

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

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
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 . . . . .	4
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
<b>2 Simulation von Dartscheiben zur Generierung von Trainingsdaten</b>	<b>9</b>
2.1 Grundlagen . . . . .	9
2.1.1 Prozedurale Datenerstellung . . . . .	9
2.1.2 3D-Rendering . . . . .	10
2.1.3 Kameraparameter . . . . .	10
2.1.4 Binärbilder und Masken . . . . .	11
2.1.5 Dartscheiben-Geometrie . . . . .	11
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 . . . . .	13
2.1.10 Seeding . . . . .	14
2.1.11 Thresholding und Maskierung . . . . .	14
2.1.12 Prozedurale Texturen . . . . .	15
2.2 Methodik . . . . .	15
2.2.1 3D-Szene . . . . .	15
2.2.2 Material und Licht . . . . .	17
2.2.3 Scripting . . . . .	19
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 Dartpfeil-Positionen . . . . .	25
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 Beispiel-Render . . . . .	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

<b>3 Vorverarbeitung von Bildern durch klassische Computer Vision</b>	<b>37</b>
3.1 Grundlagen . . . . .	37
3.1.1 Polarlinien . . . . .	39
3.1.2 Thresholding . . . . .	39
3.1.3 Binning . . . . .	39
3.1.4 Faltung . . . . .	40
3.1.5 Kantenerkennung . . . . .	40
3.1.6 Harris Corner Detection . . . . .	40
3.1.7 Hough-Transformation . . . . .	42
3.1.8 Transformationsmatrizen . . . . .	42
3.1.9 Log-polare Entzerrung . . . . .	44
3.1.10 Farbräume . . . . .	44
3.1.11 Structural Similarity (SSIM) . . . . .	44
3.1.12 RANSAC . . . . .	45
3.2 Methodik . . . . .	45
3.2.1 Warum Computer Vision? . . . . .	45
3.2.2 Vorverarbeitung . . . . .	46
3.2.3 Kantenverarbeitung . . . . .	46
3.2.4 Linienverarbeitung . . . . .	47
3.2.5 Orientierung . . . . .	51
3.2.6 Zusammenführen aller Komponenten . . . . .	54
3.3 Implementierung . . . . .	55
3.3.1 Winkelfindung aus gefilterten Linien . . . . .	55
3.3.2 Farben-Identifizierung . . . . .	55
3.3.3 Klassifizierung von Surroundings . . . . .	56
3.4 Ergebnisse . . . . .	56
3.4.1 Metriken . . . . .	57
3.4.2 Verwendete Daten . . . . .	57
3.4.3 Quantitative Auswertung . . . . .	57
<b>4 Identifizierung von Dartpfeilen mit neuronalen Netzen</b>	<b>63</b>
4.1 Grundlagen . . . . .	63
4.1.1 Was sind neuronale Netze? . . . . .	63
4.1.2 Training Neuronaler Netze . . . . .	64
4.1.3 Terminologie . . . . .	65
4.1.4 Was ist Augmentierung? . . . . .	66
4.1.5 Die YOLOv8-Architektur . . . . .	66
4.2 Methodik . . . . .	67
4.2.1 Die YOLOv8*-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 . . . . .	77
4.4 Ergebnisse . . . . .	78
4.4.1 Metriken . . . . .	78
4.4.2 Datenquellen und Herangehensweise . . . . .	80
4.4.3 Auswertung der Existenz-Metrik $\mu_X$ . . . . .	80
4.4.4 Auswertung der Klassen-Metrik $\mu_K$ . . . . .	81
4.4.5 Auswertung der Positions-Metrik $\mu_P$ . . . . .	82
4.4.6 Auswertung der PCS-Metrik . . . . .	82
<b>5 Diskussion</b>	<b>83</b>
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 von Dartscheiben . . . . .	85

5.2.1	Verwendete Technik . . . . .	85
5.2.2	Zuverlässigkeit des Systems . . . . .	85
5.3	Diskussion der Dartpfeil-Erkennung durch neuronale Netze . . . . .	86
5.3.1	Eigene Implementierung des Modells . . . . .	86
5.3.2	Training des Modells . . . . .	86
5.3.3	Vergleich mit DeepDarts . . . . .	87
<b>6</b>	<b>Fazit</b>	<b>89</b>
<b>7</b>	<b>Ausblick</b>	<b>91</b>
7.1	Ausblick der Datenerstellung . . . . .	91
7.2	Ausblick der Normalisierung . . . . .	92
7.3	Ausblick der Dartpfeilerkennung . . . . .	92

## List of todos

1	Einleitende Sätze der Einleitung. . . . .	1
2	Einleitende Sätze zu Kapitel 2. . . . .	9
3	CV-Implementierung beschreiben. . . . .	55
4	Surroundings-Implementierung . . . . .	56
5	Einleitende Sätze zu Kapitel 4. . . . .	63
6	Einleitende Sätze zur Methodik. . . . .	67
7	Daten zum Training, Graphen etc - EDIT: Das ist teils in Implementierung schon. Überdenken! . . . . .	76
8	Einleitende Sätze zur Implementierung . . . . .	76
9	Trainings-Epochen eintragen . . . . .	77
10	Einleitende Sätze: NN-Ergebnisse . . . . .	78
11	POS-Auswertung beschreiben . . . . .	82
12	PCS-Auswertung beschreiben . . . . .	82
13	Fazit ausformulieren . . . . .	89
14	Ausblick CV . . . . .	92
15	Ausblick NN . . . . .	92
16	Abkürzungen sehen nicht gut aus, aber das zu fixen ist nervig. . . . .	99



# Kapitel 1

## Einleitung

Einleitende Sätze der Einleitung.

### 1.1 Projektübersicht

Darts ist ein in beliebtes Spiel mit vielerlei Spielvariationen und geringer Einstiegsschwelle für neue Spieler. Es ist allseits bekannt und weit verbreitet. 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 Bestimmung 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 [1] und Scolia [2].

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 Computer Vision 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 ihre runde Grundform wiederherstellt. 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 möglich.

Durch die Verwendung eines neuronalen Netzes besteht eine Notwendigkeit einer ausreichenden Anzahl an korrekt annotierten 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 nahezu unendliche Anzahl unterschiedlicher Daten erstellt werden kann.

