorien an.

Konträr zur Vorhersage diskreter Klassen ist die Vorhersage kontinuierlicher Werte als Ausgabe eines neuronalen Netzes möglich, bezeichnet als Regression. Beispiele für Ausgaben einer Regression beinhalten Funktionswerte oder Koordinaten. Ziel einer Regression ist es, konkrete Zahlenwerte vorherzusagen. Sofern eine Begrenzung der ausgegebenen Werte möglich ist, ist die Normalisierung von Daten in die Intervalle $[0, 1]$ oder $[-1, 1]$ üblich.

In dieser Thesis werden sowohl binäre wie auch klassenbezogene Klassifikation sowie Regression verwendet. Die binäre Klassifikation wird zur Identifizierung von Dartpfeilen genutzt, klassenbezogene Klassifikation zur Identifizierung von Feldfarben unter Dartpfeilen und Regression wird genutzt, um die exakten Positionen der Dartpfeile auf der Dartscheibe darzustellen.

4.1.2 Training neuronaler Netze

Das Training neuronaler Netze kann abhängig von der vorliegenden Art der Daten auf unterschiedliche Weisen verlaufen. In dieser Arbeit wird Supervised Learning verwendet, bei welchem Eingabedaten mit ihren zugehörigen Ausgaben gegeben sind. Das Netzwerk lernt auf Grundlage dieser Daten Parameter, welche die Verarbeitungsschritte des Netzwerks steuern, um die Ausgaben aus den Eingabedaten abzuleiten [58]. Supervised Learning basiert auf der Korrektur von Fehlern getätigter Vorhersagen des neuronalen Netzes. Der Fehler von Vorhersagen wird durch eine Metrik gemessen, die hinsichtlich der Parameter des Netzwerks differenzierbar ist. Diese Metrik wird als Loss-Funktion¹ bezeichnet, der Fehler des Netzwerks als Loss. Durch die Differenzierbarkeit ist ihr Gradient bekannt und kann genutzt werden, um lokale Minima zu identifizieren. Je geringer der Fehler ist, desto korrekter sind die Vorhersagen des Systems. Das Erreichen eines Minimums der Loss-Funktion ist das Ziel des Trainings. Die Identifizierung der Parameterangleichungen zur Annäherung an ein Minimum der Loss-Funktion geschieht durch einen Prozess, der als Backpropagation bezeichnet wird. Während der Backpropagation werden basierend auf dem Loss Parameter der Netzwerkschichten angepasst, um die Vorhersage für die gegebenen Daten derart zu korrigieren, dass zukünftige Vorhersagen auf den selben Daten einen geringeren Loss mit sich ziehen [37].

Die Implementierung der Backpropagation und die Umsetzung der Parameteranpassung geschieht durch Optimierungsalgorithmen. Die Arbeitsweise dieser Algorithmen ist grundlegend ähnlich hinsichtlich der Eingaben des Problems, in den konkreten Arbeitsweisen unterscheiden sie sich jedoch. Die Auswahl eines passenden Optimierungsalgorithmus ist abhängig von der zugrundeliegenden Aufgabe und der Netzwerkarchitektur.

4.1.3 Terminologie

Der Themenbereich der neuronalen Netze umfasst eine Vielzahl von Konzepten und Begriffen. Dieses Unterkapitel gibt einen Überblick über die zentralen Begriffe, die in dieser Arbeit von Bedeutung sind [58].

Trainingsdaten Trainingsdaten haben üblicherweise die größte Kardinalität aller für das Training und die Evaluation verwendeten Datensätze. Um ein effektives Training eines neuronalen Netzes zu gewährleisten, ist die Wahl der Trainingsdaten essenziell. Da der Trainingserfolg eines neuronalen Netzes abhängig von den Trainingsdaten ist und die durch die Parameter erlernten Strukturen in den Daten nicht bekannt sind, ist eine möglichst uniforme Abdeckung der zugrundeliegenden Daten wichtig. Jegliche Verzerrungen der Datenlage wird potenziell von dem neuronalen Netz erlernt und kann zu einer fehlerhaften Inferenz auf neuen Daten führen.

Validierungsdaten Validierungsdaten werden neben Trainingsdaten während des Trainings eines neuronalen Netzes genutzt, konträr zu Trainingsdaten haben diese jedoch keinen Einfluss auf den Trainingserfolg. In regelmäßigen Intervallen wird der aktuelle Stand der Netzwerkparameter auf den Validierungsdaten ausgewertet, um Einblicke in die Performance des Netzwerks auf Daten zu gewinnen, die nicht für das Training verwendet werden. Durch diese Vorhersagen können Rückschlüsse auf die Fähigkeit des neuronalen Netzes gezogen werden, das Gelernte auf neue Daten zu übertragen und die zugrundeliegenden Strukturen der Daten zu generalisieren. Die strikte Separierung von Trainings- und Validierungsdaten ist dabei obligatorisch, um eine verzerrte Ableitung der Fähigkeit zur Generalisierung zu vermeiden [58].

Die Wahl der Validierungsdaten unterliegt den gleichen Voraussetzungen wie den Trainingsdaten. Darüber hinaus sollten Validierungsdaten jedoch auf eine Art und Weise gewählt werden, die nicht zu große Ähnlichkeiten zu den Trainingsdaten aufweist, da diese Nähe der Daten ebenfalls eine Verzerrung der Datenlage mit sich ziehen kann, sofern keine uniforme Verteilung der Trainingsdaten vorliegt.

¹Alternativ wird diese Funktion auch als Cost- oder Error-Funktion bezeichnet. In dieser Arbeit wird die Terminologie der Loss-Funktion verwendet.

Testdaten Nach dem Training eines neuronalen Netzes werden Testdaten genutzt, um die Performance des trainierten neuronalen Netzes zu evaluieren. Während Trainings- und Validierungsdaten Einfluss auf den Verlauf des Trainings nehmen, werden Testdaten genutzt, um einen unabhängigen Einblick in die Netzwerkperformance nach Beendigung des Trainings zu gewinnen [58]. Durch Testdaten wird die Inferenz des trainierten Netzes auf unbekannten Daten simuliert, wodurch eine objektive Abschätzung der Generalisierbarkeit ermöglicht wird.

Für die Wahl der Testdaten sind die selben Voraussetzungen zu beachten, die für Trainings- und Validierungsdaten gelten, um einer Verzerrung der Datenlage vorzubeugen. Die Wahl von Trainings-, Validierungs- und Testdaten spielt für die Auswertung des neuronalen Netzes dieser Thesis eine wichtige Rolle.

Out-of-distribution-Training Dass die für das Trainings verwendeten Daten eine universelle Abdeckung über die gesamte Datenlage geben, ist häufig nicht möglich. Verzerrungen der Datenlage sind – gewollt oder ungewollt – in den meisten Fällen nicht zu umgehen. Weichen die Trainingsdaten jedoch signifikant von den Validierungs- und Testdaten ab, spricht man von Out-of-distribution (OOD)-Training. Das neuronale Netz wird auf Daten trainiert, die einer anderen Verteilung entsprechen als der zu erwartenden Daten für die Inferenz des Netzes. Dies kann Auswirkungen auf die Inferenz des Systems auf Daten haben, die anders strukturiert sind als die für das Training verwendeten Daten.

Under- und Overfitting Die Validierungsdaten eines Trainings werden verwendet, um den Erfolg eines Trainings zu beurteilen. Während Optimierungsalgorithmen darauf ausgelegt sind, den Loss auf Trainingsdaten zu minimieren, ist es möglich, dass der Validierungs-Loss von diesem abweicht. Befindet sich der Wert des Validierungs-Loss konsequent und signifikant über dem Trainings-Loss, spricht man von Underfitting [58, 37]. In dieser Situation ist das neuronale Netz nicht in der Lage, das Gelernte auf unbekannte Daten anzuwenden, da es noch nicht ausreichend trainiert ist. Fallen Trainings- und Validierungs-Loss zeitweise gleichermaßen, gefolgt von einem Anstieg des Validierungs-Losses, wird dies als Overfitting bezeichnet [37]. In dieser Situation werden Vorhersagen auf den Trainingsdaten besser, jedoch verliert das neuronale Netz die Fähigkeit der Generalisierbarkeit gelernter Strukturen auf neue Daten. Es werden nicht mehr die relevanten Aussagen hinter den Daten erlernt, sondern die konkreten Ausprägungen in den Trainingsdaten.

4.1.4 Augmentierung von Trainingsdaten

Für ein robustes Training eines neuronalen Netzes und für die Vermeidung von Overfitting ist eine große und möglichst umfangreiche Datenlage notwendig. Zur künstlichen Vervielfältigung der für das Training verwendeten Daten kann eine Technik genutzt werden, die als Augmentierung bezeichnet wird [37]. Unter Augmentierung wird eine Manipulation der Trainingsdaten bezeichnet, bei der die Datenmenge vervielfältigt wird, ohne die Integrität der Daten zu beeinträchtigen.

Beispiele für Augmentierung von Bilddaten sind die Anwendung affiner Transformationen oder das Hinzufügen von Rauschen. Sofern die Magnitude dieser Manipulationen nicht derart groß ist, dass die relevanten Aussagen der Bilddaten unterdrückt werden, ist die Erschaffung neuer Grunddaten aus Bildern mit bekannten Annotationen möglich. Ein robust trainiertes neuronales Netz ist in der Lage, die relevanten Informationen aus den Bildern zu extrahieren und eine Invarianz gegenüber der in der Augmentierung angewandten Operationen zu entwickeln. Bei dem Overfitting eines Netzwerks ist diese Invarianz gegenüber der Augmentierungsoperationen nicht gegeben.

Durch diese Technik wird eine größere Bandbreite möglicher Parameter und resultierend mögliche Eingaben in das System abgedeckt [60, 61]. Die Verwendung von Datenaugmentierung resultiert tendenziell in einer Verbesserung der Netzwerkperformance [62].

4.1.5 Die YOLOv8-Architektur

YOLOv8 ist eine Netzwerkarchitektur, die 2023 durch Ultralytics außerhalb des Rahmens einer gesonderten wissenschaftlichen Aufarbeitung veröffentlicht wurde². Mit YOLOv8 ist eine Netzwerkarchitektur vorgestellt, die Objekterkennung durch Anchor-Free-Erkennungsmechanismen in Echtzeit und in hoher Genauigkeit ermöglicht [63]. Sie wird als Verbesserung der YOLOv5-Architektur vorgestellt und ist für den Einsatz in Edge-Devices optimiert [64].

Der Aufbau von YOLOv8 folgt einem Design, welches in Backbone, Neck und Head unterteilt ist. Als Backbone wird eine erweiterte Version des CSPDarknet-Backbones verwendet während der Neck auf einem PANet basiert. Mit dem Aufbau von YOLOv8 wird das Paradigma des Fully Convolutional Neural Network (FCNN) verfolgt, welches die Verarbeitung von Bilddaten beliebiger Eingabegrößen ermöglicht.

Die Ausgaben des Netzwerks verfolgen einen Multi-Scale-Ansatz, durch welchen die Erkennung von Objekten unterschiedlicher Größen durch Rasterung der Eingabebilder in unterschiedliche Größen geschieht. Das Bild wird in Zellen unterschiedlicher Skalierungen unterteilt und es werden Vorhersagen für jede Zelle ausgegeben hinsichtlich Mittelpunkt eines Objekts, seiner Klasse und der Ausdehnung des Objekts über das Bild durch die Angabe einer Bounding Box. Die Ausgaben des Netzwerks werden durch etablierte NMS nachverarbeitet.

Der Einsatzbereich von YOLOv8 spezifiziert sich auf Echtzeitanwendungen, in welchen robuste und akkurate Objekterkennung gewünscht ist und in welchen die zur Verfügung stehenden Ressourcen für das Deployment dieses Netzwerks begrenzt sind. Beispiele solcher Anwendungen sind der Einsatz in autonomen Fahrzeugen oder zur Objekterkennung in Drohnen [65, 66].

4.2 Methodik

Im Anschluss an die Grundlagen widmet sich dieser Abschnitt der Methodik hinter der Konzeption und der Herangehensweise der Lokalisierung von Dartpfeilen durch ein neuronales Netz. In einem ersten Unterabschnitt wird der Aufbau und das Konzept der verwendeten Netzwerkarchitektur beschrieben. Insbesondere wird auf die Hintergründe und Besonderheiten der Netzwerkarchitektur eingegangen. Danach folgt eine Betrachtung der verwendeten Loss-Funktion für das Training. Diese ist spezifisch auf die Netzwerkarchitektur zugeschnitten, sodass ein optimales Training ermöglicht wird. Zuletzt wird das Netzwerktraining thematisiert. Es wird auf die Wahl und Zusammensetzung der Trainings- und Validierungsdaten eingegangen, Parameter werden erläutert und der Verlauf des Trainings wird dargestellt.

4.2.1 Die verwendete Architektur

Die Vorhersagen der Dartpfeilpositionen in normalisierten Bildern geschieht durch ein neuronales Netz der YOLOv8*-Architektur. Diese Architektur basiert auf YOLOv8 [67], welche für den Einsatz der Dartpfeilerkennung umstrukturiert wurde.

Dieses Unterkapitel thematisiert zunächst die Hintergründe der Wahl von YOLOv8 als Basismodell in Unterabschnitt 4.2.1.1. Danach werden in Unterabschnitt 4.2.1.2 vorgenommene Adaptionen an der Architektur aufgezeigt. Im Anschluss wird der konkrete Aufbau von YOLOv8* in Unterabschnitt 4.2.1.3 dargestellt.

4.2.1.1 Hintergründe zur Wahl von YOLOv8 als Basismodell

Die Architektur von YOLOv8 ist derart parametrisiert, dass unterschiedliche Netzwerkgrößen als Varianten vorgegeben sind. Die Größe und Komplexität des Netzwerks ist aufgeteilt in die Klassen n (nano), s (small), m (medium), l (large) und x (extra large). Diese Varianten unterscheiden sich in den Größen und Anzahlen der verwendeten Schichten und sind für unterschiedliche Einsatzsituationen ausgelegt. Während l- und x-Modelle für Umgebungen ausgelegt sind, in denen vorhandene Rechenleistung keinen Engpass darstellt, sind die s- und n-Modelle für den Einsatz in mobilen Geräten oder Edge-Devices vorgesehen, in denen die Ressourcen begrenzt sind. Die Verwendung von Netzwerken geringerer Größen geht mit Einbußen in der Qualität der Vorhersagen einher.

²Siehe: <https://github.com/ultralytics/ultralytics/issues/2572>

Zusätzlich zu den genannten Charakteristiken ist YOLOv8 ein optimierter Nachfolger der für DeepDarts verwendeten YOLOv4-Architektur. Durch den Einsatz von YOLOv4 in DeepDarts konnte die Fähigkeit dieser Familie der Netzwerkarchitekturen hinsichtlich der Erkennung von Dartpfeilen gezeigt werden.

Aufgrund des vorgesehenen Einsatzbereichs des Systems dieser Arbeit in mobilen Endgeräten ist die flexible Netzwerkgröße in Kombination mit einer hohen Qualität der Vorhersagen ausschlaggebend für die Entscheidung, diese Netzwerkarchitektur als Basismodell für die Ausarbeitung dieser Arbeit zu verwenden.

4.2.1.2 Adaption des Modells

Obgleich durch McNally et al. der Einsatz von YOLO-Architekturen zur Identifizierung von Dartpfeilen gezeigt wurde, ist die generelle Strukturierung der Architektur nicht optimal für die zugrundeliegende Aufgabe. Für eine Abstimmung der Architektur auf die Aufgabe werden strukturelle Änderungen im Netzwerkaufbau unternommen. Die Adaptierte Netzwerkarchitektur wird im Folgenden als YOLOv8* bezeichnet, um eine Differenzierung zu der offiziellen YOLOv8-Architektur herzustellen.

Bounding Boxes Im Wesentlichen ist die Verwendung von Ankerpunkten mit umliegenden Bounding Boxes bei der Lokalisierung von Objekten in Bildern effektiv und zielführend. Bei der Vorhersage spezifischer Positionen in einem Bild liefert die Verwendung von Bounding Boxes jedoch keinen Vorteil. Das DeepDarts-System projiziert quadratische Bounding Boxes auf die Dartpfeilspitzen, um Outputdaten zum Training des Netzwerks zu generieren. Die Positions berechnung der identifizierten Dartpfeile bezieht die Bounding Boxes nicht mit ein, da der im Mittelpunkt der quadratischen Bounding Box liegende Ankerpunkt die Position des Dartpfeils angibt. Die YOLOv8*-Architektur wird an ihren Einsatz angepasst, indem keine Vorhersagen über die Ausdehnung von Bounding Boxes von der Architektur getroffen werden.

Multi-Scale-Output In der YOLOv8-Architektur wird ein Multi-Scale-Output zur Identifizierung von Objekten unterschiedlicher Größen eingesetzt. Objekte werden in drei unterschiedlichen Kontextgrößen identifiziert; durch Nachverarbeitungsschritte werden diese Outputs miteinander kombiniert. Dieser Multi-Scale-Output ist in der YOLOv8*-Architektur nicht vorhanden, da für die zu identifizierenden Objekte durch die Normalisierung der Bild daten keine starken Größenvariationen zu erwarten sind. Zudem unterliegt eine kleine Kontextgröße der Gefahr der fehlerhaften Klassifizierung von Abnutzungen der Dartscheibe als Dartpfeil. Der Kontext des gesamten Dartpfeils ist zur Identifizierung seines Einstichpunktes notwendig; dieser ist durch die Normalisierung der Dartscheibe auf eine feste Größe von 800×800 px durch den größten Kontext gegeben.

Dreiteilungen der Outputs Die YOLOv8*-Architektur unterteilt das Eingabebild in Regionen und bestimmt die Existenz von Dartpfeilen je Region. Da eine typische Runde Darts aus drei Würfen besteht, ist davon auszugehen, dass eine maximale Anzahl von drei Dartpfeilen in den Bildern zu identifizieren ist. Dabei kann jedoch nicht ausgeschlossen werden, dass sich die Dartpfeilspitzen in unterschiedlichen Regionen befinden. Je Region wird daher eine feste Anzahl von drei möglichen Dartpfeilpositionen vorhergesagt. Existiert lediglich ein Dartpfeil, so sind zwei mögliche Outputs genutzt.

Transition-Blocks Eine grundlegende Änderung der YOLOv8-Architektur ist das Hinzufügen von Transition-Blocks. Diese befinden sich an den Übergängen zwischen Backbone und Head sowie Head und Detect. Sie brechen die Natur des FCNN der YOLO-Netzwerke durch Einbindung von Dense-Schichten. Die Eigenschaft von FCNNs, Bilder beliebiger Eingabegrößen verarbeiten zu können, geht mit der Einschränkung einher, keinen globalen Kontext des Bildes direkt einzufangen. Da die Eingabegrößen der Bilder durch die Vorverarbeitung vorgegeben sind, liefert die Verarbeitung beliebiger Eingabegrößen keinen Mehrwert und ermöglicht damit die Lockerung dieses Paradigmas. Folglich ist eine Einbindung von Dense-Schichten architektonisch möglich und ermöglicht einen globalen Überblick über das Eingabebild, anhand derer Informationen der gesamten Dartscheibe in untergeordnete Abschnitte des Netzwerks einfließen können. Die Einschränkung auf lokale Kontextfenster wird dadurch in der YOLOv8*-Architektur aufgehoben.

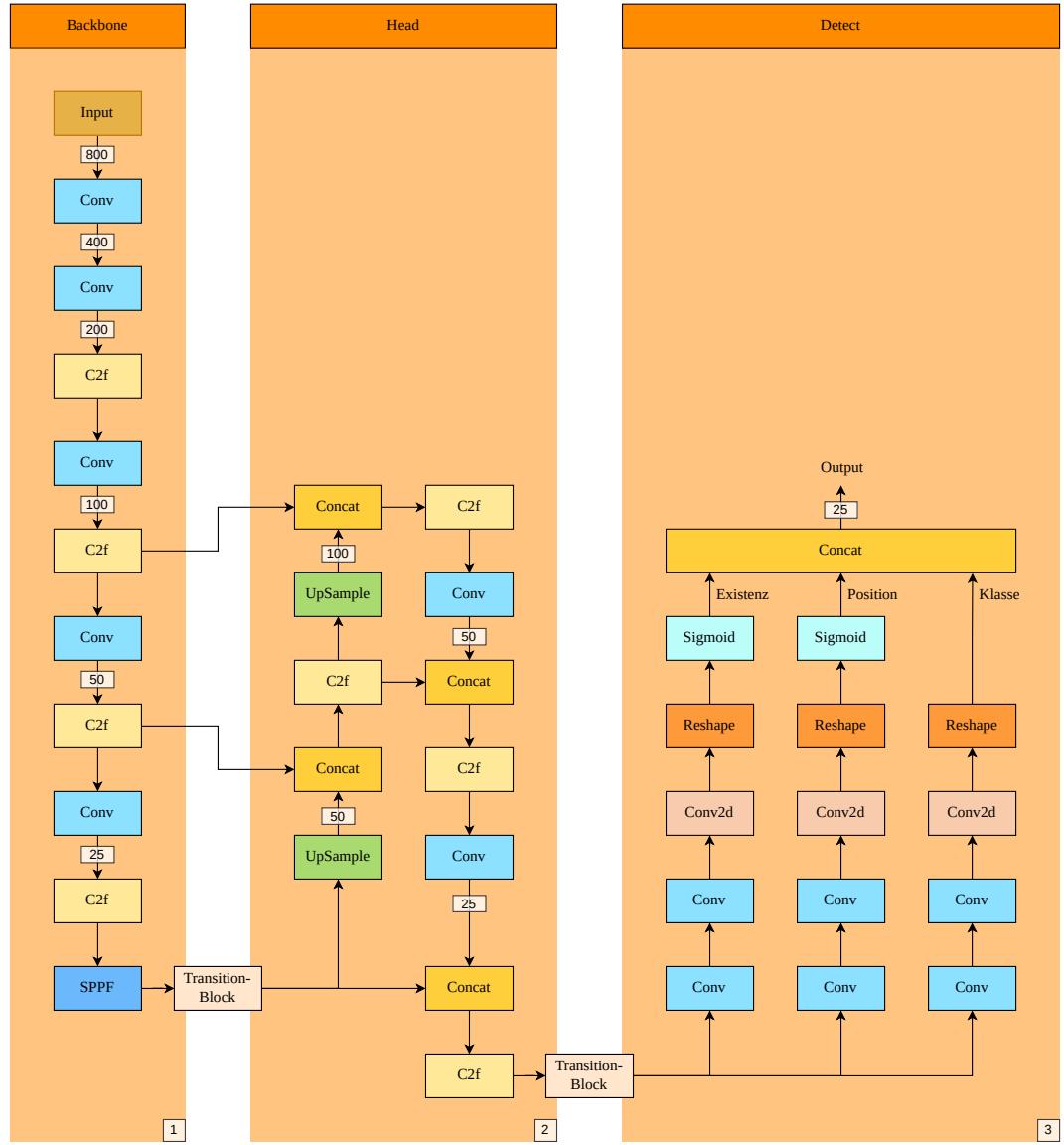


Abbildung 4.1: YOLOv8*-Architektur. (1) Bottleneck; Extraktion von Features. (2) Head; Kombination von Features. (3) Detect; Deutung von Features. Inspiriert durch Herfandi et al. [68]

4.2.1.3 Aufbau der Architektur

Der Aufbau der in dieser Thesis verwendeten YOLOv8*-Architektur ist in Abbildung 4.1 dargestellt. Sie ist unterteilt in die Bereiche Backbone, Head und Detect. Das Backbone und der Head sind weitestgehend analog zur YOLOv8-Architektur strukturiert. Der Detection ist ein gesonderter Netzwerkabschnitt zugewiesen, in welchem die Eliminierung des Multi-Scale-Outputs sowie die gesonderte Handhabung der Outputs manifestiert ist. Sie dazu unterteilt in drei parallele Stränge, in welchen Auswertungen zu Existenz, Position und Klasse abgeleitet und zu einem gemeinsamen Output konkateniert werden. Dieser Output besitzt eine Größe von $25 \times 25 \times 8 \times 3$, wie in Abbildung 4.3 dargestellt.

Die Netzwerkarchitektur setzt sich aus unterschiedlichen Blöcken zusammen. Grundlegende Blöcke sind: zweidimensionale Faltung (Conv2d), Max-Pooling (MaxPool2d), Addition (Add), zweidimensionale Normalisierung (BatchNorm2d), SiLU-Aktivierungsfunktion (SiLU) sowie Zweiteilung von Feature-Maps (Split) und Konkatenation von Feature Maps (Concat). Zusätzlich zu diesen bereits in YOLOv8 vorhandenen Grundblöcken werden Dense-Schichten (Dense), Dropout-Schichten (Dropout) und explizite Formänderungen der Tensoren (Reshape) ergänzt.

Konfiguration	Anzahl der Schichten	Trainierbare Parameter	Gesamtparameter
n	188	3,21 M	3,23 M
s	188	10,48 M	10,51 M
m	234	18,24 M	18,27 M
l	280	25,06 M	25,10 M
x	280	38,70 M	38,75 M

Tabelle 4.1: Parameter- und Schichtzahlen unterschiedlicher Konfigurationen der YOLOv8*-Architektur.

Aus diesen Grundblöcken werden weitere, größere Netzwerkblöcke zusammengesetzt, aus der die YOLOv8*-Architektur aufgebaut ist. Bereits in YOLOv8 enthaltene Blöcke sind SPPF, C2f und Bottleneck mit und ohne Shortcut. Adaptiert ist der Conv-Block, indem eine Dropout-Schicht hinzugefügt wurde. Als neuer Block ist der Transition-Block aufgenommen worden, der ähnlich zu dem Bottleneck-Block mit Shortcut aufgebaut ist, jedoch mit der Addition einer Dense-Schicht. Die zusammengesetzten Blöcke der YOLOv8*-Architektur sind in Abbildung 4.2 dargestellt.

4.2.1.4 Verwendete Konfiguration von YOLOv8*

Die variable Größe von YOLOv8 ist analog in YOLOv8* übernommen worden, sodass die Konfigurationen n, s, m, l und x möglich sind. Die verwendete Architektur dieser Thesis orientiert sich in ihrer Größenordnung an der in DeepDarts verwendeten Architektur YOLOv4-tiny, welche ca. 6 Millionen Parameter beinhaltete. Die Größenordnungen der Parametrisierungen sind in Tabelle 4.1 dargestellt. Aufgrund der im Vergleich zu DeepDarts komplexeren Daten wird sich für die Verwendung von YOLOv8*-s entschieden, welche mit etwa 10,5 Millionen Parametern etwa 40 % mehr Parameter besitzt als YOLOv4-tiny.

4.2.2 Loss-Funktionen

Die Loss-Funktion, die für das Training von YOLOv8* verwendet wird, ist nach dem Vorbild bereits existierender Loss-Funktionen für Netzwerke der YOLO-Familie konstruiert. Die Loss-Funktion setzt sich aus unterschiedlichen Teil-Losses zusammen, die zu einem gemeinsamen Loss kombiniert werden. Der Aufbau des gemeinsamen Losses durch die untergeordneten Losses folgt dabei der Architektur des Netzwerks.

4.2.2.1 Zusammensetzung des Losses

Die Loss-Funktion folgt dem Aufbau der YOLOv8*-Architektur. Sie ist definiert als gewichtete Summe aus Existenz-Loss, Klassen-Loss und Positions-Loss:

$$\mathcal{L}(y, \hat{y}) = \omega_{\text{xst}} \mathcal{L}_{\text{xst}}(y, \hat{y}) + \omega_{\text{cls}} \mathcal{L}_{\text{cls}}(y, \hat{y}) + \omega_{\text{pos}} \mathcal{L}_{\text{pos}}(y, \hat{y})$$

Existenz-Loss \mathcal{L}_{xst} Für den Existenz-Loss \mathcal{L}_{xst} wird der Focal-Loss der Existenz-Einträge aller Einträge jeglicher Regionen gebildet [69]. Die Verwendung des Focal-Loss für diesen Loss liegt in der spärlichen Menge positiver Einträge im Output-Tensoren. Bei einer Auflösung von 25×25 Regionen und drei Vorhersagen je Region besitzt der Output-Tensor $n_{\text{entries}} = 25 \times 25 \times 3 = 1.875$ Einträge. Zu erwarten sind $n_{\text{positive}} \leq 3$ positive Einträge, sodass der maximal zu erwartende Prozentsatz positiver Einträge $p_{\text{positive}} = \frac{n_{\text{positive}}}{n_{\text{entries}}} \leq \frac{1}{625} = 0,16\%$ beträgt. Der Focal-Loss gewichtet positive und negative Klassen, sodass ein gezieltes Erlernen trotz signifikanten Klassenungleichgewichts möglich ist.

Klassen-Loss \mathcal{L}_{cls} Der Klassen-Loss \mathcal{L}_{cls} beruht ebenso wie der Existenz-Loss auf dem Focal-Loss. Zur Berechnung des Losses werden die Regionen in Betracht gezogen, die einen Dartpfeil enthalten. Regionen ohne Dartpfeil werden nicht betrachtet, da diesen keine eindeutige Klasse zugeordnet werden kann. Es wird die kategoriale Focal-Kreuzentropie (Categorical Focal Cross-Entropy) [69] verwendet, um den Loss der für die Dartpfeile zugeteilten Klassen und den vorhergesagten Klassen in den jeweiligen Ziel-Regionen zu bestimmen. Die Klassen stehen für die Farben schwarz, weiß, rot und grün sowie einer Klasse zur Identifizierung des Außenbereichs, in dem keine Punkte erzielt werden (vgl. Abbildung 4.3).

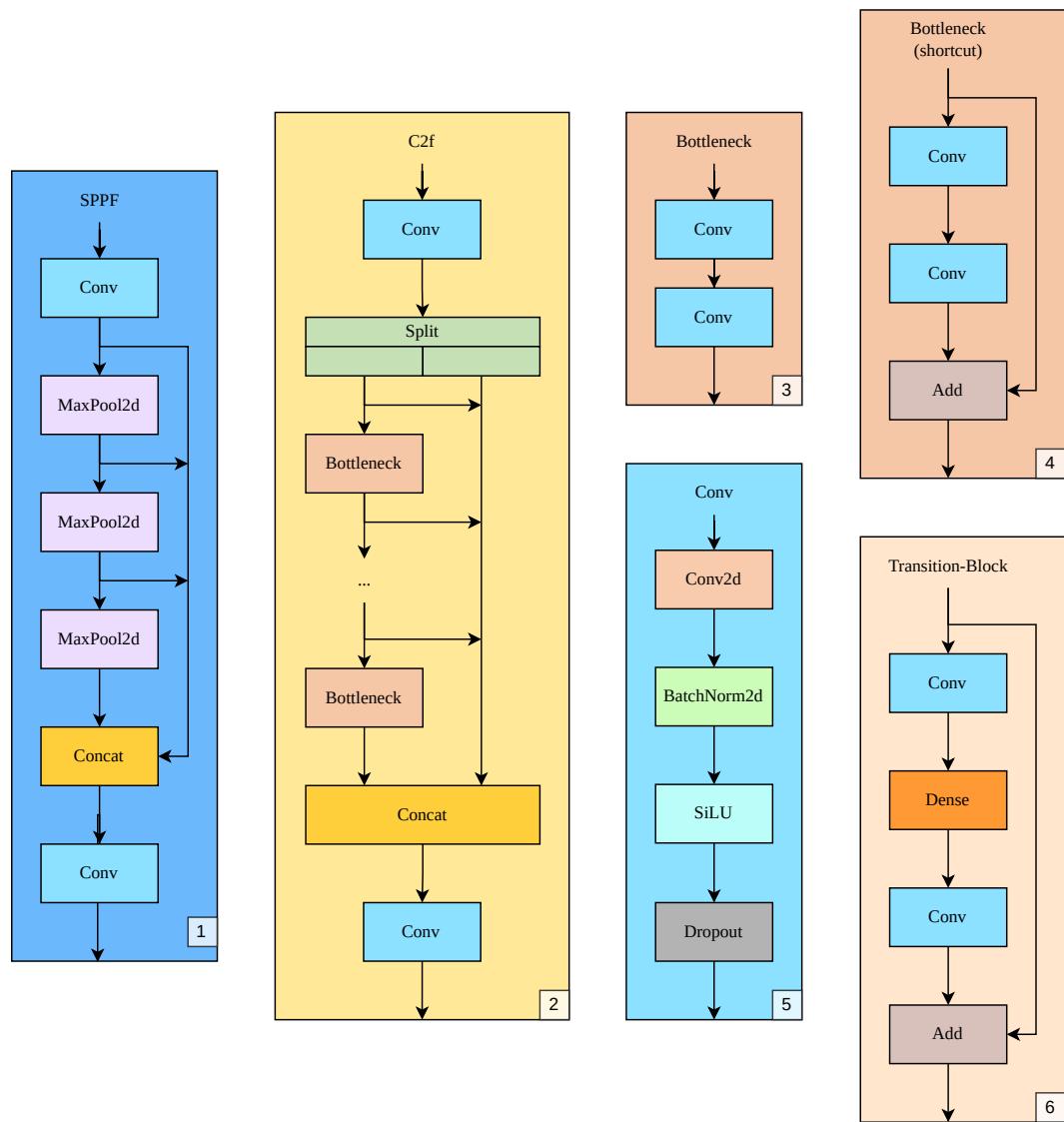


Abbildung 4.2: Netzwerkbestandteile der YOLOv8*-Architektur. (1) SPPF-Block; dieser zeichnet sich durch Hintereinanderreihung von MaxPool2d-Schichten aus. (2) C2f-Block; dieser spaltet die Feature Maps auf und wendet wiederholt Bottleneck-Blöcke an. (3, 4) Bottleneck-Block ohne und mit Shortcut; wiederholte Anwendung von Convolution, ggf. mit residualem Shortcut. (5) Conv-Block; dieser stellt den Grundbaustein der Convolution dar. (6) Transition-Block; Residualer Dense-Schicht zum Einfangen globaler Informationen.

Positions-Loss \mathcal{L}_{pos} Die Bestimmung des Positions-Losses \mathcal{L}_{pos} geschieht ebenso wie die Bestimmung von \mathcal{L}_{cls} unter Berücksichtigung der annotierten Existenzen. Je Region werden Positionen normalisiert relativ zur oberen linken Ecke angegeben. Die Differenz der normalisierten, lokalen Positionen der Vorhersagen und Wahrheitswerte werden in je x - und y -Komponente berechnet und aufsummiert. Diese kombinierte Summe wird durch die Anzahl der vorhandenen Dartpfeile geteilt, um einen Mittelwert der Positionsabweichung zu berechnen. Existieren keine Positionen, gilt $\mathcal{L}_{\text{pos}} = 0$.

Gewichtung Durch die Verwendung unterschiedlicher Losses für Existenz, Klasse und Position resultieren unterschiedliche Ausgabewerte der Losses. Die Loss-Gewichte ω_{xst} , ω_{cls} und ω_{pos} gewichten die Loss-Outputs derart, dass kein Loss wesentlich überwiegt und eine Reduktion des Losses \mathcal{L} eine uniforme Reduktion der Losses \mathcal{L}_{xst} , \mathcal{L}_{cls} und \mathcal{L}_{pos} mit sich zieht. Zum Anpassen der Losses aneinander für eine gleichwertige Konvergenz werden die gewichtete $\omega_{\text{xst}} = 400$, $\omega_{\text{cls}} = 2.000$ und $\omega_{\text{pos}} = 0,5$ verwendet.

4.2.2.2 Hintergründe und Zielsetzung

Der Hintergrund der Aufteilung der Loss-Funktion in Existenz, Klasse und Position liegt in der Netzwerkarchitektur. Diese ist auf die Vorhersage von Existenz, Klasse und Position ausgelegt, um Dartpfeile für ein Scoring zu lokalisieren. Die Kombination unterschiedlicher Losses mit eigenen Gewichten für je einen thematischen Bereich des Netzwerks ermöglicht ein ausgeglichenes und kontextbezogenes Training des gesamten Netzwerks. Auf diese Weise wird der Überschattung eines Teil-Losses durch einen anderen mit weitaus größerem Wert entgegengewirkt. Ein Overfitting eines Bereichs ist dadurch in dem kombinierten Loss eher widergespiegelt, sodass darauf dynamisch während des Trainings eingegangen werden kann.

4.2.2.3 Abweichung vom DIoU-Loss

Bei dem Training von YOLO-Architekturen ist die Verwendung vom IoU-Losses oder Adaptionen wie CIoU, DIoU oder GIoU [70], eine übliche Praxis [71, 67, 72]. Diese Erweiterungen des IoU-Losses stützen sich grundlegend auf der Annahme der Existenz von Bounding Boxes. Da diese nicht in der YOLOv8*-Architektur vorhanden sind, ist die Verwendung von IoU-basierten Loss-Funktionen obsolet. Es wurde jedoch mit einer Adaption des GIoU-Losses experimentiert, in der Quadrate vordefinierter Größe auf die vorhergesagten Positionen projiziert werden, anhand derer ein IoU-Loss berechnet werden kann. Durch diese Herangehensweise können viele Annahmen getroffen werden, durch die Optimierungen bezüglich Äquivalenz von GIoU-Loss und DIoU-Loss und effiziente Berechnung der Intersection-Area durch Distanzen der Punkte voneinander möglich sind. Da dieser Ansatz jedoch trotz Optimierungen mit einer großen Rechenleistung und starker Kongruenz zum Positions-Loss einherging, wurde diese Idee für das Training des Netzwerks verworfen.

4.2.3 Training

Das Training des neuronalen Netzes zielt auf das Erlernen der Identifizierung von Dartpfeilspitzen in normalisierten Eingabebildern ab. Das Training stützt sich dabei auf die Verwendung synthetisch generierter Trainingsdaten, die durch sporadische Anreicherung durch reale Daten erweitert und durch starke Augmentierungstechniken vervielfältigt werden. Der Aufbau der Trainingsdaten, die Arten und die Verwendung von Augmentierung sowie der Trainingsablauf werden in den folgenden Unterkapiteln thematisiert.

4.2.3.1 Trainingsdaten

Für das Training des DeepDarts-Systems wurden wenig diverse Daten verwendet, wodurch die Performance des Systems beeinträchtigt ist, wie bereits in Abschnitt 3.4 dargestellt. Um den einseitigen und wenig diversen Daten entgegenzuwirken, wird in dieser Thesis auf die Nutzung eigener, synthetisch generierter Daten gesetzt, die durch Salting realer Daten angereichert werden. Die Datenerstellung erfolgt nach dem in Kapitel 2 beschriebenen Prinzip.

Die überwiegende Mehrheit der Trainingsdaten wird durch generierte Daten ausgemacht, mit dem Ziel, durch diese ein grundlegendes Verständnis der zu lösenden Aufgabe

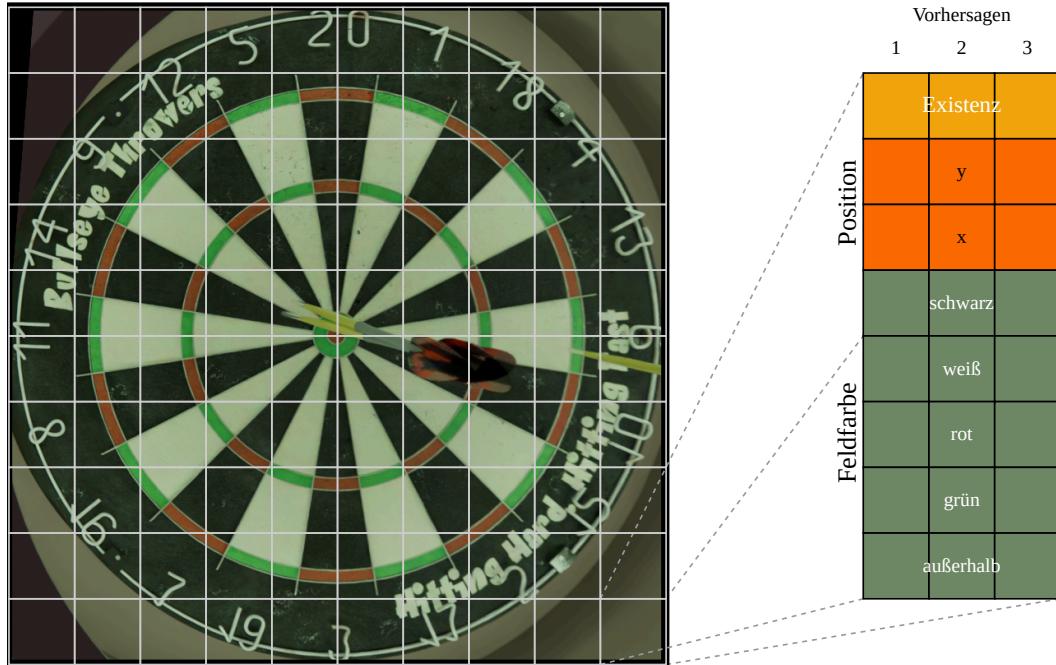


Abbildung 4.3: Schematische Darstellung des Output-Datenformats. Die Anzahl der Output-Zellen ist hinsichtlich der Übersichtlichkeit auf 10×10 Regionen begrenzt.

der Klassifikation von Existenz und Feldfarbe sowie der Regression von Positionen der Dartpfeilspitzen zu erlangen. Durch ein Salting mit realen Daten wird die Fähigkeit des Erlernens von Gegebenheiten realer Bilder ermöglicht. Die Datenquellen dafür setzen sich zusammen aus DeepDarts-Trainingsdaten und realen Aufnahmen, die für das Training dieser Arbeit aufgenommen und annotiert wurden. Daten, die zum Salting verwendet werden, besitzen zur Regulierung von Kardinalitätsunterschieden eine höhere Gewichtung als synthetische Daten.