## 1.2 DeepDarts

Der Anstoß dieses Projekts wurde durch ein Paper gegeben, in dem ein System zur Identifizierung von Dartpfeilen in Bildern mit anschließendem Scoring vorgestellt wurde [3]. Das als DeepDarts bezeichnete 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 von Dartscheiben, trainiert wurde [4]. Das System wurde 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 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 wurde 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 wurde mit unterschiedlichen Aufteilungen der gegebenen Daten trainiert. Zusammenfassend konnte DeepDarts auf Validierungs- und Testdaten Auswertungen mit Werten von PCS = 84.0% bis PCS = 94.7% erzielen. Als zentrale Schwachstelle des Systems wurde das Identifizieren der Orientierungspunkte aufgelistet, welches für fehlerhafte Erkennungen sorgte.

Bei genauer 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 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 Trainingsdaten wurden frontal aufgenommen und zeigten die selbe Ausrichtung der selben Dartscheibe. Die restlichen 2 000 Daten wurden mit einer zweiten Dartscheibe und zu Teilen aus unterschiedlichen Winkeln aufgenommen. Zudem wurde die Korrektheit der gelabelten Daten in Frage gestellt. Laut eigener Auswertung wurde für 1 200 ausgewählte Daten eine Deckungsgleichheit der notierten und annotierten Punktzahlen von 97.6% ermittelt; 29 Bilder wurden 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 massivem Overfitting und ist nicht in der Lage, zuverlässig auf unabhängigen Daten zu generalisieren. Diese Beobachtung wurde als zentrale Erkenntnis vermerkt und hat die Form der 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 als auch 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 wurde sich für die synthetische Erstellung von Trainingsdaten unter Verwendung von 3D-Modellierungssoftware entschieden. Auf diese Weise ist eine automatisierte Datenerstellung ermöglicht, die durch den Einsatz zeitgemäßer Technik prozedural und fotorealistisch erstellt werden kann. Durch einen anfänglichen Mehraufwand des Aufsetzen dieses Systems können anschließend beliebige Datenmengen erstellt werden, ohne dass manuelle Eingriffe notwendig sind. 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 Keypoints 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 Fremdobjekte wie einen Dartschrank als auch durch die Dartpfeile selbst verursacht sein. 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 auf die Genauigkeit der Vorhersage ausschlagen.

Zur Lösung dieses Problems wurde von McNally u. a. die Idee unterbreitet, Redundanz durch das Einbinden weiterer Orientierungspunkte zu schaffen. Dieser Ansatz wurde 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 abhängig von den Trainingsdaten ist. 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 ablaufen.

Aus diesen Gründen wurde sich für diese Thesis dazu entschieden, einen Schritt zurück zu gehen und die Verwendung eines neuronalen Netzes für diese Aufgabe auszuschließen. 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 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

spezifischer 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 Beispielbildern in den Algorithmus aufgenommen werden, beispielsweise die Integration neuer Farbschemata von Dartscheiben.

Der Umsetzung 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 zuletzt 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 Dartfeilspitzen in normalisierten Bildern genutzt. Das Ziel dieses Teils der Thesis ist die korrekte Lokalisierung von Dartfeilen in Bildern zur Ermittlung der erzielten Punktzahl nach einer Dartsrunde.

Die von McNally u. a. 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.

In dieser Arbeit wird daher der Ansatz verfolgt, ein neuronales Netz basierend auf der selben Modell-Familie zu verwenden, wie es in DeepDarts genutzt wurde. Seit der Veröffentlichung von DeepDarts 2021 wurden neue Architekturen der YOLO-Familie vorgestellt, deren Performance über der der verwendeten Architektur liegen. Zudem werden diese Architekturen als Grundgerüst genutzt, um durch Adaptationen 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. Die verwendeten Klassen setzen sich zusammen aus je einer Klasse pro Orientierungspunkt und einer Klasse zur Annotation eines Dartfeils. Für diese Arbeit wurden die Ausgaben derart angepasst, dass keine Bounding Boxes vorhergesagt werden und die vorhergesagten Klassen die Farben der getroffenen Feldern codieren.

Diese Adaptationen der Ausgaben sorgen dafür, dass keine irrelevanten Informationen in den Ausgaben enthalten sind 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 Dartfeil an der Grenze zweier Felder erkannt, ist die Vorhersage trotz ungenauer Normalisierung möglich, da eine Ermittlung des Feldes durch die bestimmte 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 für jeden Teil dieser Thesis eine Forschungsfrage aufgestellt und das Zusammenspiel aller Bestandteile wird in einer weiteren 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 und anschließender 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 echten 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 Teilprojekts 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.

**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 echte Daten umfasst. Auf diese Weise wird die Fähigkeit der Generalisierung des Netzes auf synthetischen Daten im Vergleich mit echten 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 u. a., in der ein System mit dem Namen DeepDarts vorgestellt wurde [3]. Dieses System konnte 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 wurden, sind in dieser Arbeit genutzt wurden, um Vergleichswerte zu setzen. 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 wurden.

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

### 1.7.3 Dart-Scoring

Neben bereits etablierten Multi-Camera-Systemen wie Scolia und Autodarts [2, 1] existieren weitere Herangehensweisen an diese Aufgabe. Scolia und Autodarts 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, jedoch ebenfalls akkurate, System mit fünf Kameras veröffentlicht [18].

Die Idee des einfacher zugänglichen und erschwinglichen Dart-Scorings hat in einigen Hobbyprojekten Anklang gefunden, sodass einige Systeme mit unterschiedlichen Herangehensweisen der Ermittlung eines 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 oder 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 [3]. Die Verwendung neuronaler Netze für Verarbeitung von Bilddaten ist dabei ein neuer Schritt der Einbindung aktueller 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 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 Dartfeilspitzen in normalisierten Bildern ab. Diese Kapitel sind jeweils unterteilt in die Abschnitte Grundlagen, Methodik, Implementierung und Ergebnisse. Im Abschnitt „Grundlagen“ werden Themen und Konzepte eingeführt, die für das Verständnis des Kapitels notwendig sind und nicht im erwarteten Wissensgebiet der Audienz 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 der jeweiligen 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. Abschließend werden in Kapitel 7 Themenbereiche angerissen, die als Einstiegspunkte zur Erweiterung und Verbesserung dieser Masterarbeit identifiziert wurden.

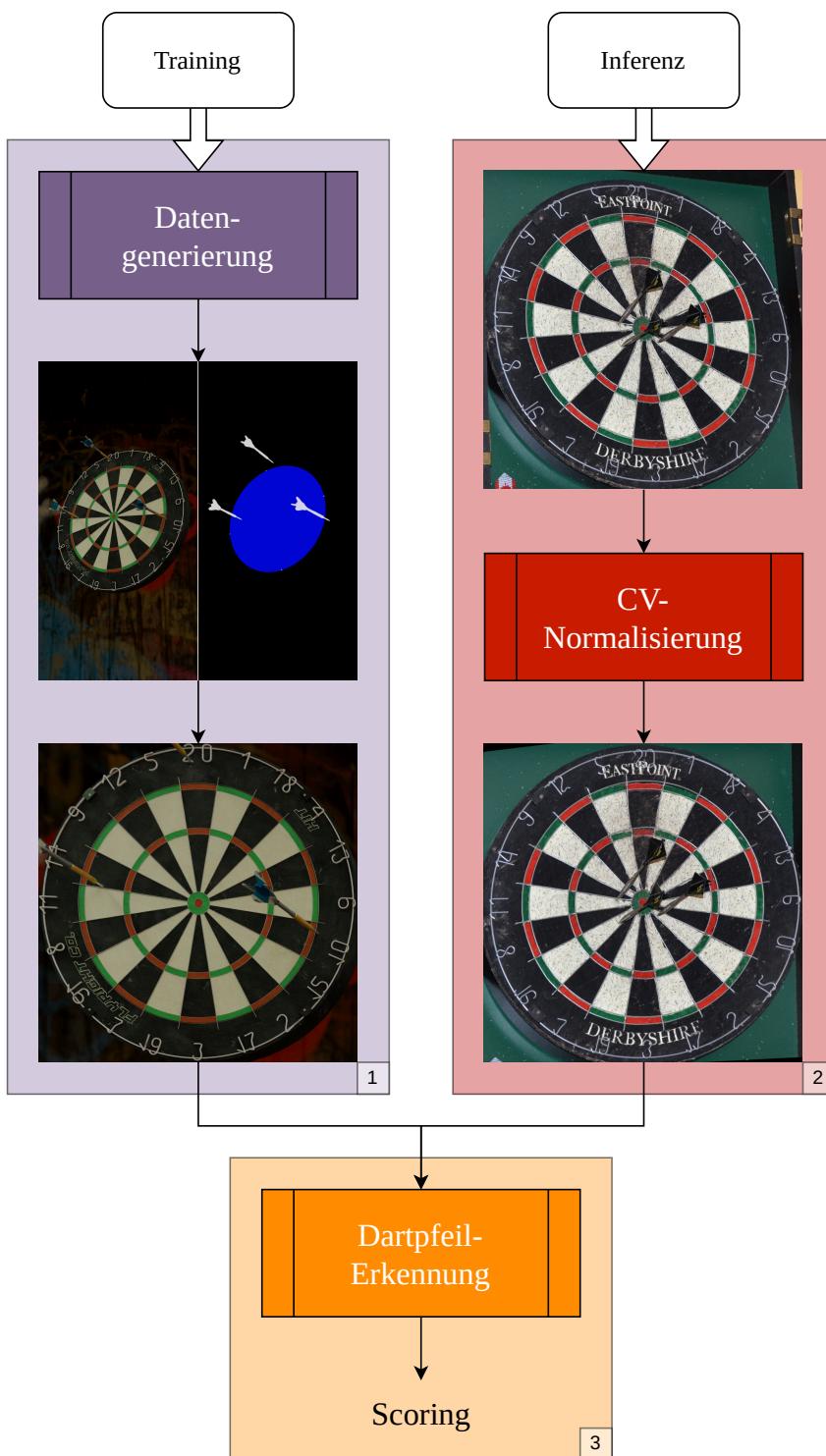


Abbildung 1.1: Überblick über die Projektstruktur. (1) Datenerstellungs-Pipeline; bei der Datengenerierung werden Bilder, Masken und Annotationen erstellt und automatisch normalisiert. (2) Inferenz-Pipeline; beliebige Bilder von Dartscheiben werden algorithmisch normalisiert. (3) Dartpfeil-Erkennung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

# Kapitel 2

# Simulation von Dartscheiben zur Generierung von Trainingsdaten

Einleitende Sätze zu Kapitel 2.

## 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 generellen Datenerstellung, Darts und Texturierung. Die generelle Datenerstellung umfasst prozedurale Datenerstellung in Unterabschnitt 2.1.1 und Grundlagen des 3D-Renderings in Unterabschnitt 2.1.2 sowie Kameraparameter in Unterabschnitt 2.1.3 und Binärbilder und Masken in Unterabschnitt 2.1.4. Der Darts-Abschnitt beginnt mit der Geometrie einer Dartscheibe in Unterabschnitt 2.1.5, gefolgt von für diese Thesis relevante Darts-Terminologie in Unterabschnitt 2.1.6. Danach folgt in Unterabschnitt 2.1.7 die Unterscheidung zwischen Soft- und Steeldarts. Zuletzt folgt der Abschnitt der Texturierung, in dem mit der Klärung der Funktionsweise von Material und Texturen in Unterabschnitt 2.1.8 eingestiegen wird. Darauf folgt eine Übersicht über Noise-Texturen in Unterabschnitt 2.1.9 und Seeding von Zufallsvariablen in Unterabschnitt 2.1.10. Abschließend werden die Begriffe Thresholding und Maskierung in Unterabschnitt 2.1.11 erklärt und die Hintergründe prozeduraler Texturen in Unterabschnitt 2.1.12 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 Erstellung 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 1- und 2-dimensionale 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 wird. 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

Die Erstellung von fotorealistischen Bildern aus 3D-Szenen wird eine Technik verwendet, die als Ray Tracing bezeichnet wird. Beim Ray Tracing 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 sorgen für eine realistische Ausleuchtung einer modellierten Szene. Die weitgehende Unabhängigkeit der Pixel des Bildes ermöglicht eine verteilte Ausführung dieses Algorithmus auf Grafikkarten, um Rendering-Zeiten durch parallele Ausführungen zu optimieren.

Ray Tracing ist der Industriestandard bezüglich der 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 Ray Tracing 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 Ray-Tracing ist die Präsenz einer Kamera unabdingbar. Ursprüngliche Kameramodelle begannen mit einer Pinhole-Kamera, die als Projektion eines 3D-Raumes in einen 2D-Raum vornimmt [27]. Auf diesem Modell aufbauend wurden weitere Kameraparameter modelliert, bis die Simulation echter 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 Echten 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 oberflächliches 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 50mm 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 analog ist er 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 des Bildes 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 die selbe 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 einhergehen. 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 dem aufgenommenen Bild. [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. Dieser Effekt kann in einer Software auf zwei unterschiedliche Arten umgesetzt werden: Simulation oder Komposition. Bei der Simulation wird das eingefangene Licht durch rekonstruierte Linsen gebrochen und die Farbsäume werden direkt durch die Kamera aufgenommen. Dieser Schritt ist rechnerisch aufwendig und wird daher im Vergleich zur Komposition selten eingesetzt. Bei der Komposition wird die Aufnahme in der Nachverarbeitung derart abgeändert, dass der Effekt der Farbsäume nachgestellt wird. Da der Effekt in herkömmlichen Aufnahmen für das ungeschulte Auge schwer erkennbar ist, ist der Unterschied dieser Methoden für diese Anwendung verschwindend gering.