Ein Training durch Supervised Learning setzt sowohl Inputs wie auch korrekte Outputs voraus, um die getätigten Vorhersagen des Netzwerks auf ihre Korrektheit zu überprüfen und die Netzwerkparameter durch Backpropagation zu adaptieren. Dazu sind einheitliche In- und Outputs notwendig. Die Inputdaten bestehen aus normalisierten 3-Kanal-Farbbildern im GBR-Farbformat mit Abmessungen von 800×800 px.

Die Outputdaten besitzen die Form $25 \times 25 \times 8 \times 3$. Das Input-Bild wird in 25×25 Regionen – entsprechend 32×32 px je Region –, für die je eine Matrix der Größe 8×3 vorhergesagt wird. Diese Matrix enthält Informationen zu Existenz von Dartpfeilspitzen der ihr zugewiesenen Region, die relative Position dieser sowie die getroffene Feldfarbe. Je Region können bis zu drei Dartpfeile identifiziert werden. Eine schematische Veranschaulichung der Datenstruktur wird in Abbildung 4.3 dargestellt.

4.2.3.2 Validierungsdaten

Zusätzlich zu den Trainingsdaten werden für das Training Validierungsdaten verwendet. Diese setzen sich ebenfalls aus unterschiedlichen Quellen zusammen, jedoch in gleichen Anteilen und resultierend ohne spezifische Gewichtung. Analog zu den Trainingsdaten stammen die Validierungsdaten aus einem Pool generierter Daten, Daten des DeepDarts-Datensatzes und manuell aufgenommen und annotierten Daten. Dabei wird auf eine strikte Trennung der Daten geachtet, sodass die generierten Daten gesondert gerendert, die DeepDarts-Daten aus einem separaten Teil der Daten mit sich unterscheidender Dartscheibe ausgewählt und die manuell aufgenommen Daten an einem anderen Ort aufgenommen wurden als die in dem Trainingssatz vorhandenen Daten. Auf diese Weise ist eine strikte Trennung der Trainings- und Validierungsdaten gegeben, durch die Verzerrungen durch Ähnlichkeiten zwischen Datensätzen minimiert werden während gleichzeitig unterschiedliche Quellen verwendet werden.

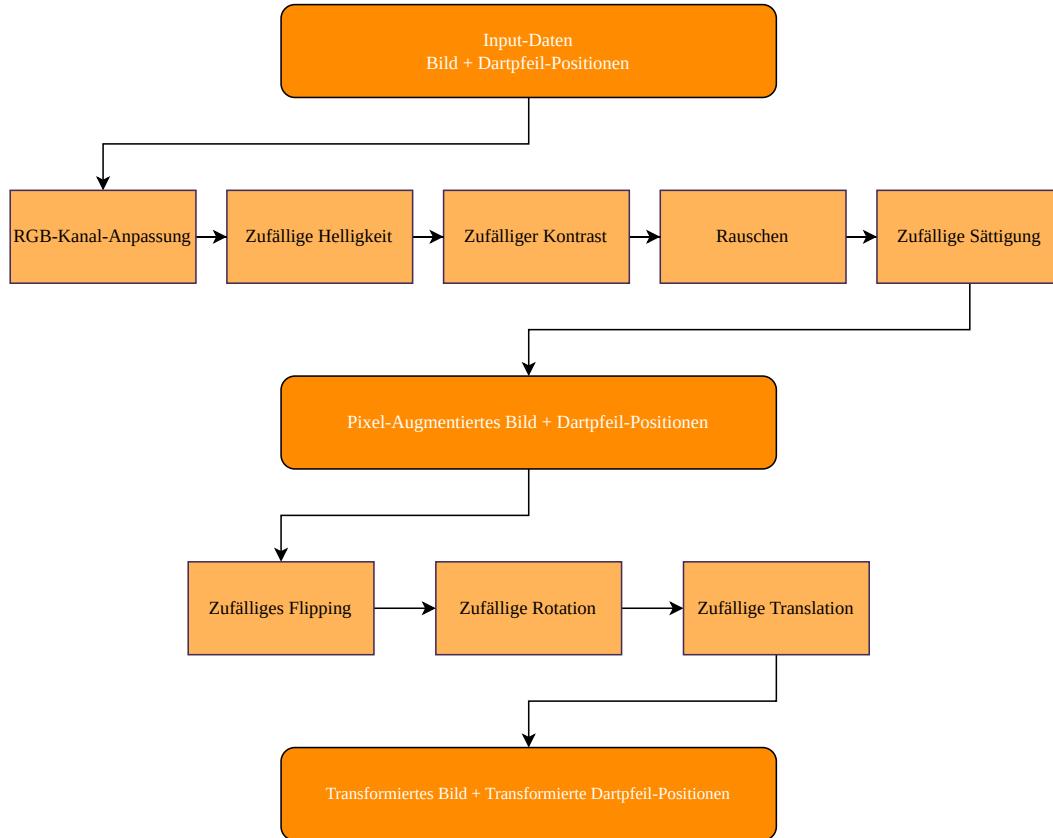


Abbildung 4.4: Schematische Darstellung der Augmentierung von Trainingsdaten.

4.2.3.3 Oversampling

Die Datenerstellung beruht auf der Verwendung von Heatmaps zur Positionierung der Dartpfeile auf der Dartscheibe, wodurch eine annähernd uniforme Verteilung über die gesamte Dartscheibe resultiert. Durch die Geometrie der Dartscheibe führt diese Art der Verteilung zu einer Ungleichheit in der Verteilung der Klassen schwarzer und weißer Felder im Vergleich zu roten und grünen Feldern. Um diesem Klassenungleichgewicht entgegenzuwirken, wird ein Oversampling roter und grüner Felder sowie ihrer unmittelbaren Umgebungen durchgeführt.

Durch das Oversampling ist ein übermäßiger Prozentsatz unterrepräsentierter Klassen in den Daten vorhanden, der zu einem ausgeglichenen Lernen aller Klassen führt [73]. Ohne Oversampling flächenmäßig kleiner Felder resultierten Vorhersagen von Double-, Triple- oder Bull-Treffern vermehrt in Fehlklassifizierungen als schwarze bzw. weiße Felder.

Trotz der Vorteile von Oversampling muss auf eine korrekte Einbindung dessen geachtet werden, um eine übermäßige Verzerrung der Datenlage zu verhindern [74]. Für diese Thesis wurden 24.576 Daten synthetisch erstellt, davon 20.480 reguläre Daten und 4.096 Oversampling-Daten. Der Prozentsatz des Oversamplings beläuft sich damit auf 16,67 % der generierten Daten.

4.2.3.4 Augmentierung

Erstrebenswert für die Erstellung von Trainingsdaten ist eine maximale Abdeckung aller plausibler und zu erwartender Parameter, die die Daten ausmachen. Insbesondere bei komplexen Daten – darunter auch Bilddaten – ist eine uniforme Abdeckung der Quellmenge an Input-Daten nicht möglich. Es ist daher davon auszugehen, dass die Trainingsdaten mit einer gewissen Verzerrung einhergehen.

Für die Augmentierung von Bilddaten existieren unterschiedliche Herangehensweisen [75]. Konkret werden in dieser Arbeit zwei Arten von Augmentierung auf die Trainingsdaten angewendet: Pixel-Manipulation und Transformation. Die Pipeline zur Augmentierung der Trainingsdaten ist in Abbildung 4.4 dargestellt.

Pixel-Manipulation Pixel-Manipulation bezeichnet die Änderung von Pixel-Werten der Eingabebilder, die keine Beeinflussung der Outputs mit sich zieht. Die Herangehensweise der eingesetzten Pixel-Manipulation ist in fünf Schritte aufgeteilt, die sukzessiv auf die Input-Daten angewandt werden. Die Manipulation beginnt mit einer Aufhellung oder Verdunklung einzelner Farbkanäle der Bilddaten, gefolgt von einer zufälligen Anpassung von Helligkeit und Kontrast. Danach werden die Bilddaten mit einem normalverteilten Gaußschem Rauschen angereichert und zuletzt wird die Sättigung des resultierenden Bildes randomisiert.

Das Analogon dieser Manipulation ist das Vorherrschen unterschiedlicher Beleuchtungen und Kamera-Parameter bezüglich ISO, Kontrastverhalten und Über- und Unterbelichtung, sowie variierende Qualität der Kameraaufnahmen. Es ist klar zu erkennen, dass diese Art der Augmentierung keinen Einfluss auf die Outputs der Daten nimmt, da lediglich die Repräsentation der Pixel, jedoch nicht die Aussage des Bildes abgeändert wird.

Transformation Entgegen der Pixel-Manipulation nimmt die Augmentierung durch Transformation Einfluss auf die Output-Daten, indem affine Transformationen auf die Daten angewandt und somit die Positionen der Dartpfeilspitzen in den Bildern manipuliert werden. Die Augmentierung durch Transformation besteht aus drei Schritten: Flipping, Rotation und Translation. Bei dem Flipping wird das Bild zufällig horizontal und vertikal gespiegelt. Bei der Rotation wird das Bild um Ganzzahlige Vielfache von 18° um den Mittelpunkt rotiert. Diese Rotationsinkremente entsprechen den Feldlinienwinkel der Dartscheibe und sorgen dafür, dass die Wertigkeit der Dartfelder rotiert, jedoch die Positionen der Feldlinien erhalten bleiben, welche durch die in Kapitel 3 vorgestellte Vorverarbeitung festgesetzt sind. Zuletzt wird das Bild durch eine Translation in x - und y -Richtung um wenige Pixel verschoben, um Ungenauigkeiten des Normalisierungsalgorithmus zu simulieren.

Diese Augmentierungen werden dynamisch beim Einlesen der Daten angewandt, sodass jedes Sample in jeder Epoche des Trainings unterschiedlich augmentiert wird. Durch diese Technik wird Overfitting durch Auswendiglernen der Daten entgegengewirkt und die Fähigkeit des neuronalen Netzes zur Generalisierung der Problemstellung wird verstärkt. Das Augmentieren ermöglicht das Fokussieren auf die Extraktion relevanter Kerninformationen in den Daten anstelle oberflächlicher Erscheinungsbilder.

4.2.3.5 Dynamisches Training

Die Geschwindigkeit des Trainings wird durch die Learning Rate gesteuert. Diese beschränkt den Grad des Einflusses der durch die Backpropagation ermittelten Änderungen auf die Netzwerkparameter. Eine hohe Learning Rate sorgt für rapide und starke Änderungen während eine geringe Learning Rate eine geringe Änderung und langsame Konvergenz der Netzwerkparameter mit sich zieht. Für das Training in dieser Thesis wurde eine dynamische Learning Rate verwendet, die basierend auf dem Verlauf des Trainings gesetzt wurde und auf eine gezielte Konvergenz ausgelegt ist [76].

Anstelle eines vorprogrammierten Schedules einer Learning Rate wurde sich vor dem Hintergrund des optimalen Lernerfolgs für eine manuelle Anpassung der Learning Rate entschieden. Initial wurde die Learning Rate auf den Wert $lr_0 = 0,001$ gesetzt. Sofern der Trainingserfolg nach subjektiver Einschätzung zu stagnieren begann, wurde die Learning Rate um den Faktor $f_{lr} = 0,1^{0,25}$ angepasst. Durch diesen Wert erfolgt eine Minderung der Learning Rate um den Faktor 10 nach vierfacher Verringerung.

4.2.3.6 Trainingsablauf

Für das Training wurden die erwähnten Techniken und Optimierungen eingesetzt. Als Optimizer wurde sich für AdamW entschieden, der sich durch adaptive Gradientenanpassung auszeichnet und empirisch für solide Generalisierungsfähigkeiten bekannt ist [77, 78, 79, 80, 81]. Zudem konnte bereits gezeigt werden, dass die YOLOv8-Architektur erfolgreich mit diesem Optimizer trainiert werden konnte [82]. Das Training verlief über 500 Epochen und der Verlauf ist in Abbildung 4.5 dargestellt. Die Abbildung ist unterteilt in die Teil-Losses der Existenz, Klassen und Positionen sowie den kombinierten Loss. Trainings- und Validierungsloss fallen tendenziell gemeinsam, jedoch ist eine größere Diskrepanz der jeweiligen Werte

bei \mathcal{L}_{xst} zu verzeichnen als bei \mathcal{L}_{cls} und \mathcal{L}_{pos} . Die Verläufe der Validierungs-Losses deuten jedoch weder auf Over- noch auf starkes Underfitting hin.

In den Graphen ist ein Sprung der Losses um Epoche 400 zu vermerken, welcher durch die manuelle Adaption der Learning Rate bedingt ist. Um einem lokalen Minimum des Losses entgegenzuwirken, wurde die Learning Rate erhöht und nach einigen Epochen inkrementell verringert. Diese Technik ist als Warm Restart bekannt und kann zu einer Verbesserungen der Losses führen [83]. In diesem Fall konnte eine Verbesserung des Losses durch den Einsatz dieser Technik erzielt werden.

4.3 Implementierung

Nach der Darstellung der Methodik wird in diesem Abschnitt auf Details zur Implementierung eingegangen. Dieser Abschnitt ist unterteilt in zwei Unterabschnitte: Im ersten Unterabschnitt werden konkrete Details zur Implementierung der YOLOv8*-Architektur thematisiert, der zweite Unterabschnitt befasst sich mit den spezifischen Aspekten des Trainings.

4.3.1 Implementierung der YOLOv8*-Architektur

Vortrainierte neuronale Netze der YOLO-Familie werden als Modelle veröffentlicht, die mit dem Framework PyTorch erstellt sind. Für diese Thesis wird TensorFlow als Framework für die Erstellung und das Training neuronaler Netze verwendet, welches nicht reibungslos mit PyTorch vereinbar ist. Die Frameworks arbeiten auf unterschiedliche Arten, wodurch eine Übersetzung der verwendeten Schichten und der vortrainierten Gewichte zwischen diesen lediglich bedingt möglich ist. Durch die Abwandlung von YOLOv8* zu YOLOv8 ist eine eigene Implementierung notwendig und eine Einbettung vortrainierter Gewichte in diese Architektur ist nicht trivial möglich. Auf Grundlage der offiziellen Dokumentation sowie des Quelltextes und Konfigurationsdateien wurde die YOLOv8-Architektur in TensorFlow übersetzt und implementiert. Atomare Bestandteile wie Convolution, Batch-Normalisierung, Pooling-Operationen und Aktivierungsschichten können analog von PyTorch zu TensorFlow übertragen werden. Schichten, die nicht in TensorFlow verfügbar sind – beispielsweise die Split-Operation – wurden durch Einbindung eigener Schichten implementiert.

Nachdem alle grundlegenden Bestandteile verfügbar waren, wurden aus diesen kombinierte Netzwerkbestandteile zusammengesetzt, dargestellt in Abbildung 4.2. Die Dimensionierungen der jeweiligen Schichten sowie die verwendeten Parameter konnten aus der Dokumentation der Architektur entnommen und analog übertragen werden. Nachdem alle Bestandteile der Architektur implementiert waren, wurde eine generische Implementierung der Architektur vorgenommen, in der die bereitgestellten Parameter für unterschiedliche Größenvariationen von YOLOv8 mit einbezogen wurden. Durch den Vergleich der Anzahl der Netzwerkparameter unterschiedlicher Größenkonfigurationen sowie der Analyse des Netzwerkaufbaus durch Verbindungen der Schichten konnte sichergestellt werden, dass die TensorFlow-Implementierung von YOLOv8 der vorgestellten Architektur entsprach.

Nachdem die grundlegende Architektur in TensorFlow verfügbar war, wurde sie durch eigene Adaptionen erweitert. Eine grundlegende Änderung ist das Hinzufügen von Dropout-Schichten, welche nicht in der Dokumentation erwähnt sind, jedoch die Stabilität des Trainings erhöhen und die Wahrscheinlichkeit des Overfittings senken. Eine weitere Adaption ist das Hinzufügen von Transition-Blocks, welche eine residuale Dense-Schicht umfassen. In einem Transition-Block wird der Eingabetensor durch einen Tensor moduliert, der den globalen Kontext des Eingabetensors durch eine Dense-Schicht einfängt.

4.3.2 Training von YOLOv8* zur Identifizierung von Dartpfeilen

In diesem Unterabschnitt wird das Training des neuronalen Netzes thematisiert. Es wird begonnen mit der verwendeten Infrastruktur und Rahmenbedingungen des Trainings. Danach folgt eine detaillierte Betrachtung der verwendeten Augmentierungsparameter. Zuletzt wird der Verlauf des Trainings erläutert.

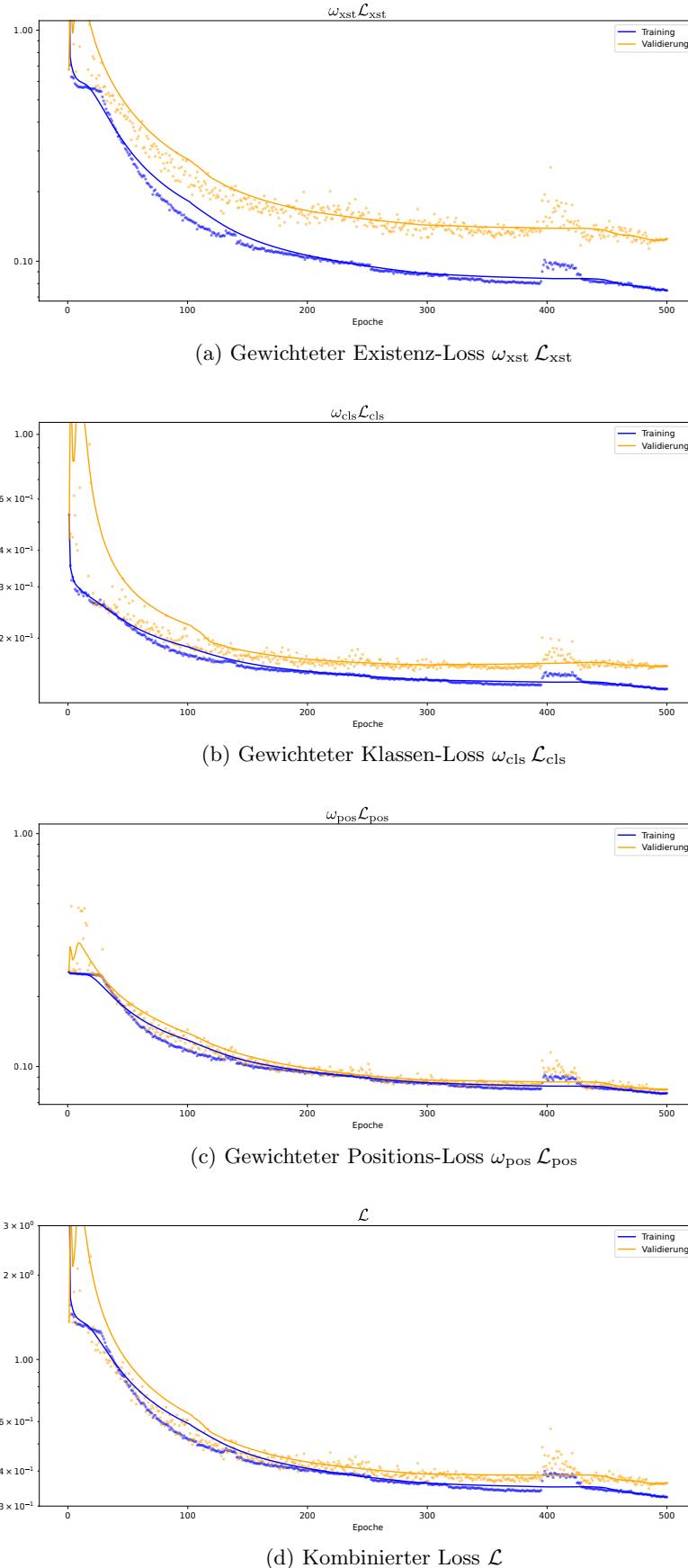


Abbildung 4.5: Trainingsverlauf der YOLOv8*-Architektur. Die Loss-Werte sind als Punkte dargestellt, welche durch eine Linie der entsprechenden Farbe geglättet dargestellt sind. Trainings-Losses werden in blau dargestellt, Validierungs-Losses in orange.