### 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 eine 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 Existenz der Dartpfeile bis zur exakten Bestimmung der Einstichstellen der Dartpfeile in die Dartscheibe.

### 2.1.5 Dartscheiben-Geometrie

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

**Die Dartscheibe** Die Dartscheibe besteht grundlegend aus einer Scheibe mit 451mm 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 8-10mm. Der Triple-Ring ist etwa 10cm vom Mittelpunkt entfernt; der Double-Ring etwa 16cm. Insgesamt ergibt sich ein Durchmesser von 34cm punktezielender Felder, der Bereich jenseits der 17cm Abstand des Mittelpunktes gibt keine Punkte [5].

In der Mitte der Dartscheibe befinden sich die Felder Bull und Double Bull (oder Bull's Eye). Sie haben Durchmesser jeweils ca. 32mm und 12,7mm. Das Bull gibt 25 Punkte, das Double Bull 50.

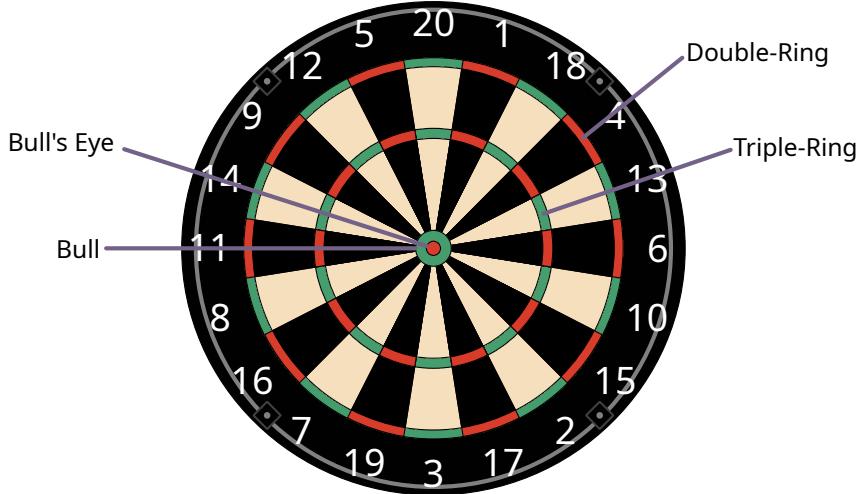


Abbildung 2.1: Schematische Darstellung einer Dartscheibe.

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.

**Rund um die Dartscheibe** Die Punktzahlen der Dartfelder werden durch aus Drähten 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 die lediglich verdrängt, wodurch die bleibenden Schäden der Einstichlöcher gering gehalten werden.

**Double und Triple** Mit Double und Triple (auch Treble) werden die Ringe an Feldern bezeichnet, die Vielfache 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  $3 \times 20 = 60$  Punkten.

### 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. Die vier wesentlichen Bestandteile sind Tip, Barrel, Shaft und Flight<sup>1</sup>, aufgezählt von der Vorderseite nach hinten [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  $90^\circ$  zueinander stehen und orthogonal zum Shaft verlaufen.

<sup>1</sup>Abweichungen dieser Bezeichnungen sind möglich, sodass der Shaft auch häufig als Stem bezeichnet wird. Zur Vereinheitlichung wurden die genannten Begriffe verwendet.

**Spinne** Als Spinne wird der Metallrahmen der Dartscheibe bezeichnet, der die Feldsegmente voneinander trennt. Die Ausprägung der Spinne ist unterschiedlich, jedoch ist ein tendenzieller 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 auf der Dartscheibe landet.

### 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 Zielscheibe besitzen. Diese Scheibe besteht üblicherweise aus Sisalfasern, die durch die Dartpfeile nicht irreparabel beschädigen. Dem gegenüber stehen Softdarts, bei welchem Pfeile mit Spitzen aus Kunststoff verwendet werden, die typischerweise auf eine elektronische Dartscheibe geworfen werden.

Wesentlicher Unterschied zwischen Soft- und Steeldarts ist neben der unterschiedlichen Ausrüstung die Art des Scorings. Während die meist elektronischen Dartscheiben von Softdarts ein automatischen 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 er 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 der Textur 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 gibt unterschiedliche Arten von Noise-Texturen, die für die Generierung zufälliger Texturen verwendet werden [32]. In dieser Arbeit wurden vorgehend White Noise und Perlin Noise in der Datengenerierung genutzt, daher werden diese in den folgenden Unterabschnitten genauer erläutert.

#### 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 1-dimensionalen Beispiel von Audio entspricht weißes Rauschen der Charakteristik, dass jede Frequenz in einem Signal gleichermaßen vertreten ist. Hinsichtlich einer diskreten 2-dimensionalen Textur ist die Wahl jedes Intensitätswerts eines Pixels unabhängig voneinander gleichverteilt über das Intervall  $[0, 1]$ .

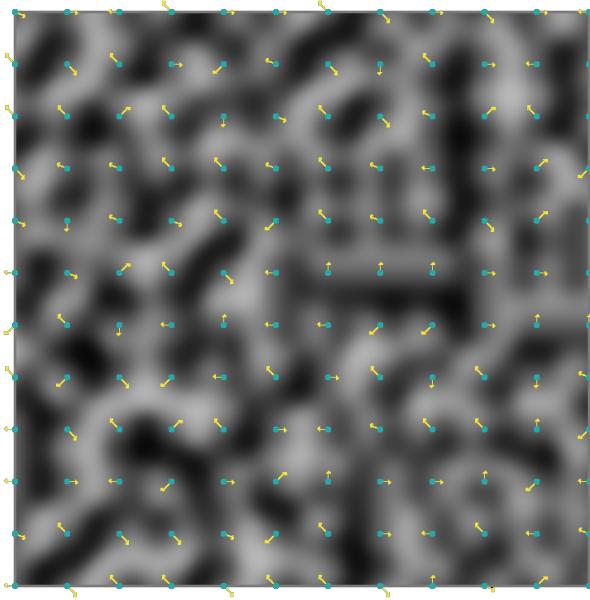


Abbildung 2.2: Generierung von Perlin Noise [36]. Zufällige Rotationsvektoren werden uniform über ein Bild verteilt. Durch diese wird die Intensität der Textur bestimmt.

### 2.1.9.2 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 Gitterauflösung 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 werden von Computern generierte Zufallszahlen auch als pseudo-zufällig bezeichnet. Die Generierung von Zufallszahlen beruht auf deterministischen Algorithmen, die üblicherweise einen Seed verwenden, um Zahlen zu generieren. Durch die Verwendung des selben Seeds ist das Ziehen von Zufallszahlen deterministisch wiederholbar, wodurch die Generierung von Daten reproduzierbar und nachvollziehbar ist.

### 2.1.11 Thresholding und Maskierung

Im Rahmen der Datengenerierung dieser Thesis werden Thresholding und Maskierung verwendet, um Texturen gezielt miteinander zu kombinieren. Beim Thresholding wird in der vorliegenden Arbeit ein definierter 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. Alternativ kann auch das Gegenteil umgesetzt werden. Ein typischer Anwendungsfall des Thresholdings in dieser Arbeit besteht in der Begrenzung von Noise-Texturen auf Bereiche mit den höchsten Pixelintensitäten oberhalb eines definierten Schwellenwerts.

Thresholding ist definiert durch [37]:

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

Wird dieses globale 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 der 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 Kern 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 begrenzt sein, die konkrete Änderung unterliegt jedoch weiterhin der Pseudo-Zufälligkeit. Unter Einbindung dieser Technik kann eine beliebige Anzahl unterschiedlicher Texturen generiert werden.

## 2.2 Methodik

In diesem Abschnitt wird die Methodik der automatisierten Datenerstellung thematisiert. Es werden die unterliegenden Konzepte erläutert und die Hintergründe dieser werden 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, welche in Unterabschnitt 2.2.1 näher beschrieben wird. In der Szene befinden sich Objekte, 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. Diese Randomisierung geschieht durch ein externes Skript, welches in Unterabschnitt 2.2.3 näher betrachtet wird. Nachdem alle Objekte vorbereitet sind, folgt das Rendering der Szene, welches mit Imperfektionen wie Rauschen und Verzerrungen, wie es in Kameras aus Mobiltelefonen vorkommt, angereichert wird, wodurch ein weiterer Schritt des Realismus der erstellten Daten erzielt wird. Zusätzlich werden binäre Masken unterschiedlicher Objekte generiert, welche zur Extraktion relevanter Informationen in den Daten relevant sind. Zuletzt werden durch Nachverarbeitungsschritte, welche in Unterabschnitt 2.2.4 detailliert beschrieben werden, in den Daten enthaltene Metainformationen extrahiert und explizit gespeichert. Im Wesentlichen umfasst dies die Lokalisierung der Dartpfeilspitzen in den gerenderten Bildern und die Ermittlung der Normalisierung durch zuvor erwähnte Masken.

### 2.2.1 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

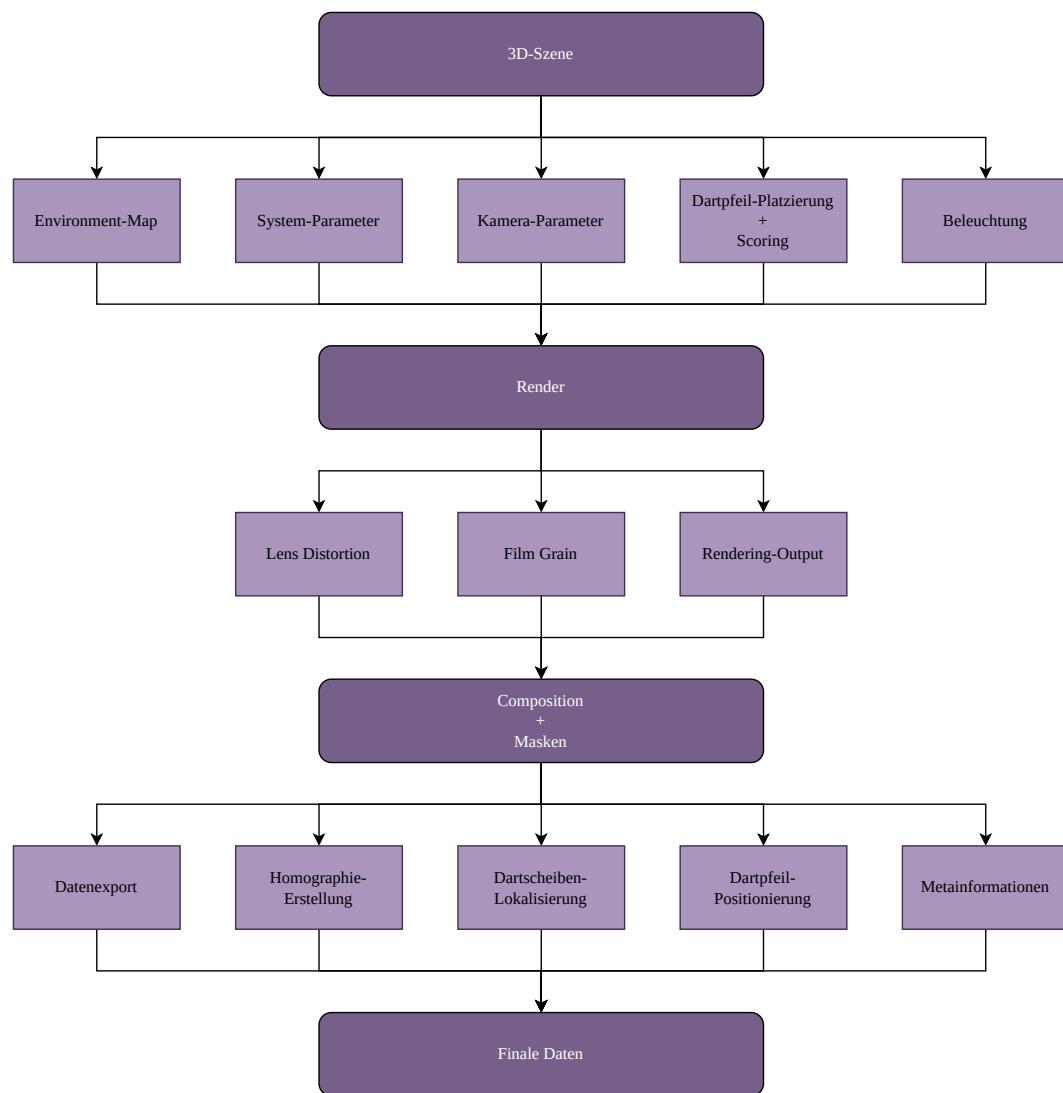


Abbildung 2.3: Rendering-Pipeline

erstellten Daten ausschlaggebend sind. An Objekten werden die Dartscheibe in Unterabschnitt 2.2.1.1, die Dartpfeile in Unterabschnitt 2.2.1.2 und die Beleuchtungsmöglichkeiten in Unterabschnitt 2.2.2.3 beschrieben, sowie die globalen Parameter der Szene in Unterabschnitt 2.2.1.5.