4.3.2.1 Infrastruktur und Rahmenbedingungen

Das Training von YOLOv8* wurde auf einer NVIDIA GeForce RTX 4090 aus einem TensorFlow-Docker-Container ausgeführt. Es wurde eine Batch-Size von 32 verwendet mit dem AdamW-Optimizer und einer dynamischen Learning Rate, wie in Unterabschnitt 4.2.3.5 erläutert. Für das Training wurden 24.960 Trainings- und 672 Validierungsdaten verwendet. Die Trainingsdaten setzen sich zusammen aus 24.576 generierten Daten (20.480 Daten auf Grundlage regulärer Heatmaps und 4.096 Daten auf Grundlage von Multiplier-Heatmaps), 256 Daten des DeepDarts- d_1 -Trainingsatzes und 128 manuell aufgenommenen Daten. Die Validierungsdaten sind zusammengesetzt aus 256 synthetischen Daten, 256 Daten der DeepDarts- d_2 -Trainingsdaten sowie 160 manuell aufgenommenen Daten. Die Trainings- und Validierungsdaten unterliegen einer strikten thematischen Trennung, sodass die realen Datensätze in unterschiedlichen Umgebungen aufgenommen wurden. Auf diese Weise wird Darstellung der Generalisierbarkeit durch den Validation-Loss nicht durch eine vorteilhafte Datenlage verzerrt.

4.3.2.2 Augmentierungsparameter

Die Daten werden vektorisiert als TensorFlow Datasets eingelesen, wodurch ein effizientes und parallelisiertes Laden der Daten ermöglicht wird. Ein Schritt des Einlesens der Daten ist die Augmentierung, welche dynamisch auf jedes Bild der Trainingsdaten mit zufälligen Parametern angewendet wird. Die Farbkanäle für rot, grün und blau werden unabhängig voneinander mit einem zufälligen, uniform gewählten Gewicht $A_{\text{cha},\{r,g,b\}} \in [0, 5..1]$ moduliert. Die Helligkeit des Bildes wird durch den Parameter A_b bestimmt, der die mittlere Helligkeit des Bildes zufällig uniform verteilt in dem Intervall $[\min(0,1, b_{\text{img}} - b_{\text{adj}}), b_{\text{img}} + b_{\text{adj}}]$ setzt, wobei b_{img} die mittlere Helligkeit des Bildes ist und $b_{\text{adj}} = 0,03$ die maximale Änderung der Helligkeit vorgibt. Der Kontrast des Bildes wird durch den zufällig uniform gewählten Parameter $A_{\text{cont}} \in [0, 7..1, 3]$ bestimmt und die Farbsättigung wird um einen ebenfalls uniform verteilten Faktor $A_{\text{sat}} \in [0, 8..1, 2]$ moduliert. Das Hinzufügen von normalverteiltem Rauschen auf jeden Pixel jedes Farbkanals findet mit einer Normalverteilung $\sigma_{\text{noise}} = 0,15$ statt. Hinsichtlich der transformativen Augmentierungsparameter wird das Bild horizontal und vertikal mit einer Wahrscheinlichkeit von je 50 % gespiegelt. Eine Rotation um den Mittelpunkt des Bildes erfolgt zufällig um einen Winkel aus der Menge $A_{\text{rot}} \in \{i \in \mathbb{N} \mid i = n \cdot 18^\circ, n \in \mathbb{N}_{<20}\}$. Zuletzt erfolgt eine normalverteilte Translation des Bildes $A_{\text{trans}} \in \mathbb{R}^2$ mit einer Standardabweichung von $\sigma_{\text{trans}} = 5 \text{ px}$.

4.4 Ergebnisse

Im Zentrum dieses Abschnitts steht die Auswertung des in den vorherigen Abschnitten erläuterten neuronalen Netzes und seines Trainings. Es werden sowohl absolute Ergebnisse betrachtet sowie in Relation gesetzte Ergebnisse hinsichtlich der Referenzsysteme DeepDarts- d_1 und DeepDarts- d_2 . Die Auswertung geschieht auf Grundlage unterschiedlicher Metriken, die teilweise spezifisch für diese Auswertung konzipiert sind, sowie den PCS, welcher für die Auswertung von DeepDarts entwickelt wurde. Durch erneutes Aufgreifen des PCS ist einerseits die Verifizierung der korrekten Verwendung der DeepDarts-Systeme überprüfbar, andererseits kann einer Verzerrung der Darstellung von Ergebnissen durch eine voreingenommene Wahl von Metriken entgegengewirkt werden.

Dieser Abschnitt ist unterteilt in mehrere Bereiche. Zuerst werden die Metriken und die zur Auswertung verwendeten Datensätze erläutert. Anschließend werden die Ergebnisse der Auswertung dieser Metriken auf den beschriebenen Daten dargestellt. Abschließend werden die Ergebnisse des PCS dargestellt, anhand derer Rückschlüsse auf die Auswertung von DeepDarts geschlossen werden können.

4.4.1 Metriken

Für die Auswertung der Genauigkeit der jeweiligen Systeme werden mehrere Metriken verwendet. Zur Auswertung von DeepDarts wurde der in Abschnitt 1.2 beschriebene PCS verwendet, um die relative Anzahl korrekt vorhergesagter Daten zu bestimmen. Diese Metrik ist jedoch dahingehend fehleranfällig, dass False Positives zustande kommen können. PCS

misst die Fähigkeit, die korrekte Punktzahl vorherzusagen, statt der Fähigkeit, die Dartpfeile korrekt zu ermitteln. Um einen Einblick in die Fähigkeiten der Systeme zu gewinnen, werden in dieser Arbeit drei weitere Metriken verwendet: Existenz-Metrik μ_{xst} , Klassen-Metrik μ_{cls} und Positions-Metrik μ_{pos} .

4.4.1.1 Existenz-Metrik μ_{xst}

Mit dieser Metrik wird bestimmt, ob die korrekte Anzahl der Dartpfeile bestimmt wird. μ_{xst} ist definiert als:

$$\mu_{xst} = \frac{1}{N} \sum_{i=1}^N 1 - \left| \frac{1}{3} \cdot (N_{Dart, i} - \hat{N}_{Dart, i}) \right|$$

In dieser Formel stehen $N_{Dart, i} \in \mathbb{N}$ und $\hat{N}_{Dart, i} \in \mathbb{N}$ für die Anzahl vorhandener und vorhergesagter Dartpfeile je Bild mit Index i . Anhand des Werts von μ_{xst} wird ermittelt, wie die Anzahl der zu ermittelnden Dartpfeile zu der Vorhersage der Dartpfeile vergleichbar ist. Ohne weiteren Kontext gibt diese Metrik keinen Aufschluss über die Korrektheit der Vorhersagen der Dartpfeile aus. Eine Korrelation zwischen Existenz und Position von Dartpfeilen wird in dieser Metrik nicht festgehalten.

4.4.1.2 Klassen-Metrik μ_{cls}

Die Klassen-Metrik μ_{cls} betrachtet die Korrektheit der vorhergesagten Klassen der Dartpfeile. Für diese Metrik wird ein Matching vorgenommen, anhand dessen die Klassen vorhergesagter Dartpfeile mit den Klassen existierender Dartpfeile verglichen werden:

$$\mu_{cls} = \frac{1}{N} \sum_{i=1}^N \frac{1}{3} \cdot N_{K,correct,i}$$

$N_{K,correct,i} \in \mathbb{N}$ beschreibt die Anzahl korrekt vorhergesagter Klassen in dem Bild mit Index i . Das Matching der Klassen wird mit einem Greedy-Algorithmus durchgeführt, in welchem zusätzlich erkannte Klassen verworfen werden. Diese Ungenauigkeit der Metrik wird durch die Kombination mit der Metrik μ_{xst} ausgeglichen.

4.4.1.3 Positions-Metrik μ_{pos}

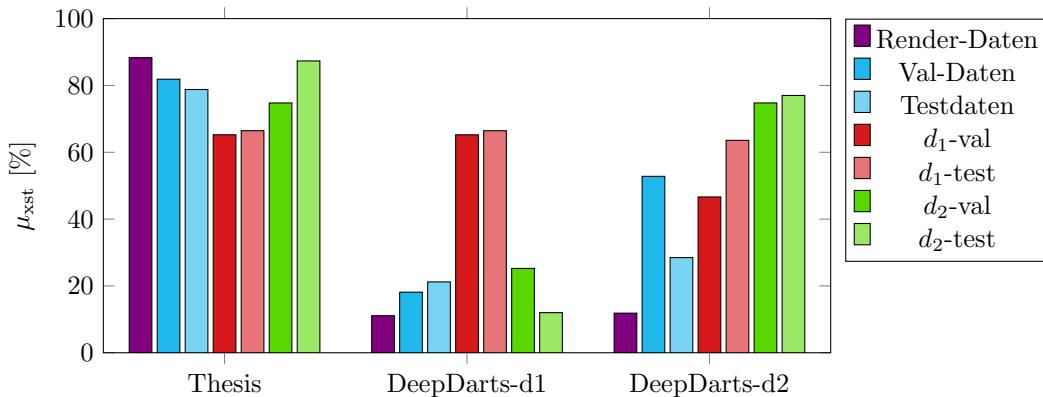
Ziel dieser Metrik ist es, die durchschnittlichen Abweichungen der Dartpfeilspitzen einzufangen. Die Dartpfeilspitzen werden analog zu μ_{cls} durch ein Greedy-Matching korreliert, indem die vorhandenen und vorhergesagten Dartpfeilpositionen mit den je geringsten Abständen zueinander gepaart werden, sofern sie noch nicht gepaart wurden. Diese Metrik gibt einen Einblick in die Präzision, mit welcher Dartpfeilspitzen erkannt werden. Der Wert von μ_{pos} ist definiert durch:

$$\mu_{pos} = \frac{1}{N} \sum_{i=1}^N \sum_{d=1}^3 \left\| P_{i,d} - \hat{P}_{i,d} \right\|_2$$

$P_{i,d} \in \mathbb{R}^2$ und $\hat{P}_{i,d} \in \mathbb{R}^2$ sind die annotierten und vorhergesagten Positionen der Dartpfeile mit dem Index d in dem Bild mit dem Index i . Vorhergesagte Positionen für nicht vorhandene Dartpfeile haben keinen Einfluss auf diese Metrik. Dieser Aspekt wird analog zu μ_{cls} durch die Auswertung von μ_{xst} abgebildet.

Datenquelle	Gerenderte Bilder	Reale Bilder Validierung	Test	DeepDarts- d_1 Validierung	Test	DeepDarts- d_2 Validierung	Test
Anzahl Bilder	2048	125	55	1000	2000	70	150

Tabelle 4.2: Datenquellen für die Auswertung der Dartscheibenentzerrungen.

Abbildung 4.6: Auswertung von μ_{xst} der Systeme auf unterschiedlichen Datenquellen.

4.4.2 Datenquellen und Herangehensweise

Für die Auswertung der Performance des neuronalen Netzes werden Daten unterschiedlicher Quellen verwendet, aufgelistet in Tabelle 4.2. Analog zur Auswertung der algorithmischen Normalisierung der Bilder in Abschnitt 3.4 werden die selben gerenderten Bilder sowie die Validierungs- und Testdaten von DeepDarts einbezogen. Zusätzlich wurden Bilder von Darts-Runden manuell aufgenommen und händisch annotiert, um weitere unabhängige Daten einzubinden. Diese sind aufgeteilt in Daten, die zur Validierung während des Trainings verwendet wurden, und Daten, die ausschließlich zum Testen verwendet werden.

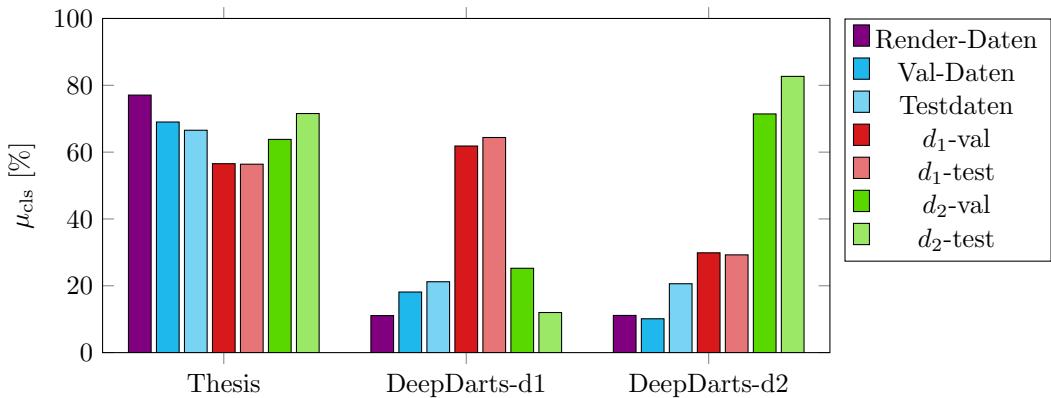
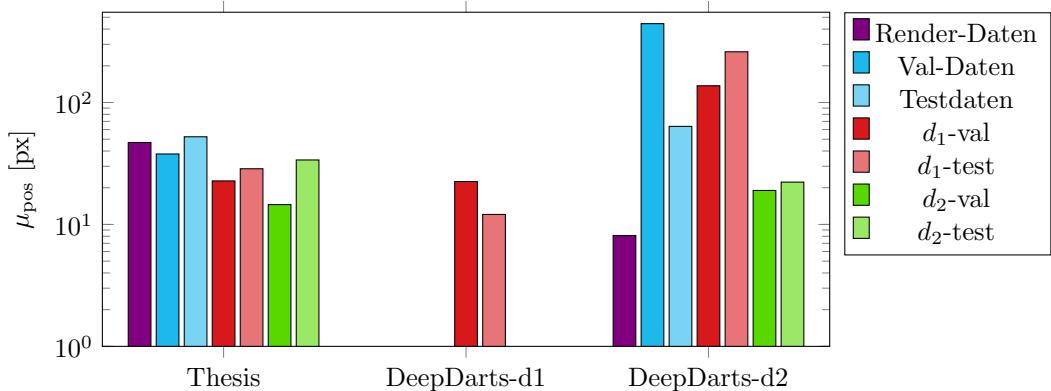
Die in den folgenden Unterabschnitten dargestellten Auswertungen stellen die Ergebnisse des gesamten Systems unter Einbezug der Normalisierung dar. Hintergrund dieses Zusammenschlusses der Verarbeitungsschritte ist die Vergleichbarkeit mit DeepDarts, in welchem die Verarbeitungsschritte miteinander verschmolzen und ebenfalls als Gesamtsystem evaluiert sind. Es werden das für diese Arbeit trainierte System sowie die für DeepDarts trainierten Systeme ausgewertet, um einen objektiven Vergleich der Performances darzustellen und einen Vergleich der Systeme zu ermöglichen.

4.4.3 Auswertung der Existenz-Metrik μ_{xst}

Die Auswertungen der Existenz-Metrik μ_{xst} sind in Abbildung 4.6 dargestellt; die Abbildung zeigt die Auswertungen der Metrik der Systeme auf den jeweiligen Datenquellen. Das neuronale Netz dieser Thesis konnte Werte zwischen 65 % und 88 % erzielt werden. Die Verteilung der Auswertungsdifferenzen ist nahezu gleichverteilt über die unterschiedlichen Datenquellen. Die größte Genauigkeit wird auf den gerenderten Daten erzielt, während die geringsten Metrik-Werte auf den Validierungsdaten von DeepDarts- d_1 erzielt werden.

Die Auswertung von DeepDarts- d_1 zeigt eine signifikante Korrelation zwischen Datenquelle und Metrik-Auswertung. Die Auswertung auf den dem System zugeordneten Daten fällt mit durchschnittlich 66 % weitaus besser aus als auf unabhängigen Quellen, die nicht Teil des Trainings oder der Auswertung des Systems waren. Auf diesen Daten wird eine durchschnittliche Auswertung von 17 % erzielt.

DeepDarts- d_2 zeigt eine weitaus bessere Auswertung als DeepDarts- d_1 , sodass auf DeepDarts- d_1 -Daten durchschnittlich 56 % der Existenz identifiziert werden und bei DeepDarts- d_2 -Daten durchschnittlich 76 %. Die Ergebnisse auf den für diese Auswertung aufgenommenen reale Daten liegen bei 20 % und bei den gerenderten Daten sind es lediglich 11 %.

Abbildung 4.7: Auswertung von μ_{cls} der Systeme auf unterschiedlichen Datenquellen.Abbildung 4.8: Auswertung von μ_{pos} der Systeme auf unterschiedlichen Datenquellen. Je geringer die Werte, desto besser die Auswertung.

4.4.4 Auswertung der Klassen-Metrik μ_{cls}

Die Auswertung der unterschiedlichen Systeme hinsichtlich der Metrik μ_{cls} ist in Abbildung 4.7 dargestellt. Die Resultate spiegeln die Auswertung von μ_{xst} wider indes die relativen Verteilungen einer ähnlichen Struktur folgen. Die Auswertung des Ansatzes dieser Thesis zeigt Werte im Bereich um 70 % für die Daten, die für diese Arbeit erstellt wurden mit einer messbar besseren Auswertung auf gerenderten Daten im Vergleich zu realen Daten. Auf den DeepDarts- d_1 -Datensätzen konnten mit 56 % die geringsten Resultate erzielt werden während auf den DeepDarts- d_2 -Daten im Mittel eine Auswertung von 68 % erzielt wird. Mit diesen Resultaten liegen die Genauigkeiten der Findung von Feldfarben unter den Genauigkeiten der Fähigkeit, die Dartpfeile zu identifizieren.

DeepDarts- d_1 zeigt hinsichtlich μ_{cls} ähnliche Auswertungen zu μ_{xst} : Es werden einzig gute Ergebnisse mit Werten um 63 % auf den diesem System zugeschriebenen Datensätzen erzielt. Auf Datensätzen, die nicht zum Training oder zur Auswertung des Systems verwendet werden, lag im Durchschnitt lediglich eine Klassen-Genauigkeit von 17 % vor.

Ähnliche Züge der Evaluation hinsichtlich μ_{cls} sind für DeepDarts- d_2 zu verzeichnen: Die d_2 -Datensätze werden mit einer hohen Genauigkeit von durchschnittlich 77 % erkannt während weitere Datensätze mit durchschnittlich 20 % Genauigkeit erkannt werden. Die Fähigkeit, Feldfarben korrekt zu identifizieren, liegt für DeepDarts- d_2 deutlich unter der Fähigkeit, Dartpfeile zu identifizieren.

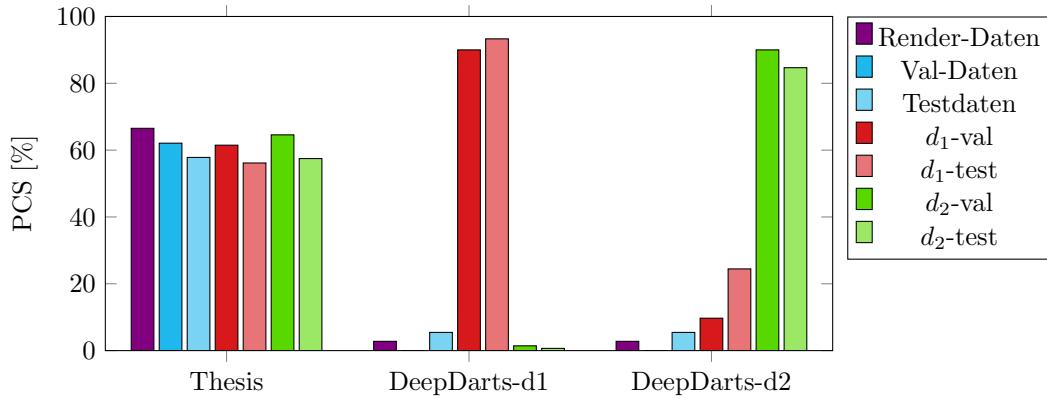


Abbildung 4.9: Auswertung von PCS der Systeme auf unterschiedlichen Datenquellen.

4.4.5 Auswertung der Positions-Metrik μ_{pos}

Mit der Metrik μ_{pos} wird die Genauigkeit der Lokalisierung der Dartpfeilspitzen untersucht. Je geringer die Wertigkeit, desto besser ist die Vorhersage. Die erzielten Ergebnisse der unterschiedlichen Systeme sind in Abbildung 4.8 dargestellt. Es ist für das in dieser Thesis trainierte neuronale Netz zu erkennen, dass die Vorhersagen realer und synthetischer Daten nicht dem Bild der Auswertungen von μ_{xst} und μ_{cls} folgen, indem die Ergebnisse der synthetischen Daten nicht signifikant von den Ergebnissen realer Daten abweichen und die Ergebnisse auf realen Daten tendenziell besser sind als auf synthetischen Daten. Die mittlere Genauigkeit aller Datensätze liegt bei einer Verschiebung von etwa 33 px.

DeepDarts- d_1 hingegen zeigt ähnliche Auswertungen in dieser Metrik wie in den zuvor ausgewerteten Metriken. Auf den Validierungs- und Testdaten von DeepDarts- d_1 konnte eine mittlere Abweichung von 17 px zu den annotierten Dartpfeilspitzen festgestellt werden.

Die Auswertungen von μ_{pos} für DeepDarts- d_2 folgen ebenso wie die Auswertung des Systems dieser Thesis nicht der Struktur der Auswertungen von μ_{xst} und μ_{cls} . Die geringsten Werte werden mit einer mittleren Verschiebung von 8 px auf den synthetischen Daten erzielt während auf den eigenen Daten im Durchschnitt 21 px Abweichung und auf allen weiteren Daten 226 px vorliegen.

4.4.6 Auswertung der PCS-Metrik

Für die Auswertung des PCS zeichnet sich ein ähnliches Bild, wie es bereits in Abbildung 3.13 ermittelt wurde, ab, indem signifikant bessere Vorhersagen von DeepDarts auf den Systemen zugeordneten Daten erzielt wird als auf den Systemen unbekannten Daten. Mit dem System dieser Thesis wird eine mittlere Korrektheit der Vorhersagen von etwa 61 % erzielt. Zu erkennen ist eine geringfügig bessere Auswertung auf synthetischen Daten, jedoch beläuft sich der Unterschied auf wenige Prozentpunkte; die Auswertungen befinden sich innerhalb einer Spanne von 9 %.

Die Spanne der durch DeepDarts- d_1 erzielten Werte des PCS ist hingegen weitaus größer. Auf den d_1 -Daten konnte eine mittlere Korrektheit von 92 % erzielt werden während auf restlichen Daten lediglich durchschnittlich 2 % der Daten korrekt vorhergesagt werden konnten. Vor den Hintergrund der CV-Auswertung in Unterabschnitt 3.4.3.2 kann abgeleitet werden, dass diese Genauigkeiten auf die Tatsache der Standard-Antwort von 0 Punkten zurückzuführen ist, die im Fehlerfall ausgegeben wird.

Hinsichtlich DeepDarts- d_2 ist ebenfalls eine starke Präferenz eigener Daten aus der Auswertung des PCS zu erkennen. Die Korrektheit der Vorhersagen auf d_2 -Daten beträgt 90 % auf Validierungs- und 85 % auf Testdaten. Diese Auswertung deckt sich mit der für DeepDarts angegebenen Korrektheit von 84 % im DeepDarts-Paper [1]. Hinsichtlich Daten, die nicht zum Training von DeepDarts- d_2 verwendet wurden, werden lediglich PCS zwischen 0 % und 5 % auf Daten dieser Thesis erzielt und 10 % und 24 % auf den d_1 -Daten.

Kapitel 5

Diskussion

In diesem Kapitel werden die Ergebnisse und Beobachtungen der Unterprojekte dieser Thesis aufgefasst und miteinander diskutiert, um mögliche Schwachstellen sowie Ungereimtheiten aufzuzeigen. Es wird mit der Diskussion der synthetischen Datenerstellung in Abschnitt 5.1 begonnen, gefolgt von der Diskussion zur Normalisierung der Dartscheiben in Abschnitt 5.2. Zuletzt folgt die Diskussion der Verwendung neuronaler Netze zur Identifizierung von Dartpfeilspitzen in normalisierten Bildern in Abschnitt 5.3.

5.1 Diskussion der Datenerstellung

Mit der Pipeline zur automatischen Datenerstellung ist die Möglichkeit gegeben, realitätsnahe Bilder von Dartscheiben zu erstellen, die zudem korrekt annotiert sind hinsichtlich der Dartpfeilpositionen im Bild sowie der Metainformationen zu dem Bild. Trotz der erzielten Erfolge sind im Verlauf der Arbeit einige Punkte aufgetreten, anhand derer die Datenerstellung erweitert und optimiert werden kann.

5.1.1 Datenumfang

Der Umfang der Daten im Bezug auf ihre Variabilität ist ein relevanter Aspekt der Datenerstellung, da dieser einen wesentlichen Aspekt zur Generalisierung der von dem neuronalen Netz erlernten Charakteristiken auf reale Bilder darstellt.

Ein wesentlicher Kritikpunkt im Bezug auf die Variabilität der Daten ist die Anzahl der möglichen Dartpfeile. Wie in Unterabschnitt 2.3.2 beschrieben ist, werden Dartpfeile aus vorgefertigten Bestandteilen zusammengesetzt. Die Anzahl der möglichen Dartpfeile ergibt sich aus der Multiplikation der Anzahlen existierender Ausprägungen der Bestandteile. Es wurden 4 Tips, 7 Barrels, 8 Shafts und 15 Flights modelliert, wodurch sich eine Gesamtzahl von 3.360 möglichen Dartpfeilen ergibt, jedoch ist der limitierende Faktor die Wiederverwendung existierender Bestandteile. Das neuronale Netz erfährt während des Trainings nicht mehr als sieben Barrels, wodurch eine Verzerrung der Datenlage nicht auszuschließen ist. Diese Beobachtung kann ein Indiz für ein unterliegendes Overfitting der Erscheinungsbilder der Dartpfeile sein, welches nicht ausgeschöpftes Potenzial der Datenerstellung überschatten kann.

Weiterhin ist die Anzahl der Dartpfeile je Bild ein Aspekt, der strengere Betrachtung vermag. Wie in Abbildung 2.11b zu sehen ist, verzeichnet die Verteilung der Dartpfeilanzahl je Bild keinerlei Bilder ohne Dartpfeile. Die Notwendigkeit von Bildern ohne Dartpfeile liegt in der Fähigkeit des neuronalen Netzes, eben diese Gegebenheit zu erlernen. Aufgrund der Architektur ist die Ausgabe keiner Dartpfeile möglich, ohne dass dieses Phänomen in den Trainingsdaten liegt, jedoch kann diese Fähigkeit durch explizites Training mit Bildern, die keine Dartpfeile enthalten, weiter vertieft werden.

Ein weiterer Aspekt des Datenumfangs ist die Verwendung von Hintergründen. Für die Datenerstellung dieser Arbeit wurde ein Pool, bestehend aus 208 Environment Maps, verwendet, aus dem für jedes Sample eine zufällige Environment Map ausgewählt und in ihrer Rotation und Helligkeit randomisiert wurde. Durch Einbindung weiterer Datenquellen ist eine Vervielfältigung der Hintergründe möglich, die für eine diversere Datenlage sorgen. Ebenso

ist die Auswahl der möglichen Beleuchtungen der Szene durch lediglich fünf unterschiedliche Lichtquellen stark limitiert. Obgleich diese Lichtquellen in ihrer Zusammensetzung kombiniert und modifiziert werden, kann die Anzahl der Beleuchtungsmöglichkeiten eine Einseitigkeit in die Daten einfließen lassen.

Zuletzt wird ein statisches Compositing verwendet, in welchem Imperfektionen der Kamera, Kontrast und Rauschen auf das Kamerabild gelegt werden, um es an das Aussehen von Aufnahmen aus Handykameras anzupassen. Die Parameter dieser Nachverarbeitung sind statisch und für alle generierten Daten gleich. Durch Einbindung zufälliger Änderungen kann die Variabilität der Nachverarbeitung erweitert werden, um eine größere Variabilität der Daten zu erzielen. In der Datenerstellung für diese Thesis wurde jedoch eine universelle Nachverarbeitung vorgenommen, durch die ebenfalls eine Einseitigkeit in das Aussehen der Daten eingeflossen sein kann.

5.1.2 Realismus der Daten

Bei der qualitativen Betrachtung der synthetischen Daten ist eine Differenzierung zwischen realen und synthetischen Daten mit geringer Unsicherheit möglich. Synthetische Aufnahmen sind ohne Probleme als diese zu identifizieren. Zwar ist gezeigt worden, dass diese Daten den Kern der Datenlage realer Aufnahmen einfangen und diese für ein Training eines neuronalen Netzes ausreichen, jedoch ist die Generalisierung der gelernten Charakteristiken und Merkmale auf Bilder realer Dartscheiben nicht reibungslos möglich. Durch starke Augmentierung der Bilder, beispielsweise Kontrastanpassung und Hinzufügen von Rauschen, werden diese Bilder derart verzerrt, dass die Diskrepanz zwischen augmentierten realen und augmentierten synthetischen Bildern gemindert wird, jedoch kann sie auch nicht durch diese Technik gänzlich geschlossen werden.

Für die Texturierung der Objekte wurden vor dem Hintergrund der Variabilität weitestgehend prozedurale Materialien verwendet. Diese bieten den Vorteil, eine beliebige Anzahl unterschiedlicher Erscheinungsbilder darzustellen. Diese Flexibilität wirkt sich jedoch auf den Realismus der Daten aus. Am anderen Ende des Spektrums liegt die Verwendung von Scans realer Objekte. Diese weisen fotorealistische Charakteristiken auf, jedoch minimale bis keine Variabilität. Die Vereinigung von Prozeduralität und Realismus ist ein komplexes Thema, welche viel Zeit und Ressourcen beansprucht. Eine Optimierung der Materialien ist jedoch ein Aspekt, durch welchen die Qualität der Daten weiter gehoben werden kann.

Wie bereits in Unterabschnitt 2.4.3 erwähnt, ist die Umgebung der Dartscheiben ein wesentlicher Aspekt, der den Grad des Realismus stark einschränkt. Bei der Existenz von Objekten wie Lichtring oder Dartschrank im unmittelbaren Hintergrund der Dartscheibe ist ein starker subjektiver Unterschied mit Hinsicht auf den Realismus der Bilder zu erkennen. Das Zusammenspiel von Lichtreflexionen des Hintergrunds und umliegenden Objekten um die Dartscheibe ist ein Aspekt, der in der Datenerstellung kaum vorhanden ist. In Aufnahmen realer Dartscheiben ist dieser Unterschied deutlich zu sehen, indem Dartscheibe und Dartpfeile durch die umliegenden Objekte indirekt beleuchtet werden.

5.1.3 Genauigkeit der Datenerstellung

In Unterabschnitt 2.3.6 wurde die Herangehensweise der Lokalisierung von Orientierungspunkten der Dartscheibe in dem gerenderten Bild erklärt. Diese beinhaltet das Identifizieren von Pixelclustern, die aus einer binären Maske des Bildes extrahiert werden. Durch die Kameraperspektive und Diskretisierung der Maskenerstellung stimmt der Mittelpunkt der jeweiligen Cluster nicht notwendigerweise mit dem gesuchten Punkt überein, sodass eine minimale Verschiebung um wenige Pixel resultieren kann. Da für die Lokalisierung der Dartscheibe vier Orientierungspunkte verwendet werden, welches die minimale Anzahl notwendiger Punkte zur Berechnung einer Homographie sind, herrscht keine Redundanz für Fehlerkorrektur. Diese Schwachstelle ist bereits bei der Normalisierung der Bilddaten durch DeepDarts kritisiert worden. In diesem Fall beläuft sich der mögliche systematische Fehler jedoch auf wenige Pixel. Die extrahierten Trainingsdaten sind daher nicht makellos.

Die selbe Technik der Mittelpunktfindung von Pixelclustern in Maskenbildern wird zur Lokalisierung der Dartpfeile verwendet. Das Herunterbrechen einer Fläche, in diesem Fall die Schnittfläche der Dartpfeilspitzen und der Dartscheibe, auf einen Punkt ist grundlegend fehleranfällig, da eine Dimensionsreduktion von zwei Dimensionen (Fläche) auf keine Dimensionen (Punkt) geschieht.

5.1.4 Effizienz der Datenerstellung

Die Geschwindigkeit der Datenerstellung wurde mit 30 Sekunden je Sample errechnet. Dazu wurden die Daten jedoch parallel auf mehreren Grafikkarten erstellt. Die Auslastung der GPUs wurde trotz Erkennung und Einbindung in die Pipeline nicht in der Art ausgeschöpft wie es unter Verwendung der Nutzeroberfläche der Fall war. Scheinbar wurden viele Berechnungen auf die CPU verlagert, die von der GPU hätten übernommen werden können. Durch dieses Bottleneck wurden wesentliche Einbußen in der Geschwindigkeit verzeichnet.

Darüber hinaus ist die Art und Weise des Einlesens und der Berechnung der Randomisierung weitestgehend sequenziell. Parallelisierung der Ausführungsschritte durch vorgezogene Erstellung von Szenenparametern und Ermittlung von Objektparametern während des Renderns von Szenen können für eine Optimierung der Datenerstellung sorgen. Ebenso ist das sequenzielle Rendern der Maskenbilder ein sehr zeitaufwändiger Prozess, der durch Parallelisierung optimiert werden kann.

Darüber hinaus sorgt ein Memory Leak in der Implementierung der Bibliothek *bpy* für zunehmende Speichernutzung bei subsequentem Rendern mehrerer Sample. Zur Umgehung dieses Problems wird die Datenerstellung für jedes Sample neu gestartet, indem das Projekt neu eingelesen wird und jegliche Einstellungen redundant ausgeführt werden müssen. Dadurch ist mit deutlichem Overhead zu rechnen im Vergleich zu sukzessivem Erstellen von Daten.

5.2 Diskussion der algorithmischen Normalisierung

Die Erkennung und die Normalisierung der Dartscheiben in Bildern dient als Vorverarbeitungsschritt und ist in dieser Arbeit als gesonderter Schritt in der Inferenz gehandhabt. Bei dieser Vorverarbeitung der Daten sind im Verlauf der Arbeit Aspekte aufgetreten, die in diesem Unterkapitel diskutiert werden.

5.2.1 Verwendete Technik

Mit DeepDarts wurde ein System vorgestellt, welches durch die Verwendung neuronaler Netze zuverlässige Ergebnisse auf Testdaten erzielen konnte. Von diesem Ansatz wurde sich aus Gründen der Flexibilität und des vorhandenen Hintergrundwissens im Bereich der herkömmlichen CV gelöst, indem der Prozess zweigeteilt wurde. Die resultierende Aufteilung in algorithmische Normalisierung und Dartpfeilerkennung auf Grundlage neuronaler Netze zieht sowohl Vorteile wie Nachteile mit sich.

Da die Schritte der Normalisierung algorithmisch durchgeführt werden und die Funktionsweise im Gegensatz zur Verwendung neuronaler Netze bekannt ist, kann Fehlerfällen gezielt nachgegangen werden. Jedoch zieht dieser Vorteil einen Schwachpunkt in der Ausführungszeit mit sich. Neuronale Netze arbeiten durch die Verwendung abstrakter Informationen stark parallel, um zu einem Ergebnis zu gelangen. Dem gegenüber stehen algorithmische Methoden, in denen eine geringe Zahl konkreter Informationen sequenziell verarbeitet wird. Durch die starke Parallelität neuronaler Netze ist die Verarbeitung der Daten effizienter als algorithmische Methoden. Die Ausführungszeit der in dieser Thesis erarbeiteten Algorithmen ist durch die sequenzielle Natur der Herangehensweise limitiert. Zudem ist die Implementierung eines Großteils des Algorithmus in einer interpretierten Programmiersprache ein Bottleneck der Performance. Trotz der Verwendung kompilierter Bibliotheken weist dieser Algorithmus weitaus längere Ausführungszeiten auf als in kompilierten Programmiersprachen zu erwarten ist.

Die Ausführungszeit von DeepDarts beträgt im Mittel zwischen 0,1 und 0,2 Sekunden auf DeepDarts-Daten mit einer festen Größe von 800×800 px und etwa 0,25 Sekunden auf den gerenderten Daten variabler Größen. Die Ausführungszeit des Algorithmus zur Normalisierung in dieser Arbeit beläuft sich mit etwa 0,3 Sekunden auf DeepDarts-Daten und über 0,4 Sekunden auf den gerenderten Testdaten auf fast die doppelte Dauer. Absolut gesehen ist diese Dauer der Ausführung akzeptabel, jedoch ist dieser Unterschied der Herangehensweise hervorzuheben, insbesondere hinsichtlich der Tatsache, dass mit der Normalisierung lediglich der erste von zwei Schritten zum Scoring vollbracht ist, wohingegen mit DeepDarts die gesamte Vorhersage nach der gemessenen Zeitspanne vollzogen ist.

5.2.2 Zuverlässigkeit des Systems

Durch die Analysen in Abschnitt 3.4 konnte ein hoher Grad der Genauigkeit und Robustheit des Systems aufgezeigt werden. Mitverantwortlich für diesen hohen Grad der Robustheit ist ein durchweg pessimistisches Einschätzen der Resultate und großzügiges Thresholding von Daten. Zur Identifizierung einer Entzerrung der Dartscheibe ist ein Minimum von vier Punkten notwendig, welche aus bis zu 60 Kandidaten erkannt werden können. Resultierend sind lediglich 5 % der potenziell erkennbaren Orientierungspunkte notwendig für eine Normalisierung. Weiterhin wird RANSAC verwendet, um einen robusten Umgang mit Outliern zu gewährleisten. Die Kombination dieser Techniken sorgt jedoch in speziellen Fällen für fehlerhafte Identifizierungen und kann durch optimierte Prozesse in der Findung der Orientierungspunkte optimiert werden. Ist eine zuverlässige Findung vieler Orientierungspunkte möglich, ist der Einsatz von RANSAC weniger fehleranfällig und es können bessere Entzerrungen gefunden werden.

Ebenfalls wird Nichtdeterminismus durch die Verwendung von RANSAC in das System eingeführt. Der Algorithmus produziert daher keine eindeutigen Resultate, sodass unterschiedliche Durchläufe auf dem selben Bild zu unterschiedlichen Ergebnissen führen können. Dieser Nichtdeterminismus sorgt für Ungewissheiten der Vorhersagen des Algorithmus.

Zuletzt ist die Art der Vorverarbeitung der Bilder für den Algorithmus ein Problempunkt, an dem potenziell relevante Informationen verworfen werden. Wie in Unterabschnitt 3.2.2 beschrieben, werden Bilder auf eine maximale Seitenlänge von 1.600 px iterativ um den Faktor 2 verkleinert. Der Hintergrund dieser Skalierung ist der Zeitaufwand der Operationen. Sowohl Genauigkeit wie Berechnungsdauer der Algorithmen skalieren mit der Größe der Bilder. Um eine zeitliche Obergrenze der Berechnungsdauer zu setzen, wurde sich für eine Vorverarbeitung zur Reduktion der Bildgröße entschieden. Der damit einhergehende Informationsverlust im Bild sorgt potenziell für Ungenauigkeiten bei der Berechnung der Normalisierung.

Trotz der erwähnten Kritikpunkte konnte ein weitestgehend zuverlässiger Normalisierungs-Algorithmus erarbeitet werden. Die Erfolgsrate der Findung von Normalisierungen in Bildern beträgt $\geq 97\%$, wie in Unterabschnitt 3.4.3.2 dargestellt ist. Zusätzlich ist die Genauigkeit der Normalisierungen mit einer maximalen mittleren Abweichung von < 35 px ermittelt worden, wie in Unterabschnitt 3.4.3.3 gezeigt. In Relation zu den Dimensionen des normalisierten Bildes mit einer Größe von 800×800 px beträgt die Abweichung $< 5\%$ der Seitenlänge.

Hinsichtlich der DeepDarts-Systeme konnten starke Präferenzen für den Systemen zugewiesene Datensätze dargestellt werden. Die Erfolge der Normalisierungen gelingen bei der Verwendung von DeepDarts- d_1 ausschließlich auf den Validierungs- und Testdaten von DeepDarts- d_1 . Die Genauigkeiten dieser Normalisierungen weisen hingegen die geringsten Abweichungen zu den annotierten Normalisierungen auf. DeepDarts- d_2 konnte Normalisierungen auf allen Datensätzen identifizieren, jedoch ist ebenfalls eine starke Präferenz der eigenen Daten zu erkennen, sowohl in der Erfolgsrate als auch in der Genauigkeit. Die mittlere Verschiebung auf gerenderten Daten mit 1.500 px liegt weitaus über den Abmessungen der Bilder und ist damit nicht als valide Normalisierung einzuordnen. Es ist aus diesen Ergebnissen abzuleiten, dass keine Generalisierbarkeit von DeepDarts auf den Systemen unbekannten Daten erzielt werden kann.