### **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 wurde gemäß der Richtlinien „Playing and Tournament Rules“ der World Dart Federation (WDF) erstellt [5]. Beschreibungen dieses Regelwerks wurden in dieser Thesis als Quelle für Maße und Toleranzen genutzt; Dartscheiben mit esoterischen Maßen und Feldfarben, die nicht konform mit den Regeln sind, wurden nicht explizit für diese Arbeit mit einbezogen. Zusätzlich zu diesen Regeln wurden unterschiedliche reale Dartscheiben als Referenzen genutzt, die reale Gebrauchsspuren aufweisen und anhand derer zu erwartende Beschaffenheiten eingefangen wurden.

### **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 Beleuchtungsmöglichkeiten verwendet. Globale Beleuchtung wird durch die Verwendung von Environment 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. Für die 3D-Szene wurde daher ein Dartschrank aus Holz modelliert, dessen Holzfarbe 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 wird in Form von Abnutzung durch Kratzer, Einstichlöcher und Staubansammlung sowie durch Risse im Material. 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 teilweise reflektives Metall modelliert. Die Spinne der Dartscheibe ist ebenfalls von einem Altersprozess betroffen, da beobachtet wurde, dass die Dicke und Präsenz der Spinne bei Dartscheiben zunehmenden Alters stärker ausgeprägt sind. Historisch ist dies dadurch begründet, dass der Herstellungsprozess von Dartscheiben im Laufe der Zeit fortschrittlicher wurde. Alte Spinnen

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 verformt, sodass die 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 zufällig generierten 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 wurden, die zufällig hinter eine Tip gefügt werden. Darüber hinaus wird die Länge der Barrels zufällig variiert, um weitere Variationen einzubinden. Die Beschaffenheit von Barrels realer Dartpfeile ist sehr unterschiedlich, sodass eine sehr große Variabilität ihrer Erscheinungsbilder möglich ist. Um weitere Variabilitäten einzubinden, wurden teilweise 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 teilweise mit zufälligen Farben versehen. Reale Dartpfeile folgen häufig einem kohärenten Farbschema während die Abstimmung von Barrels und Shafts in dieser Thesis zufällig ist. Hinsichtlich der Variabilität der Dartpfeile ist diese Herangehensweise jedoch präferiert.

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

wurden in dieser Thesis durch Projektion eines zufälligen Bereichs eines Texturatlas<sup>2</sup> 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 wurden fünf unterschiedliche Arten der Beleuchtung modelliert.

**Environment Maps** Environment Maps, auch als HDRI bezeichnet, sind 360°-Scans von realen Umgebungen. Diese können bei dem Rendern von Szenen als Hintergrund genutzt werden, sodass die Farben 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 von  $1/2$  die Helligkeit der Environment Map reduziert, sodass eine Spanne unterschiedlicher Beleuchtungen unter der Verwendung der selben Environment Map möglich ist.

**Kamerablitz** Wenige Zentimeter neben der Kamera befindet sich ein Punktlicht, das als Kamerablitz fungiert. Es kann ein- und ausgeschaltet werden und sorgt unter dessen Verwendung für eine helle Ausleuchtung der Szene. Die Farbe des Lichtes ist kaltweiß und sorgt durch sein Positionierung für harte Kanten entlang der Kanten 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 variierender Größe modelliert. Diese Art der Beleuchtung wurde im Strongbows Pub<sup>3</sup> in Kiel beobachtet.

**Ringlicht** Ein typisches Accessoire für Dartscheiben sind Ringlichter. Diese bestehen aus einem Gestell, das an der Dartscheibe befestigt wird und an dem LEDs in einem Ring vor dieser 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.

### 2.2.3 Scripting

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

---

<sup>2</sup>Der Texturatlas beinhaltet Länderflaggen sowie geometrische Formen und zufällige Farben. Teile des Atlas wurden durch die Bilderstellungs-KI von DeepAI [38] generiert.

<sup>3</sup><https://www.strongbowspub.de>

### 2.2.3.1 Wozu externes Skript?

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, jedoch nicht ihre Positionierung oder Existenz. Der Seed beeinflusst lediglich das Aussehen und die Beschaffenheit der Dartscheibe sowie die Zusammensetzung der Dartpfeile.

Der Seed hält einen zufälligen Wert, jedoch ist diesem Wert keine konkrete Bedeutung zugeschrieben, da er zur Generierung von Zufallsvariablen genutzt wird, 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 Anblick der Dartscheibe und der Dartpfeile nieder.

### 2.2.3.3 Spezifisches Setzen von Parametern

Der Seed und der Alters-Parameter 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 Dartpfeil-Positionen ein wichtiger Schritt des Skriptes. Diese werden anhand von Wahrscheinlichkeitsverteilungen auf der Dartscheibe verteilt und zufällig rotiert. Aus diesen Positionen kann die erzielte Punktzahl des simulierten Dartspiels abgeleitet werden.

Die Kamera wird von dem Skript in einem vordefinierten Raum positioniert und ihre internen sowie externen Parameter werden in definierten Intervallen randomisiert, zu Teilen auch basierend auf ihrer Position. 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, besteht die Gefahr der Spezialisierung eines aus diesen Daten trainierten Systems auf diese Gegebenheiten. Dadurch besteht die Gefahr der fehlerhaften Inferenz auf Daten, die nicht diese exakten internen Kameraparameter vorweisen. Durch Randomisierung der Parameter wird diese Einschränkung der Generalisierungsfähigkeit umgangen.

### 2.2.3.4 Statische und dynamische Objekte

Die Präsenz und 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. Diese sind in jeder Zusammensetzung der Szene vorhanden, obgleich die Environment Map lediglich sehr fade erkennbar ist.