5.3 Diskussion der Lokalisierung durch Verwendung neuronaler Netze

Die Erkennung der Dartpfeile geschieht durch ein neuronales Netz, welches in einer eigenen Implementierung durch synthetische Daten trainiert wurde. Dieses neuronale Netz basiert auf einer etablierten Architektur, welche durch Integration eigener Änderungen adaptiert wurde. Durch diese Änderungen wurde eine Spezifizierung des Netzwerks auf die zugrundeliegende Aufgabe vorgenommen. Obgleich diese Herangehensweise Vorteile mit sich zieht, ist eine Adaption etablierter Architekturen ein Aspekt, der kritisch betrachtet werden muss. In den folgenden Unterkapiteln werden Aspekte des Modells und des Trainings diskutiert.

5.3.1 Eigene Implementierung des Modells

Die übliche Handhabung des Trainings einer bereits etablierten Architektur beinhaltet die Verwendung der vortrainierten Netzwerkparameter. Durch die eigene Implementierung zur Adaption der Architektur ist das Zurückgreifen auf die vortrainierten Parameter nicht möglich. YOLOv8 wurde mit Hilfe von PyTorch entwickelt und implementiert während in dieser Arbeit TensorFlow verwendet wurde. Vortrainierte Modelle beherbergen den Vorteil, deutlich mehr Bildern ausgesetzt gewesen zu sein, sodass die Netzwerkparameter in entsprechendem Einklang miteinander sind, dass eine Vielzahl an Strukturen sinnvoll verarbeitet werden kann. Dieser Startpunkt des Trainings ist im Vergleich zu einem nicht vortrainierten Netzwerk vorteilhaft, da eine generelle Strukturerkennung bereits antrainiert ist und ein erheblicher Teil des Trainings bereits vollzogen ist.

Auch hinsichtlich der Generalisierbarkeit auf neue Situationen ist die Verwendung vortrainierter Netzwerke von Vorteil, da mehr Wissen über nicht in den Trainingsdaten vorhandene Objekte zur Adaption auf die eigene Aufgabe in den Parametern des Netzwerks eingebettet sind. Wird ausschließlich auf eigenen Daten trainiert, ist die Spanne der Generalisierbarkeit durch die Variabilität der eigenen Daten vorgegeben.

5.3.2 Training des Modells

Für diese Arbeit wurde sich für ein nahezu ausschließliches OOD-Training entschieden. Der Hintergrund liegt in der Beschaffung qualitativ hochwertiger und korrekter Daten. Trotz Salting weniger realer Daten besteht der Großteil der Daten aus synthetisch erstellten Bildern. In Unterabschnitt 2.4.3 wurden sichtbare Unterschiede zwischen synthetischen und realen Daten hervorgehoben. Für ein OOD-Training ist diese Beobachtung ein Indiz dafür, dass ein systematischer Fehler in die Trainingsdaten einfließt, der Einfluss auf die Fähigkeit zur Generalisierung des Netzwerks auf reale Daten haben kann. Obwohl dieser Fehler messbar ist, wie in Abschnitt 4.4 aufgezeigt, ist die Magnitude der Diskrepanz realer und synthetischer Testdaten lediglich geringfügig.

Bei der Betrachtung der unterschiedlichen Metriken, welche die Funktionsweisen verschiedener Bereiche der Netzwerkarchitektur beleuchten, ist zu vermerken, dass die Identifizierung der Feldfarbe deutliche Differenzen zwischen synthetischen und realen Daten aufzeigt. Dieser Umstand deutet auf eine nicht ausreichende Abdeckung möglicher Feldfarben in der Datenerstellung hin, wodurch ein gewisser Grad der Verzerrung der Trainingsdaten aufgezeigt werden kann.

Trotz der Messbarkeit dieses systematischen Fehlers durch das OOD-Training ist die Fähigkeit der Übertragung erlernter Charakteristiken synthetischer Daten auf reale Daten gezeigt worden, indem wesentliche Konzepte in synthetischen Daten erlernt und auf reale Daten übertragen werden konnten. Einige Fehlerquellen der Vorhersagen auf realen Daten konnten spezifisch ausfindig gemacht werden. So wurden in vielen Fällen Logos und Markierungen als Dartpfeilspitzen identifiziert, die in dieser Art nicht in den synthetischen Daten vertreten waren. Ebenso wurden Abnutzungsspuren in den Dartfeldern gelegentlich als Dartpfeile identifiziert, deren Erscheinungsbilder nicht durch die Simulationen der Datenerstellungen abgedeckt sind.

5.3.3 Vergleich mit DeepDarts

Die für DeepDarts trainierten Modelle belaufen sich auf eine Größe von je ca. 6 Millionen Parameter; das in dieser Thesis trainierte Netzwerk umfasst etwa 10,5 Millionen Parameter. Der Aufgabenbereich des Modells von DeepDarts umfasst das Identifizieren von Dartpfeilen sowie von Orientierungspunkten in nicht normalisierten Bildern. Die Anforderung an das neuronale Netz in dieser Thesis beinhaltet die Identifizierung von Dartpfeilen und den Farben der getroffenen Felder in normalisierten Bildern. Sowohl die Parameterzahlen wie auch die zu bewältigenden Aufgaben der Netzwerke weichen signifikant voneinander ab. Während die Aufgaben für DeepDarts mehr Komplexität umfassen und zugleich weniger Parameter für die Umsetzung vorhanden sind, ist die Wahl von mehr Parametern für dieses System und seine Aufgaben nicht unbegründet. Mit DeepDarts konnte durch starkes Overfitting lediglich eine untere Schranke im Bezug auf die Parameterzahl ermittelt werden. Die wesentlich diversere Datenlage dieser Thesis erfordert mehr Parameter in einem Netzwerk, um ähnliche Ergebnisse zu erzielen.

Kapitel 6

Fazit

In diesem Kapitel wird ein Fazit zu den in dieser Masterarbeit erarbeiteten Systemen gezogen. Dazu werden die in der Einleitung aufgestellten Forschungsfragen erneut aufgegriffen und unter Einbezug der in den Ergebnissen und der Diskussion erarbeiteten Aspekte beantwortet. Anschließend werden die erzielten Ergebnisse und das Zusammenspiel der einzelnen Systeme zusammengefasst, um ein Fazit dieser Arbeit zu ziehen. Geschlossen wird mit einem Ausblick, in welchem spezifische Aspekte zur Erweiterung und Verbesserung der jeweiligen Systeme aufgezeigt werden. Durch diese kann zukünftige Arbeit an diesem Thema fortgeführt werden.

6.1 Beantwortung der Forschungsfragen

Die Forschungsfragen dieser Arbeit wurden in Abschnitt 1.6 eingeführt. Sie beziehen sich thematisch auf die Themenbereiche dieser Arbeit indes je eine Forschungsfrage für die Datenerstellung in Kapitel 2, die Normalisierung durch CV in Kapitel 3 und die Dartpfeil-Lokalisierung in Kapitel 4 aufgestellt wurden. Zusätzlich wurde eine weitere Forschungsfrage hinsichtlich des Gesamtumfangs des Projekts gestellt, in welcher der Bezug zu dem Referenzsystem DeepDarts gezogen wird. Die Beantwortung dieser Forschungsfragen geschieht in den folgenden Unterabschnitten.

1. Welche Qualität synthetischer, realistischer und variabler Daten können in einer automatisierten Pipeline zur Erstellung von Bildern von Dartscheiben mit korrekter Annotation erreicht werden?

Diese Forschungsfrage bezieht sich auf den ersten Themenbereich dieser Thesis, in welchem Daten durch Simulation automatisch generiert wurden. Die Beantwortung dieser Frage kann hinsichtlich unterschiedlicher Gesichtspunkte ausgelegt werden. Aufgrund einer fehlenden Metrik zur Quantisierung des Realismus der generierten Bilder wurde eine qualitative Analyse vollzogen. In dieser wurden die Ergebnisse der Datengenerierung kritisch betrachtet und es konnte festgestellt werden, dass die Aufnahmen keinen Fotorealismus darstellen. Bei der Betrachtung der Ergebnisse fällt die synthetische Natur der Bilder ins Auge und es ist in den meisten Fällen nicht von der Hand zu weisen, dass es sich bei den Bildern nicht um reale Aufnahmen handelt. Trotz dessen konnten realistische Gebrauchsspuren, Beleuchtungen und Aufnahmen simuliert werden, wie sie in Aufnahmen realer Dartscheiben zu finden sind. Durch Metadatenanalyse konnten realistische Spannen unterschiedlicher Parameter identifiziert und synthetisch nacherzeugt werden. Zusammenfassend lässt sich die Qualität der erstellten Daten als annähernd realistisch einordnen.

Hinsichtlich der Korrektheit der Annotationen der Daten sind geringfügige Ungenauigkeiten in der automatischen Normalisierung der Bilder zur Generierung von Trainingsdaten für das Training neuronaler Netze vorzufinden. Diese Ungenauigkeiten befinden sich im Umfang von Abweichungen lediglich weniger Pixel. Gleiche Abweichungen gelten für die Lokalisierung von Dartpfeilen in Bildern. Die Magnitude dieser Fehler ist jedoch geringfügig und qualitativ gleichauf mit manueller Annotation, jedoch ohne fehlerhafte Annotation oder übersehene Datenpunkte.

Das Scoring der Dartpfeile ist durch den Zugriff auf die Positionen der Dartpfeile im 3D-Raum zuverlässig und fehlerfrei möglich. Die zur Verfügung stehenden Informationen der Dartpfeile im Raum ermöglichen eine exakte Lokalisierung der Dartpfeile auf der Dartscheibe. Auf diese Weise ist eine eindeutige Identifizierung der getroffenen Felder möglich, durch welche die erzielte Punktzahl korrekt abgeleitet werden kann.

2. Zu welchem Grad lässt sich eine zuverlässige algorithmische Erkennung und Normalisierung von Dartscheiben in Bildern ohne den Einsatz neuronaler Netze umsetzen?

Mit der Unterteilung des Scorings in Normalisierung der Dartscheibe und Lokalisierung der Dartpfeile wird sich im Wesentlichen von der Herangehensweise von DeepDarts gelöst. Aufgrund der Nachteile in der Nutzung neuronaler Netze wurde sich für die Normalisierung für die Verwendung herkömmlicher CV entschieden. Die Forschungsfrage dieses Themenbereichs der Arbeit thematisiert den Mehrwert dieser Aufteilung, indem die Zuverlässigkeit betrachtet wird. Die Auswertung der Normalisierung wurde mittels dreier Metriken vorgenommen, die die Dauer, den Erfolg und die Genauigkeit untersuchen.

Die Dauer der algorithmischen Normalisierung ist im Vergleich zu dem DeepDarts-Ansatz messbar größer. Die Ausführungszeit der Normalisierung beläuft sich auf etwa die doppelte Dauer der vollständigen DeepDarts-Inferenz. Absolut betrachtet ist die durchschnittlich gemessene Dauer von $\leq 0,5$ s jedoch kein schwerwiegender Kritikpunkt.

Die Fähigkeit der Normalisierung wurde mit $> 97\%$ gemessen und ist damit sehr zuverlässig. Einige Kritikpunkte der Ausführung wurden in der Diskussion bereits aufgefasst, anhand derer die Ausführung der Normalisierung verbessert werden kann. Die bereits erzielte Fähigkeit zur Identifizierung stellt jedoch eine sehr gute Grundlage dar.

Hinsichtlich der Genauigkeit der erzielten Normalisierungen wurde eine Metrik verwendet, in welcher die mittlere Verschiebung der Orientierungspunkte betrachtet wurde. Die Auswertungen beliefen sich in unterschiedlichen Datensätzen auf unterschiedliche Wertebereiche. Die geringste Genauigkeit wurde auf synthetischen Daten, die ebenfalls die größte Komplexität vorweisen, erzielt. Die gemessene mittlere Genauigkeit auf den gerenderten Daten beläuft sich auf < 35 px, was $< 5\%$ der Bildbreite entspricht.

Zusammenfassend lässt sich zur Beantwortung dieser Forschungsfrage festhalten, dass die Kombination der Metriken das Bild einer sehr zuverlässigen Normalisierung ohne den Einsatz neuronaler Netze zeichnen. Die Erfolgsrate sowie die Genauigkeit des Algorithmus sind zuverlässig, einzig die Dauer der Ausführung ist als potenziell nicht zufriedenstellend festzuhalten.

3. Wie zuverlässig ist eine Generalisierung eines durch OOD-Training mit synthetischen Daten trainiertes neuronales Netzwerk auf Daten realer Dartscheiben?

Das Training des neuronalen Netzes zur Lokalisierung der Dartpfeile in normalisierten Bildern verlief auf 98,5 % synthetischen Daten mit Salting von 1,5 % realer Daten. Damit ist kein reines OOD-Training vollzogen. Die Auswertung des neuronalen Netzes wurde auf Datensätzen unterschiedlicher Quellen vollzogen, in welchen die Effekte des OOD-Trainings dargestellt werden konnten.

Die Auswertung auf unterschiedlichen Metriken zur Betrachtung der Performance unterschiedlicher Bestandteile der Netzwerkarchitektur zeigt eine zwar messbare, jedoch geringfügige Präferenz synthetischer Daten gegenüber realen Daten. Die wesentliche Fehleranfälligkeit wurde in der Identifikation der Feldfarben ermittelt, was auf eine Diskrepanz simulierter und realer Farben hinweist. Trotz dieser Differenzen ist von keiner starken Verzerrung oder kritischer Einseitigkeit der Trainingsdaten auszugehen, welche nicht in realen Aufnahmen widergespiegelt ist.

Die Zuverlässigkeit des in dieser Arbeit trainierten neuronalen Netzes ist daher im Wesentlichen nicht durch den Einsatz von OOD-Training beeinflusst. Bei der Inferenz auf synthetischen und realen Daten können annähernd gleichwertige Auswertungen erzielt werden.

4. Ist das in dieser Thesis erarbeitete Gesamtsystem in der Lage, signifikante Verbesserungen hinsichtlich der Performance und Genauigkeit im Vergleich zu DeepDarts zu erzielen?

Das Zusammenspiel der in dieser Thesis erarbeiteten Komponenten – synthetische Datengenerierung, algorithmische Normalisierung und Lokalisierung von Dartpfeilen – zielt auf eine Verbesserung des DeepDarts-Systems ab. Die Erkenntnisse von DeepDarts wurden genutzt, um die Strukturierung und den Aufbau dieser Arbeit zu bestimmen und die Ausarbeitung der Themenbereiche ist ausgerichtet auf die gezielte Steigerung der Performance unterschiedlicher Aspekte des Systems.

Die Auswertung dieses Systems zeigt eine klare Verbesserung hinsichtlich der Fähigkeit zur Generalisierbarkeit auf neue Daten auf. Während mit den Systemen von DeepDarts lediglich Erfolge auf eigenen Daten erzielt werden konnten, ist die erfolgreiche Inferenz auf dem System unbekannten Daten durch die strikte Trennung von Trainings-, Validierungs- und Testdaten gezeigt worden. Die Auswertungen dieses Systems zeigen eine signifikant bessere Performance hinsichtlich unterschiedlicher Metriken. Bei der Wahl von Daten und Metriken wurde sich zu Teilen auf die in DeepDarts verwendeten Vorgaben gestützt, um einen Vergleich ziehen zu können, der nicht durch vorteilhaft gestrickte Daten oder Metriken beeinflusst ist.

6.2 Zusammenfassung des Systems

In dieser Thesis ist eine neue Herangehensweise an ein bereits bestehendes System vorgestellt. Es wurde nicht nur eine automatisierte Erweiterung der Trainingsgrundlage eines neuronalen Netzes geschafft, sondern zusätzlich wurden neue Techniken zur Bewältigung der Aufgabe eingebunden und die Erkenntnisse aus dem bestehenden System wurden verwendet, um ein robustes System zu erschaffen.

Mit der synthetischen Datenerstellung wird das Erstellen beliebiger Datenmengen zum Trainieren eines neuronalen Netzes ermöglicht. Qualitativ betrachtet ist kein vollständiger Fotorealismus erzielt, jedoch ist die erzielte Qualität der Daten bei Weitem nicht unrealistisch. Der Realismus beläuft sich auf einen Grad, auf welchem das Trainieren eines neuronalen Netzes ohne wesentliche Verzerrungen möglich ist und eine Inferenz auf realen Daten mit geringer Diskrepanz zu synthetischen Daten ermöglicht. Durch gezieltes Angehen unterschiedlicher Kritikpunkte der Datenerstellung können die Unterschiede zwischen synthetischen und realen Daten verringert und ein erweitertes Training ermöglicht werden.

Im Vergleich zu DeepDarts konnte auf Grundlage der synthetischen Daten und der Verwendung ausgiebiger Augmentierung ein System trainiert werden, dessen Auswertungen nicht auf Overfitting der eigenen Daten schließen lässt. Dadurch ist die Generalisierbarkeit des Systems weitestgehend aufrechterhalten, sodass die Inferenz auf neuen Daten unter vorhersehbarer Genauigkeit abläuft.

Durch die algorithmische Normalisierung der Dartscheibe wurde ein System vorgestellt, welches in der Lage ist, beliebige Bilder von Dartscheiben zu normalisieren. Die algorithmische Natur dieser Herangehensweise schlägt sich in der Wartbarkeit und Interpretierbarkeit des Systems nieder. Während die Arbeitsweise von Ansätzen unter Einbindung neuronaler Netze nicht bekannt sind und Fehlerfälle durch intensive Auswertungen und Interpretierungen angegangen werden müssen, ist die Adaption eines Algorithmus gezielt möglich. Ursachen der fehlerhaften Verarbeitung können schnell ausfindig gemacht werden und es bedarf keine großen Datenmengen sowie Infrastruktur mit großer Rechenleistung, um das System auf neue Daten anzupassen.

Zusammenfassend kann von einem Erfolg bezüglich der gestellten Aufgabe gesprochen werden. Das erarbeitete System ist zu dem aktuellen Zeitpunkt nicht in geeignet für den realen Einsatz, beispielsweise in einer App. Die Fehlerquote des Systems liegt mit etwa 40 % deutlich zu hoch, jedoch ist eine solide Grundlage zur Erarbeitung eines leistungsstarken Systems gegeben.

6.3 Ausblick

Diese Arbeit stellt ein System vor, mit welchem ein automatisches Dart-Scoring durch die Verwendung herkömmlicher CV sowie ein aktuelles neuronales Netz, welches durch automatisch und synthetisch erstellte Daten trainiert wurde. Das Potenzial dieser Systeme ist nicht vollkommen ausgeschöpft und kann auf unterschiedliche Arten weitergeführt werden. Dieser Abschnitt gibt einen Überblick über mögliche Ausgestaltungen und Verbesserungen dieser Systeme. Es wird begonnen mit dem Ausblick der Datenerstellung in Unterabschnitt 6.3.1, gefolgt von dem Ausblick der Pipeline zur Normalisierung in Unterabschnitt 6.3.2 und abschließend wird in Unterabschnitt 6.3.3 ein Überblick über weitere mögliche Arbeit an der Dartpfeilerkennung gegeben.

6.3.1 Ausblick der Datenerstellung

Die Datenerstellung ist ein komplexer Prozess, welcher ein Zusammenspiel vieler unterschiedlicher Komponenten ist. Die Möglichkeiten zur Erweiterung dieses Systems sind daher sehr vielseitig. Anstatt der Optimierung von Implementierungsdetails wird sich in diesem Ausblick auf die methodischen Aspekte zur Erweiterung der Datenerstellung fokussiert.

Integration von physically-based rendering (PBR) Ein wesentlicher Bestandteil der Datenerstellung ist die prozedurale Erstellung von Materialien. Diese ist aktuell nicht einheitlich über die Objekte eingebunden und bedarf der Integration von PBR. In diesem werden unterschiedliche Texturen zur Steuerung verschiedener Parameter wie Glanz und Oberflächenbeschaffenheit strikt getrennt voneinander definiert, um realistische Texturen zu erstellen.

Anzahl der Objekte in der Szene Die Szene umfasst aktuell neben den Dartpfeilen fünf Objekte, die dynamisch ein- und ausgeblendet werden können, lediglich zwei dieser Objekte – Lichtring und Dartschrank – sind in den Aufnahmen zu sehen; die restlichen Objekte sind Lichtquellen. Durch Hinzufügen weiterer Objekte in die Szene, beispielsweise Wände und Dekoration, können mehr Bedingungen und Störfaktoren in die Daten aufgenommen werden, die in realen Aufnahmen zu erwarten sind. Die Aufnahmen können durch das gezielte Hinzufügen weiterer Objekte realistischer gestaltet werden.