Entgegen der Annahme, die Dartpfeile seien ebenfalls statische Objekte, sind diese nicht in jeder Szene vorhanden. Da die variable Anzahl an Würfen abgebildet werden muss, ist ein zufälliges Ausblenden der Dartpfeile möglich. Dadurch besteht die Möglichkeit, dass alle Dartpfeile ausgeblendet werden und eine leere Dartscheibe abgebildet wird.

Darüber hinaus werden jegliche Lichtobjekte zu den dynamischen Objekten gezählt, da sie durch das Skript ein- und ausgeblendet werden können. Obwohl die Existenz mindestens einer Lichtquelle vorgegeben ist, ist keines der Lichtobjekte statisch in jeder Szene vorhanden. Diese Tatsache fußt ebenfalls auf der Unterbindung systematischer Fehler durch einseitige Modellierung der Szene.

Zuletzt ist ebenfalls der Dartschrank ein dynamisches Objekt, dessen Existenz von dem Skript gesteuert wird. Diese Existenz ist an die Abwesenheit gewisser Lichtobjekte geknüpft, sodass keine ungewollte Überschneidungen von Objekten in der Szene vorhanden sind.

### 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 wurde, 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 wurde. Zusätzlich zu diesem Bild werden weitere Masken von Objekten gerendert. Diese Masken werden als binäre Schwarzweißbilder exportiert und zeigen lediglich einzelne Objekte. Unter anderem werden die Dartpfeile, die Fläche der Dartscheibe, die Schnittpunkte der Dartpfeile und der Dartscheibe sowie Orientierungspunkte der Dartscheibe, wie sie im DeepDarts-System verwendet wurden, generiert. Durch diese Masken wird die Ableitung unterschiedlicher Informationen aus dem Bild genutzt.

Zusätzlich zum Speichern der Bilder werden Metainformationen zu der Szene gespeichert. 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 durch ein weiteres Skript. Durch dieses werden weitere Informationen von den gespeicherten Informationen angeleitet und für das Training eines datengestützten Systems vorbereitet.

#### 2.2.4.1 Normalisierung der Dartscheibe

Für das Training des neuronalen Netzes zum Scoring von Dartsrunden wird in dieser Thesis von normalisierten Bildern ausgegangen. Da das Ziel der Datenerstellung eine Nachempfindung möglichst realistischer Daten ist, die in ihrem Umfang möglichst wenig beschränkt werden, besteht eine Diskrepanz zwischen gerenderten Bildern und Netzwerk-Inputs. In der Inferenz des in dieser Thesis entwickelten Systems wird diese durch die in Kapitel 3 dargestellte Normalisierung geschlossen. Für den Schritt des Trainings wird diese Diskrepanz durch einen Nachverarbeitungsschritt in der Datenerstellung angegangen; die Ausgabe normalisierter Trainingsdaten ist daher Teil der Datenerstellung.

Die Normalisierung der Dartscheibe basiert auf Orientierungspunkten, deren Positionen relativ zur Dartscheibe bekannt sind. Diese Orientierungspunkte werden in den gerenderten Bildern durch binäre Masken ermittelt. Da die Positionen aller Punkte in den normalisierten Bildern bekannt sind, ist ein Mapping der Orientierungspunkte der gerenderten Bilder auf ihre Zielpunkte trivial zu ermitteln. Diese Herangehensweise hat sich bereits im DeepDarts-System bewährt und wird daher analog in dieser Thesis angewendet [3].

Dazu werden 4 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. Genauere Hintergrundinformationen zu den Techniken werden dort ebenfalls erläutert.

#### 2.2.4.2 Identifizierung von Dartpfeil-Positionen in Bildern

Beim Rendering der Bilder werden unter anderem Masken der Einstichstellen von Dartpfeilen 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 Positionen im Bild und erzielten Punktzahlen ermöglicht. Der Hintergrund der Notwendigkeit dieser Überlagerung liegt in der Speicherung der Daten: Die Punktzahlen werden anhand der Dartpfeil-Indizes sortiert während die Positionen als einzelne Maske exportiert wird. Durch die einzelnen Masken der Dartpfeile lässt sich ein korrektes Mapping herstellen.

Unter Verwendung der im vorherigen Unterabschnitt erläuterte Normalisierung der Dartscheibe lassen sich die Positionen der Dartpfeile im Ausgangsbild auf normalisierte Positionen 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 Darts 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 Statistiken über Sample erheben

Zusätzlich zu den essenziellen Informationen zum Trainieren eines neuronalen Netzes werden Statistiken 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 ein Maß zur Überdeckung der Dartpfeile zueinander. Darüber hinaus wird festgehalten, wie viele der Einstichlöcher durch andere Dartpfeile verdeckt sind und damit nicht eindeutig zu sehen sind.

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

## 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. Es wird in einem ersten Unterabschnitt betrachtet, wie die Implementierung der Parameter in der Dartscheibe realisiert ist und wie die Texturen

### 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, welches durch seine Beschaffenheit an Sisal angelehnt ist und farblich einer neuen Dartscheibe entspricht.

Der Einfluss des Seeds für diese Dartscheibe beläuft sich auf eine leichte Variation der Farben und Oberflächenstruktur. Mit zunehmendem Altersfaktor werden die Farben der Felder zwischen dem Grundfarbton und einer alten Variante der Feldfarben 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 mit zunehmendem Alter und einer Verschiebung der Zahlen beeinflusst. Die Verschiebung dieser Objekte wird durch eine auf Perlin Noise beruhende Translation der Vertices im Mesh realisiert. Die Magnitude dieser Translation ist durch den Altersfaktor bestimmt, sodass neue Dartscheiben minimale Verschiebungen aufweisen und alte Dartscheibe sehr starke Verschiebungen mit sich ziehen. Zusätzlich wirkt sich der Altersfaktor bei Spinne und Zahlen aus, indem Rost mit zunehmendem 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 zudem Gebrauchsspuren durch Einstichlöcher, Risse und Staubpartikel auf, die ebenso wie ihre Farbe von den Parametern des globalen Seeds und des Alters 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.