Aussehen der Dartscheibe Das Aussehen der Dartscheibe ist konform der Regelwerke der WDF und der Professional Darts Corporation (PDC) gestaltet [5, 6]. Diese gibt Form und Farben der Dartscheibe mit etwaigen Toleranzen weitestgehend vor. Für die aktuelle Datenerstellung werden lediglich diese und ähnliche Dartscheiben als Vorlagen in Betracht gezogen, wodurch Möglichkeiten weiterer Dartscheiben ausgeschlossen sind. Durch Hinzufügen von Parametern zur Steuerung der Art der Dartscheibe können ebenfalls Dartscheiben mit anderen Farbgebungen und Geometrien in die Datenerstellung eingebunden werden, beispielsweise Dartscheiben mit blauen und roten Feldern statt schwarzer und weißer Grundfelder.

Ebenfalls ist eine Adaption der unmittelbaren Ausgestaltung der Dartscheibe durch Zahlenring und Texte denkbar. In der hier vorgestellten Dartscheibengenerierung werden Beschriftungen und Logos um die Dartfelder herum durch Textbausteine abstrahiert. Bei der Identifizierung von Dartpfeilen auf realen Dartscheiben hat sich jedoch gezeigt, dass diese Abstraktion von Logos durch Texte nicht ausreichend ist, um diese nicht als Dartpfeile zu klassifizieren. Eine weitergehende Ausgestaltung dieser Bereiche der Dartscheibe ist daher notwendig.

6.3.2 Ausblick der Normalisierung

Der in dieser Thesis vorgestellte Algorithmus zur Erkennung und Normalisierung von Dartscheiben in beliebigen Bildern basiert ausschließlich auf Techniken herkömmlicher CV. Dadurch ist die Adaption des Algorithmus weitreichend möglich, sodass spezifische Aspekte gezielt angegangen und verbessert werden können. Im Verlauf der Entwicklung sowie der Auswertung sind verbesserungswürdige Bereiche des Algorithmus identifiziert worden, die in diesem Abschnitt betrachtet werden. Sie bieten eine Grundlage zur gezielten Erweiterung und Verbesserung der bestehenden Methodiken.

Verbesserung der Erkennung und Klassifizierung von Orientierungspunkten

Die Erkennung von Orientierungspunkten geschieht in einem zweischrittigen Verfahren. Im ersten Schritt werden mögliche Kandidaten von Orientierungspunkten lokalisiert, welche in einem zweiten Schritt klassifiziert werden. Diese Klassifizierungen basieren auf der Einordnung von Farbwerten, die durch die Analyse aller identifizierter Kandidaten abgeleitet werden. Die Klassifizierung ist dadurch tendenziell fehleranfällig, sodass eine pessimistische Klassifizierung zur Minimierung von Fehlklassifizierungen stattfindet. Unter den Kandidaten der Orientierungspunkte befindet sich voraussichtlich eine ausreichende Anzahl korrekter Orientierungspunkte, welche großzügiges Aussortieren von Kandidaten ermöglichen.

Trotz erfolgreicher Normalisierungen geschieht an diesem Punkt ein Verlust relevanter Daten, die potenziell für robustere Ergebnisse des Algorithmus sorgen können. Durch eine Überarbeitung der Methodik zur Klassifizierung von Orientierungspunkten kann dieser Bereich des Algorithmus weiter ausgebaut werden. Zur Bewerkstelligung dieser Optimierung ist der Einsatz von CNNs denkbar. Die CNNs könnten die Aufgabe der Einordnung von Surroundings übernehmen, indem sie auf zuvor korrekt erkannten Surroundings trainiert werden. Die Aufgabe dieser CNNs befindet sich in einem Bereich der Komplexität, in welchem ein Netzwerk mit einer geringen Parameterzahl eingesetzt werden kann.

Darüber hinaus besteht die Möglichkeit, die Funktionsweise Erkennung der Dartscheibe mit einer Ellipsenerkennung zu erweitern, mit der die Geometrie der Dartscheibe abgeleitet werden kann [84]. Die Verwendung von Ellipsenerkennung wurde bei der Implementierung dieser Arbeit zu Teilen implementiert, jedoch konnten keine zufriedenstellenden Ergebnisse erzielt werden. Trotz dessen liegt in der Verwendung dieser Technik viel Potenzial, welches in die Normalisierung von Dartscheiben einfließen kann.

Kompilierung des Algorithmus Zur Implementierung des Algorithmus, wie er in dieser Thesis vorgestellt und ausgewertet ist, wurde Python als Programmiersprache verwendet. Trotz der Verwendung der Bibliotheken NumPy und OpenCV, welche zu Teilen kompilierte Funktionen einsetzen, geschieht ein großer Anteil der Berechnungen durch interpretierten Quelltext. Interpretation von Quelltext ist weitaus ineffizienter als die Ausführung kompilierten Quelltextes.

Die Kompilierung des Algorithmus kann auf unterschiedliche Weisen umgesetzt werden. Bibliotheken wie Cython und Numba ermöglichen die Ausführung von Python-basiertem Quelltext bzw. Kompilierung von Teilen des Python-Quelltextes zur Optimierung von Ausführungszeiten [85, 86]. Durch diese Bibliotheken kann potenziell eine Optimierung der Laufzeit mit geringer Abänderung des vorhandenen Quelltextes erzielt werden. Ebenso ist eine Implementierung in einer kompilierten Programmiersprache wie C++ oder Rust denkbar, durch welche erhebliche Verbesserungen in der Ausführungszeit zu erwarten sind.

Auswertungen auf unterschiedlichen Plattformen Die Auswertung hinsichtlich der Ausführungszeit dieses Systems beläuft sich auf ein bestimmtes System, auf welchem die Ausführungszeiten ermittelt wurden. Das verwendete System ist jedoch weitaus leistungsstärker als die für die vorgesehene Verwendung des Systems ausgelegten Systeme, namentlich Mobiltelefone. Eine Auswertung auf verschiedenen mobilen Geräten ermöglicht als fortführende Arbeit in der Hinsicht die Optimierung der Parameter des Algorithmus, um eine optimale Abstimmung von Geschwindigkeit und Genauigkeit zu ermitteln. Die in Abschnitt 2.4 erzielten Resultate hinsichtlich der Ausführungszeiten von DeepDarts und dem in dieser Thesis erarbeiteten Algorithmus sind in diesem Aspekt lediglich relativ zueinander zu betrachten; eine Ableitung realer Ausführungszeiten ist nur bedingt möglich.

6.3.3 Ausblick der Dartpfeilerkennung

Die Lokalisierung der Dartpfeile beruht auf der Verwendung eines neuronalen Netzes, welches auf synthetischen Daten trainiert wurde. Hinsichtlich dieses Trainings und der Rahmenbedingungen werden in diesem Unterabschnitt mögliche zukünftige Aufgaben erläutert und Verbesserungsvorschläge hinsichtlich bestimmter Schwachstellen aufgelistet. Durch weitere Ausarbeitung dieser Themenbereiche wird eine Verbesserung des Systems als wahrscheinlich erachtet.

Netzwerktraining Das Training des neuronalen Netzes basiert auf einem OOD-Ansatz, in welchem synthetische Daten generiert wurden, um ein System zu trainieren, welches auf reale Daten angewandt wird. Zwischen synthetischen und realen Daten liegen jedoch nicht von der Hand zu weisende Diskrepanzen, die einem optimalen Training im Wege stehen. Durch die Aufnahme weiterer realer und zugleich variabler Daten kann die Datenlage realer Bilder für das Training vervielfacht werden, wodurch eine bessere Inferenz auf realen Daten zu erwarten ist.

Zusätzlich sind einige Rahmenbedingungen des Trainings verbesserungswürdig. Das Training verlief mit einer Batch Size von 32, welche für ein stabileres Training erhöht werden kann; aus technischen Gründen war dies im Umfang dieser Arbeit nicht möglich. Darüber hinaus wurde die Learning Rate manuell angepasst und unterlag keinem Algorithmus oder festgelegtem Muster. Für weiteres Training ist in Ausschau zu setzen, dass ein Schedule zur Anpassung der Learning Rate verwendet wird, der dem Verhalten der manuellen Learning Rate folgt.

Netzwerkarchitektur Ein wesentlicher Kritikpunkt des Netzwerktrainings dieser Thesis ist die verwendete Netzwerkarchitektur. Durch die eigene Implementierung und die vorgenommenen Adaptionen der YOLOv8-Architektur ist die Verwendung vortrainierter Gewichte nicht möglich. Es kann daher kein Transfer Learning durchgeführt werden und nicht auf bereits initialisierte und auf reale Bedingungen angepasste Gewichte zurückgegriffen werden. Durch diese Tatsache ist ein signifikanter Verlust der Performance zu erwarten. Durch ein vorheriges Trainieren des Backbones durch Unsupervised Learning auf realen Bildern können die Gewichte des Backbones an reale Gegebenheiten angepasst werden.

Zusätzlich können weitere Recherchen zur Verwendung unterschiedlicher Netzwerkarchitekturen vorgenommen werden, durch die Einblicke in die unterschiedlichen Performances erlangt werden können. Auf diese Weise kann ermittelt werden, welche Techniken in Netzwerkarchitekturen zielführend für diese Aufgabe sind und es kann ein Netzwerk konzipiert werden, welches optimal für die Erkennung von Dartpfeilspitzen in Bildern geeignet ist. Weiterhin kann zu der Untersuchung weiterer Netzwerkarchitekturen der Einfluss eigener Netzwerkbestandteile, wie dem Transition-Block, der in dieser Arbeit eingeführt wurde, genauer betrachtet und ausgewertet werden.

Quantisierung des Netzwerks Das Quantisieren von Netzwerken ist eine Technik, bei der die Netzwerkgewichte von 32-bit Float-Werten zu Integern verschiedener Längen überführt werden. Dadurch kann eine effizientere und schnellere Inferenz des Netzwerks erzielt werden während lediglich geringfügige Einbußen in der Qualität der Ausgaben in Kauf genommen werden. Die Quantisierung von Netzwerken dient der Ausführung neuronaler Netze in Umgebungen, deren Ressourcen begrenzt sind. Insbesondere mit Blick auf ein potenzielles mobiles Deployment dieses Systems ist die Quantisierung des verwendeten neuronalen Netzes ein Thema, in das die Investition weiterer Ressourcen für sinnvoll erachtet wird.

Erweiterte Herangehensweise für Vorhersagen Zuletzt kann die Art der Vorhersage des Netzwerks überarbeitet werden, um eine Anpassung an die spezifischen Gegebenheiten der verwendeten Dartscheibe vorzunehmen. Durch die Aufnahme eines Kalibrierungsbildes, in dem die Dartscheibe ohne Pfeile zu sehen ist, kann das Aussehen der Dartscheibe gespeichert werden. Bei der Vorhersage von Aufnahmen kann diese Referenzaufnahme der leeren Dartscheibe verwendet werden, um fehlerhafte Vorhersagen auf Grundlage des Erscheinungsbildes der Dartscheibe zu unterbinden. Beispielsweise sind besondere Abnutzungen oder Logos bzw. Schriftzüge um die Dartscheibe herum als häufige Fehlerquellen auf realen Daten identifiziert worden. Sofern das Aussehen der Dartscheibe ohne Pfeile bekannt ist, besteht die Möglichkeit, dieser Art fehlerhafter Vorhersagen gezielt entgegenzuwirken.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Weiterhin versichere ich, dass die digitale Fassung dieser Arbeit, die dem Prüfungsamt per E-Mail zugegangen ist, der vorliegenden schriftlichen Fassung entspricht.

Kiel, 02.06.2025

Symbolverzeichnis

$(\rho, \theta) \in \mathbb{R}^2$	Koordinaten (Abstand und Winkel) in einem polaren Koordinatensystem
$(c_x, c_y) \in \mathbb{R}^2$	Position eines Mittelpunktes in einem Bild
$(w, h) \in \mathbb{N}^2$	Breite und Höhe eines Bildes
$\Lambda : ((\mathbb{R}^{3 \times 3})^2 \mapsto \mathbb{R})$	CV-Metrik zur Identifizierung der Ähnlichkeiten von Homographien
(d_{\min}, d_{\max})	Minimaler und maximaler Abstand der Kamera von der Dartscheibe
(y_{\min}, y_{\max})	Minimale und maximale Höhe der Kamera im Raum
μ_{cls}	Klassen-Metrik zur Bestimmung korrekter Klassen der Dartpfeile je Bild.
μ_{pos}	Positions-Metrik zur Bestimmung der Abweichungen der Dartpfeilpositionen.
μ_{xst}	Existenz-Metrik zur Bestimmung korrekter Anzahl der Dartpfeile je Bild.
ϕ_h	Horizontaler Öffnungswinkel des Kamera-Space zu den Seiten der Dartscheibe
$\phi_i \in \mathbb{R}$	Winkel der Feldlinie i
$\phi_o \in \mathbb{R}$	Feldlinienwinkel der Orthogonalen Linie L_o
ϕ_v	Vertikaler Öffnungswinkel des Kamera-Space in die Höhe
σ_{noise}	Standardabweichung der Rausch-Augmentierung.
σ_{trans}	Standardabweichung der Translations-Augmentierung.
$\text{Id} \in \mathbb{R}^{3 \times 3}$	Identitäts-Transformation
lr_0	Initiale Learning Rate
sim	Similarity-Metrik zur Bestimmung des Ähnlichkeitsgrads zweier Surroundings.
$\theta_{i,p} \in \mathbb{R}$	Winkel eines Pixels p auf der Linie i zum Mittelpunkt der Dartscheibe
φ_i	Winkel eines Pixels zum Mittelpunkt der Dartscheibe.
$\hat{H} \in \mathbb{R}^{3 \times 3}$	Ermittelte Entzerrungstransformation
$\hat{N}_{\text{Dart}, i} \in \mathbb{N}$	Anzahl vorhergesagter Dartpfeile in dem Bild mit Index i .
$\hat{P}_{i,d} \in \mathbb{R}^2$	Vorhergesagte Position des Dartpfeils mit Index d in Bild i .
$\tilde{L}_{\text{points}} \in (\mathbb{R}^{2 \times 2})^n$	Gefilterte Liste kartesischer Start- und Endpunkte von Linien, die auf den Mittelpunkt der Dartscheibe gerichtet sind
$\tilde{L}_{\text{polar}} \in (\mathbb{R}^2)^n$	Gefilterte Liste polarer Linien, die auf den Mittelpunkt der Dartscheibe gerichtet sind
$\tilde{P} \in \mathbb{R}^3$	Homogene Position $(\tilde{x}, \tilde{y}, \tilde{z})$ in kartesischem Koordinatensystem
$\tilde{P}' \in \mathbb{R}^3$	Transformierte homogene Position in kartesischem Koordinatensystem
$A^{360} \in \mathbb{R}^n$	Akkumulator-Array für Winkel
A_{cont}	Augmentierungsgewicht der Kontraständerung.
A_{rot}	Augmentierungsgewicht der Rotation.
A_{sat}	Augmentierungsgewicht der Sättigungsänderung.
$A_{\text{cha}, \{r,g,b\}}$	Augmentierungsgewichte für rot-, grün- unb blau-Kanäle.
b_{adj}	Maximale Helligkeitsänderung der Augmentierung.
b_{img}	Mittlere Helligkeit eines Bildes.

$c_0 \in \mathbb{N}$	Anzahl der Eingabekanäle in einen Kernel
$c_1 \in \mathbb{N}$	Anzahl der Ausgabekanäle eines Kernels
$C_p \in \mathbb{R}^3$	CrYV-Farbe eines Patches.
$C_r \in \mathbb{R}^3$	CrYV-Referenzfarbe.
D	Generischer Definitionsbereich
d_{Kamera}	Abstand der Kamera zur Dartscheibe
d_r, d_g	Mittlere Differenz roter bzw. grüner Bereiche des Lab-Abgleichs zweier Surroundings.
$d_{i,p} \in \mathbb{R}$	Abstand einer Pixels p auf einer Linie i zum Mittelpunkt der Dartscheibe
dx_{\max}	Maximaler seitlicher Abstand der Kamera zum Dartscheibenmittelpunkt
f_{lr}	Adaptionsfaktor der Learning Rate
$H \in \mathbb{R}^{3 \times 3}$	Homographie
$h_{i,j \in [0,2]} \in \mathbb{R}$	Parameter einer Homographie
$I \in \mathbb{N}^2$	Bild mit einem Kanal
$k \in \mathbb{N}^2$	Faltungs-Kernel
$k_x, k_y \in \mathbb{N}$	x - und y -Dimension eines Kernels
l_{Kamera}	Brennweite der Kamera
$l_{\text{lower}}, l_{\text{upper}}$	Unterer und oberer Grenzwert für die Brennweite der Kamera
l_{\min}, l_{\max}	Minimale und maximale Brennweite der Kamera
$L_{\text{points}} \in (\mathbb{R}^{2 \times 2})^n$	Kartesische Start- und Endpunkte von Liniensegmenten in einem Bild
$L_{\text{polar}} \in (\mathbb{R}^2)^n$	Polare Darstellungen von Liniensegmenten in einem Bild
$L_o \in \mathbb{R}^2$	Orthogonale Linie zur Startlinie L_s
$L_s \in \mathbb{R}^2$	Startlinie der Winkelentzerrung
$M_{\text{align}} \in \mathbb{R}^{3 \times 3}$	Transformation zur Winkelentzerrung einer Dartscheibe
$m_{\text{Dart}} \in \mathbb{R}^2$	Mittelpunkt einer Dartscheibe in einem Bild
$M_{\text{final}} \in \mathbb{R}^{3 \times 3}$	Finale Entzerrungstransformation der Dartscheibe
$M_{\text{project}} \in \mathbb{R}^{3 \times 3}$	Projektionsmatrix zur Überlagerung der Orientierungspunkte einer Dartscheibe
$M_{\text{rot}} \in \mathbb{R}^{3 \times 3}$	Rotationsmatrix
$M_{\text{scale}} \in \mathbb{R}^{3 \times 3}$	Skalierungstransformation der Eingabebildes der Normalisierung
$M_{\text{scl}} \in \mathbb{R}^{3 \times 3}$	Skalierungsmatrix
$M_{\text{shr}} \in \mathbb{R}^{3 \times 3}$	Scherungsmatrix
$M_{\text{trans}} \in \mathbb{R}^{3 \times 3}$	Translationsmatrix
$M_{i \in \mathbb{N}} \in \mathbb{R}^{3 \times 3}$	Transformationsmatrix
$N_{\text{Dart, } i} \in \mathbb{N}$	Anzahl vorhandener Dartpfeile in dem Bild mit Index i .
$n_{\text{lines}} \in \mathbb{N}$	Anzahl gefundener Liniensegmente in einem Bild
$N_{\text{OP, max}}$	Maximale Anzahl der Orientierungspunkte in einem Bild
N_{OP}	Anzahl Identifizierter Orientierungspunkte in einem Bild
$N_{K, \text{correct}, i} \in \mathbb{N}$	Anzahl korrekt vorhergesagter Klassen in dem Bild mit Index i .
$O \in \mathbb{N}^2$	Koordinatenursprung
$P \in \mathbb{R}^2$	Position in kartesischem Koordinatensystem
$p \in I$	Pixel in einem Bild
$p_{\text{Dartscheibe}}$	Position der Dartscheibe im Raum
$P_{\max} \in \mathbb{R}^{\mathbb{K}}$	Kartesischer Punkt mit maximalen Linieneberschneidungen in einem Bild
$p_{\text{Pfeil}, i}$	Position der Dartpfeile im Raum
$P_{i \in [N_{\text{OP}}]} \in \mathbb{R}^2$	Positionen der identifizierten Orientierungspunkte
$P_{i,d} \in \mathbb{R}^2$	Position des Dartpfeils mit Index d in Bild i .
$s \in \mathbb{N}$	Indizes für Startlinien der Winkelentzerrung
$s_{\max} \in \mathbb{N}$	Maximale Breite bzw. Höhe eines Eingabebildes
S_i	Surrounding mit Index i .

T	Generischer Threshold
$t_{\text{green}} \in \mathbb{R}$	Cr-Kanal-Referenzwert für grüne Farbe.
$t_{\text{red}} \in \mathbb{R}$	Cr-Kanal-Referenzwert für rote Farbe.
$T_C \in \mathbb{R}$	Threshold zur Klassifizierung von CrYV-Farbdifferenzen.
$x, y \in \mathbb{N}$	Variable Koordinaten in einem kartesischen Koordinatensystem

Abkürzungen

CNN Convolutional Neural Network

CV Computer Vision

FCNN Fully Convolutional Neural Network

NMS Non-Maximum-Suppression

OOD out-of-distribution

PBR physically-based rendering

PCS Percent Correct Score

PDC Professional Darts Corporation

SSIM Structural Similarity

WDF World Dart Federation

Abkürzungen sehen nicht
gut aus, aber das zu fixen
ist nervig.

Literatur

- [1] William McNally et al. *DeepDarts: Modeling Keypoints as Objects for Automatic Scorekeeping in Darts using a Single Camera*. 2021. arXiv: 2105.09880 [cs.CV]. URL: <https://arxiv.org/abs/2105.09880>.
- [2] Auto Darts. *Auto Darts - Official Website*. Accessed: 2024-10-23. 2024. URL: <https://autodarts.io>.
- [3] SCOLIA Technologies. *Scolia - Official Website*. Accessed: 2024-10-23. 2024. URL: <https://scoliadarts.com>.
- [4] William McNally. *DeepDarts Dataset*. 2021. DOI: 10.21227/05e7-xs69. URL: <https://dx.doi.org/10.21227/05e7-xs69>.
- [5] World Dart Federation. *Playing and Tournament Rules (Twentieth revised edition)*. World Dart Federation Website. Zuletzt aufgerufen: 01.02.2025. 2018. URL: https://dartsfdf.com/storage/uploads/fb05b306-c92c-4f08-b512-affb092a1b3d/2018-02-28_WDF_Playing_and_Tournament_Rules_rev20.pdf.
- [6] Professional Darts Corporation. *Darts Regulation Authority - Rule Book*. Website. 2023. URL: https://www.thedra.co.uk/_files/ugd/20bb2f-fbc16a1efdf34cf9b8d377a5657a4daa.pdf.
- [7] Michael Beukman, Christopher W Cleghorn und Steven James. „Procedural content generation using neuroevolution and novelty search for diverse video game levels“. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '22. Boston, Massachusetts: Association for Computing Machinery, 2022, S. 1028–1037. ISBN: 9781450392372. DOI: 10.1145/3512290.3528701. URL: <https://doi.org/10.1145/3512290.3528701>.
- [8] Diogo Miguel P.L. DIAS et al. „A Novel Procedural Content Generation Algorithm for Tower Defense Games“. In: *Proceedings of the 6th International Conference on Algorithms, Computing and Systems*. ICACS '22. Larissa, Greece: Association for Computing Machinery, 2023. ISBN: 9781450397407. DOI: 10.1145/3564982.3564993. URL: <https://doi.org/10.1145/3564982.3564993>.
- [9] Gillian Smith et al. „PCG-based game design: enabling new play experiences through procedural content generation“. In: *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*. PCGames '11. Bordeaux, France: Association for Computing Machinery, 2011. ISBN: 9781450308724. DOI: 10.1145/2000919.2000926. URL: <https://doi.org/10.1145/2000919.2000926>.
- [10] Stefan Hinterstoisser et al. *On Pre-Trained Image Features and Synthetic Images for Deep Learning*. 2017. arXiv: 1710.10710 [cs.CV]. URL: <https://arxiv.org/abs/1710.10710>.
- [11] Apostolia Tsirikoglou et al. *Procedural Modeling and Physically Based Rendering for Synthetic Data Generation in Automotive Applications*. 2017. arXiv: 1710.06270 [cs.CV]. URL: <https://arxiv.org/abs/1710.06270>.
- [12] Hassan Abu Alhaija et al. „Augmented Reality Meets Deep Learning for Car Instance Segmentation in Urban Scenes“. In: *Proceedings of the British Machine Vision Conference 2017*. Sep. 2017.
- [13] Gul Varol et al. „Learning from Synthetic Humans“. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Juli 2017, S. 4627–4635. DOI: 10.1109/cvpr.2017.492. URL: <http://dx.doi.org/10.1109/CVPR.2017.492>.

- [14] Ole Schmedemann et al. „Procedural synthetic training data generation for AI-based defect detection in industrial surface inspection“. In: *Procedia CIRP* 107 (2022). Leading manufacturing systems transformation - Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022, S. 1101–1106. ISSN: 2212-8271. DOI: <https://doi.org/10.1016/j.procir.2022.05.115>. URL: <https://www.sciencedirect.com/science/article/pii/S2212827122003997>.
- [15] Yair Movshovitz-Attias, Takeo Kanade und Yaser Sheikh. *How useful is photo-realistic rendering for visual learning?* 2016. arXiv: 1603.08152 [cs.CV]. URL: <https://arxiv.org/abs/1603.08152>.
- [16] Enes Özeren und Arka Bhowmick. *Evaluating the Impact of Synthetic Data on Object Detection Tasks in Autonomous Driving.* 2025. arXiv: 2503.09803 [cs.CV]. URL: <https://arxiv.org/abs/2503.09803>.
- [17] Josh Tobin et al. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World.* 2017. arXiv: 1703.06907 [cs.R0]. URL: <https://arxiv.org/abs/1703.06907>.
- [18] Ervin Domazet. „Real-time optical dart detection and scoring algorithm for steel tip dartboards“. In: *ICGA Journal* 44.3 (2022), S. 72–90. DOI: 10.3233/ICG-230214. eprint: <https://journals.sagepub.com/doi/pdf/10.3233/ICG-230214>. URL: <https://journals.sagepub.com/doi/abs/10.3233/ICG-230214>.
- [19] Michael Ehrenberg Ankush Patel. *The Design and Implementation of an Automated Dartboard.* Final Project Report for MIT 6.111, Fall 2005. Submitted December 14, 2005. 2005. URL: https://web.mit.edu/6.111/www/f2005/projects/mje_Project_Final_Report.pdf.
- [20] Hannes Hoettinger. *opencv-steel-darts.* GitHub Project. 2019. URL: <https://github.com/hanneshoettinger/opencv-steel-darts>.
- [21] DartCaller. *darts-recognition.* GitHub Project. 2021. URL: <https://github.com/DartCaller/darts-recognition>.
- [22] Fabian Krauss Lars Gudjons. *Darts_Project.* GitHub Project. 2024. URL: https://github.com/LarsG21/Darts_Project.
- [23] Hannes Hoettinger Pär Sundbäck. *DartScore.* GitHub Project. 2020. URL: <https://github.com/teddycool/DartScore>.
- [24] Turner Whitted. „An improved illumination model for shaded display“. In: *Commun. ACM* 23.6 (Juni 1980), S. 343–349. ISSN: 0001-0782. DOI: 10.1145/358876.358882. URL: <https://doi.org/10.1145/358876.358882>.
- [25] Robert L. Cook, Thomas Porter und Loren Carpenter. „Distributed ray tracing“. In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '84. New York, NY, USA: Association for Computing Machinery, 1984, S. 137–145. ISBN: 0897911385. DOI: 10.1145/800031.808590. URL: <https://doi.org/10.1145/800031.808590>.
- [26] James T. Kajiya. „The rendering equation“. In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986, S. 143–150. ISBN: 0897911962. DOI: 10.1145/15922.15902. URL: <https://doi.org/10.1145/15922.15902>.
- [27] Sameer Agarwal et al. „An Atlas for the Pinhole Camera“. In: *Foundations of Computational Mathematics* 24.1 (Sep. 2022), S. 227–277. ISSN: 1615-3383. DOI: 10.1007/s10208-022-09592-6. URL: <http://dx.doi.org/10.1007/s10208-022-09592-6>.
- [28] Emily A. Cooper, Elise A. Piazza und Martin S. Banks. „The perceptual basis of common photographic practice“. In: *Journal of Vision* 12.5 (Mai 2012), S. 8–8. ISSN: 1534-7362. DOI: 10.1167/12.5.8. URL: <https://doi.org/10.1167/12.5.8>.

- [29] Eero Kurimo et al. „The Effect of Motion Blur and Signal Noise on Image Quality in Low Light Imaging“. In: *Image Analysis*. Hrsg. von Arnt-Børre Salberg, Jon Yngve Hardeberg und Robert Jenssen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 81–90. ISBN: 978-3-642-02230-2.
- [30] Jonathan B. Phillips und Henrik Eliasson. *Camera Image Quality Benchmarking*. Newark, UNITED KINGDOM: John Wiley & Sons, Incorporated, 2018. ISBN: 9781119054528. URL: <http://ebookcentral.proquest.com/lib/christianalbrechts/detail.action?docID=5150092>.
- [31] Mark Brady und Gordon E Legge. „Camera calibration for natural image studies and vision research“. In: *Journal of the Optical Society of America A* 26.1 (2008), S. 30–42.
- [32] Junyu Dong et al. „Survey of Procedural Methods for Two-Dimensional Texture Generation“. In: *Sensors* 20.4 (2020). ISSN: 1424-8220. DOI: 10.3390/s20041135. URL: <https://www.mdpi.com/1424-8220/20/4/1135>.
- [33] Si Si Hida Takeyuki. *Lectures on White Noise Functionals*. World Scientific Publishing Company, 2008. ISBN: 978-981-281-204-9. URL: <https://ebookcentral.proquest.com/lib/christianalbrechts/detail.action?docID=1193660>.
- [34] Ken Perlin. „An image synthesizer“. In: *Seminal Graphics: Pioneering Efforts That Shaped the Field, Volume 1*. New York, NY, USA: Association for Computing Machinery, 1998, S. 147–156. ISBN: 158113052X. URL: <https://doi.org/10.1145/280811.280986>.
- [35] Ken Perlin. „Improving noise“. In: *ACM Trans. Graph.* 21.3 (Juli 2002), S. 681–682. ISSN: 0730-0301. DOI: 10.1145/566654.566636. URL: <https://doi.org/10.1145/566654.566636>.
- [36] Noiseposting. *The Perlin Problem: Moving Past Square Noise*. Website. Zuletzt aufgerufen: 10.03.2025. 2022. URL: <https://noiseposting.ng/posts/2022-01-16-The-Perlin-Problem-Moving-Past-Square-Noise.html>.
- [37] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2. Aufl. Zuletzt aufgerufen: 03.05.2025. Springer, 2022. ISBN: 978-3-030-34371-2. URL: <https://szeliski.org/Book/>.
- [38] DeepAI. *AI Image Generator*. DeepAI Website. Zuletzt aufgerufen: 02.02.2025. 2025. URL: <https://deepai.org/machine-learning-model/text2img>.
- [39] PolyHaven. *PolyHaven HDRIs: Indoor*. PolyHaven Website. Zuletzt aufgerufen: 02.02.2025. 2025. URL: <https://polyhaven.com/hdris/indoor>.
- [40] Blender Online Community. *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam: Blender Foundation, 2018. URL: <http://www.blender.org>.
- [41] Darts 1. *Gelingt der große Wurf? - Dart wissenschaftlich bei Darts 1*. Darts1 Weisite. Zuletzt aufgerufen: 03.02.2025. 2025. URL: <https://www.darts1.de/training/dart-mathematik-4.php>.
- [42] Alisa Favinger. „The Polar Coordinate System“. In: *MAT Exam Expository Papers* 12 (2008). Zuletzt aufgerufen: 03.03.2025. URL: <https://digitalcommons.unl.edu/mathmidexpap/12>.
- [43] Alejandro Domínguez Torres. *Origin and History of Convolution*. <https://www.slideshare.net/slideshow/origin-adn-history-of-convolution/5650913>. Zuletzt aufgerufen: 03.03.2025. 2010.
- [44] Giovanni Capobianco et al. „Image convolution: a linear programming approach for filters design“. In: *Soft Computing* 25.14 (Juli 2021), S. 8941–8956. ISSN: 1433-7479. DOI: 10.1007/s00500-021-05783-5. URL: <https://doi.org/10.1007/s00500-021-05783-5>.
- [45] Irwin Sobel. „An Isotropic 3x3 Image Gradient Operator“. In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).

- [46] C. Harris und M. Stephens. „A Combined Corner and Edge Detector“. In: *Proceedings of the 4th Alvey Vision Conference*. 1988, S. 147–151. URL: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=88cdfbeb78058e0eb2613e79d1818c567f0920e2>.
- [47] Kris Kitani. *Harris Corners*. Zuletzt aufgerufen: 03.03.2025. URL: https://www.cs.cmu.edu/~16385/s17/Slides/6.2_Harris_Corner_Detector.pdf.
- [48] P. V. C. Hough. „Method and Means for Recognizing Complex Patterns“. US3069654A. Issued December 18, 1962. URL: <https://patents.google.com/patent/US3069654A/en>.
- [49] Richard Hartley und Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2. Aufl. Cambridge University Press, 2004, S. 37–44.
- [50] Gilbert Strang. *Introduction to Linear Algebra*. 4. Aufl. Wellesley, MA: Wellesley-Cambridge Press, Feb. 2009, S. 375–379.
- [51] Helder Araujo und Jorge Dias. „An Introduction to the Log-Polar Mapping“. In: *Proc. of Second Workshop on Cybernetic Vision* (Jan. 1996).
- [52] Jens Gravesen. „The metric of colour space“. In: *Graphical Models* 82 (2015), S. 77–86. ISSN: 1524-0703. DOI: <https://doi.org/10.1016/j.gmod.2015.06.005>. URL: <https://www.sciencedirect.com/science/article/pii/S1524070315000247>.
- [53] Zhou Wang et al. „Image quality assessment: from error visibility to structural similarity“. In: *IEEE Transactions on Image Processing* 13.4 (2004), S. 600–612. DOI: [10.1109/TIP.2003.819861](https://doi.org/10.1109/TIP.2003.819861).
- [54] Martin A. Fischler und Robert C. Bolles. „Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography“. In: *Commun. ACM* 24.6 (Juni 1981), S. 381–395. ISSN: 0001-0782. DOI: [10.1145/358669.358692](https://doi.org/10.1145/358669.358692). URL: <https://doi.org/10.1145/358669.358692>.
- [55] Ugo Montanari. „Continuous Skeletons from Digitized Images“. In: *J. ACM* 16.4 (Okt. 1969), S. 534–549. ISSN: 0004-5411. DOI: [10.1145/321541.321543](https://doi.org/10.1145/321541.321543). URL: <https://doi.org/10.1145/321541.321543>.
- [56] B. Spain. *Analytical Conics*. Dover Books on Mathematics. Dover Publications, 2007, S. 14. ISBN: 9780486457734. URL: <https://books.google.de/books?id=qHFXUAau5vcC>.
- [57] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems*. Hrsg. von F. Pereira et al. Bd. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
- [58] J. A. Stegemann und N. R. Buenfeld. „A Glossary of Basic Neural Network Terminology for Regression Problems“. In: *Neural Computing & Applications* 8.4 (Nov. 1999), S. 290–296. ISSN: 1433-3058. DOI: [10.1007/s005210050034](https://doi.org/10.1007/s005210050034). URL: <https://doi.org/10.1007/s005210050034>.
- [59] Conor Sheehan, Ben Day und Pietro Liò. *Introducing Curvature to the Label Space*. 2018. arXiv: 1810.09549 [cs.LG]. URL: <https://arxiv.org/abs/1810.09549>.
- [60] Connor Shorten und Taghi M. Khoshgoftaar. „A survey on Image Data Augmentation for Deep Learning“. In: *Journal of Big Data* 6.1 (Juli 2019), S. 60. ISSN: 2196-1115. DOI: [10.1186/s40537-019-0197-0](https://doi.org/10.1186/s40537-019-0197-0). URL: <https://doi.org/10.1186/s40537-019-0197-0>.
- [61] Arthur P Dempster, Nan M Laird und Donald B Rubin. „Maximum likelihood from incomplete data via the EM algorithm“. In: *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), S. 1–22.
- [62] Jason Wang, Luis Perez et al. „The effectiveness of data augmentation in image classification using deep learning“. In: *Convolutional Neural Networks Vis. Recognit* 11.2017 (2017), S. 1–8.

- [63] Muhammad Hussain. „Yolov1 to v8: Unveiling each variant—a comprehensive review of yolo“. In: *IEEE access* 12 (2024), S. 42816–42833.
- [64] Muhammad Hussain. „Yolov5, yolov8 and yolov10: The go-to detectors for real-time vision“. In: *arXiv preprint arXiv:2407.02988* (2024).
- [65] Zilong Li. „Optimization of autonomous driving target detection in complex scenarios based on improved YOLOv8“. In: *Proceedings of the International Conference on Image Processing, Machine Learning and Pattern Recognition*. 2024, S. 397–402.
- [66] Jing Qin, Degang Yang und Fei Liu. „Lightweight object detection algorithm of UAV aerial image based on improved YOLOv8“. In: *Proceedings of the 2024 8th International Conference on Electronic Information Technology and Computer Engineering*. 2024, S. 134–137.
- [67] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: 2408.15857 [cs.CV]. URL: <https://arxiv.org/abs/2408.15857>.
- [68] Herfandi Herfandi et al. „Real-Time Patient Indoor Health Monitoring and Location Tracking with Optical Camera Communications on the Internet of Medical Things“. In: *Applied Sciences* 14 (Jan. 2024), S. 1153. DOI: 10.3390/app14031153.
- [69] Tsung-Yi Lin et al. „Focal loss for dense object detection“. In: *Proceedings of the IEEE international conference on computer vision*. 2017, S. 2980–2988.
- [70] Zhaozhui Zheng et al. „Distance-IoU loss: Faster and better learning for bounding box regression“. In: *Proceedings of the AAAI conference on artificial intelligence*. Bd. 34. 07. 2020, S. 12993–13000.
- [71] Joseph Redmon et al. „You only look once: Unified, real-time object detection“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 779–788.
- [72] Peiyuan Jiang et al. „A Review of Yolo Algorithm Developments“. In: *Procedia Computer Science* 199 (2022). The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19, S. 1066–1073. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2022.01.135>. URL: <https://www.sciencedirect.com/science/article/pii/S1877050922001363>.
- [73] Ruizhi Zhang et al. „A density-based oversampling approach for class imbalance and data overlap“. In: *Computers & Industrial Engineering* 186 (2023), S. 109747.
- [74] Ahmad S Tarawneh et al. „Stop oversampling for class imbalance learning: A review“. In: *IEEE Access* 10 (2022), S. 47643–47660.
- [75] Agnieszka Mikołajczyk und Michał Grochowski. „Data augmentation for improving deep learning in image classification problem“. In: *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. 2018, S. 117–122. DOI: 10.1109/IIPHDW.2018.8388338.
- [76] Qianqian Tong, Guannan Liang und Jinbo Bi. „Calibrating the adaptive learning rate to improve convergence of ADAM“. In: *Neurocomputing* 481 (2022), S. 333–356.
- [77] Ilya Loshchilov und Frank Hutter. „Decoupled weight decay regularization“. In: *arXiv preprint arXiv:1711.05101* (2017).
- [78] Ricardo Llugsí et al. „Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito“. In: *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*. IEEE. 2021, S. 1–6.
- [79] Sri Nurdjati et al. „PERFORMANCE COMPARISON OF GRADIENT-BASED CONVOLUTIONAL NEURAL NETWORK OPTIMIZERS FOR FACIAL EXPRESSION RECOGNITION“. In: *BAREKENG: Jurnal Ilmu Matematika dan Terapan* 16.3 (Sep. 2022), S. 927–938. DOI: 10.30598/barekengvol16iss3pp927–938. URL: <https://ojs3.unpatti.ac.id/index.php/barekeng/article/view/6105>.
- [80] Jun Tang Zhuang et al. „Adabelief optimizer: Adapting stepsizes by the belief in observed gradients“. In: *Advances in neural information processing systems* 33 (2020), S. 18795–18806.

- [81] Nefeli Gkouti et al. „Should I try multiple optimizers when fine-tuning pre-trained Transformers for NLP tasks? Should I tune their hyperparameters?“ In: *arXiv preprint arXiv:2402.06948* (2024).
- [82] Marshaniswah Syamsul und Suryo Adhi Wibowo. „Optimizers Comparative Analysis on YOLOv8 and YOLOv11 for Small Object Detection“. In: *2024 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*. IEEE. 2024, S. 978–983.
- [83] Ilya Loshchilov und Frank Hutter. „Sgdr: Stochastic gradient descent with warm restarts“. In: *arXiv preprint arXiv:1608.03983* (2016).
- [84] Modesto Medina-Melendrez et al. „Irregular Ellipse Detection Method by Confined Space Sweep“. In: *2021 5th International Conference on Advances in Image Processing (ICAIP)*. ICAIP 2021. ACM, Nov. 2021, S. 20–25. DOI: 10.1145/3502827.3502838. URL: <http://dx.doi.org/10.1145/3502827.3502838>.
- [85] Stefan Behnel et al. „Cython: The best of both worlds“. In: *Computing in Science & Engineering* 13.2 (2011), S. 31–39.
- [86] Siu Kwan Lam, Antoine Pitrou und Stanley Seibert. „Numba: a LLVM-based Python JIT compiler“. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM ’15. Austin, Texas: Association for Computing Machinery, 2015. ISBN: 9781450340052. DOI: 10.1145/2833157.2833162. URL: <https://doi.org/10.1145/2833157.2833162>.