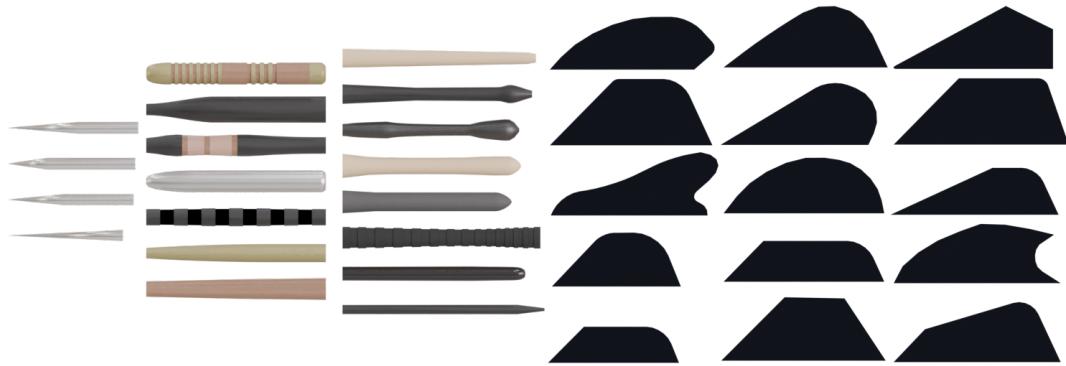


Abbildung 2.4: Bestandteile von Dartpfeilen. Von links nach rechts: Tips, Barrels, Shafts und Flights.

**Einstichlöcher** Die Einstichlöcher werden durch ein Zusammenspiel mehrerer Masken generiert. Zum einen existiert eine Maske, die uniform verteilt Einstichlöcher über die gesamte Dartscheibe verteilt. Diese wird moduliert durch eine Maske, die die Stärke der Existenz bestimmt. An den Stellen, an denen die Existenzmaske stark vorhanden ist, werden Einstichlöcher vermehrt sichtbar als an schwachen Stellen der Existenzmaske. Diese Verteilung ist zu diesem Punkt unabhängig von dem Altersfaktor und lediglich durch den globalen Seed parametrisiert. 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 jeweils die Ausprägung der Risse und ihre Existenz bestimmen. Die Ausprägung der Risse ist durch eine verzerrte Voronoi-Textur erzielt, 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.

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

### 2.3.2 Zusammensetzung der Dartpfeile

Die Umsetzung der Generierung von Dartpfeilen beruht auf der Nutzung von Geometry Nodes in Blender. Geometry Nodes bieten die Möglichkeit der deskriptiven Zusammensetzung von Objekten. 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 4 Objekten gewählt und ihre Spitze wird zum Ursprung des finalen Dartpfeils. Ihre Textur der Tip wird ebenfalls auf Grundlage des Seeds zufällig gesetzt, sodass ein möglichst großes Spektrum unterschiedlicher Dartpfeilspitzen abgedeckt wird.



Abbildung 2.5: Auswahl zufällig erstellter Dartpfeile des Systems.

**Barrels** Auf die Tip folgt die Platzierung der Barrel. Die Barrel wird aus einem Pool von 7 Objekten ausgewählt, deren Ursprung jeweils derart positioniert ist, dass eine lückenlose Platzierung an der Tip ermöglicht ist. 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. Die Spanne der unterschiedlichen Farben, die eine dynamische Barrel annehmen kann, ist im Gegensatz zu dem Farbspektrum der Tips größer, da alle RGB-Kanäle und die Oberflächenbeschaffenheit variabel sind. 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.

**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  $90^\circ$  platziert sind. Um die Variation der Formen einzufangen, wurden 15 unterschiedliche Formen für Flights modelliert, die sich an realen Formen von Flights orientieren. Ihre Textur wird aus einem Texturatlas mit einer Größe von  $1920 \times 1920$  Pixeln gesampled. Dieser besteht aus 9 unterschiedlichen Grundtexturen, die aus Landesflaggen und abstrakten Formen unterschiedlicher Farbpaletten bestehen. 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 lediglich gleiche Dartpfeile existieren. Eine Variation der Dartpfeile innerhalb einer Szene ist möglich, es wurde sich jedoch gegen diese Art der Umsetzung entschieden, da die Verwendung unterschiedlicher Dartpfeile für den selben Wurf 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.

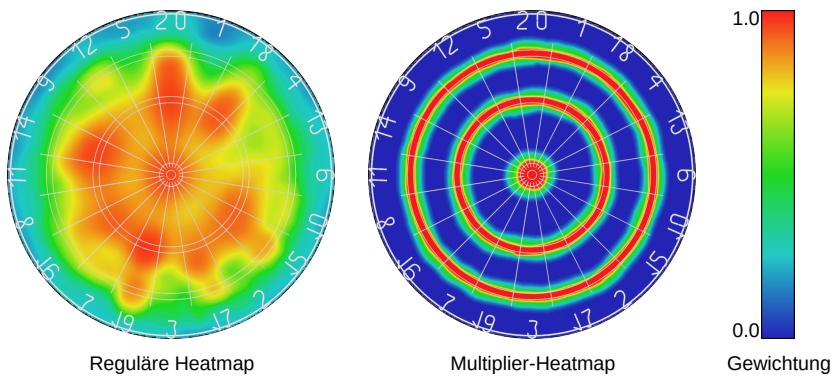


Abbildung 2.6: Heatmaps für die Datenerstellung; (links) Generelle Heatmap; (rechts) Multiplier-Heatmap für Oversampling der Daten.

### 2.3.3 Generierung von Dartpfeil-Positionen

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. Unterabschnitt 2.3.3.1 thematisiert die Positionierung der Dartpfeile auf der Dartscheibe während die Bestimmung der Existenz von Dartpfeilen zur Variation der Anzahlen der Dartpfeile in Unterabschnitt 2.3.3.2 erklärt wird. Nachdem die Existenz und Position der Dartpfeile gesetzt ist, wird die Rotation in Unterabschnitt 2.3.3.3 erklärt. Zuletzt wird die Bestimmung der erzielten Punktzahl in Unterabschnitt 2.3.3.4 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 wurden in Form von Heatmaps in die Szene eingearbeitet.

Die für die Datengenerierung dieses Thesis genutzten Heatmaps sind in Abbildung 2.6 dargestellt. Es wurden zwei unterschiedliche Heatmaps genutzt: Eine realistische Heatmap 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 generelle Heatmap orientiert sich an den für DeepDarts gefundenen Wahrscheinlichkeitsverteilungen [3], Verteilungen aus Online-Recherchen [39] 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 der 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, können gezielt Positionen forcierter werden. So wurde 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 jeder 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 wurde sich gegen eine Adaption der Position entschieden, um zu starke Abweichung von Heatmaps und erneute Überschneidung der Spinne zu vermeiden; eine neue Positionierung des Dartpfeils wurde 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 wurden, 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  $0^\circ$  mit einem Clipping einer maximalen Rotation von  $\pm 15^\circ$ . Die Rotation entlang ihrer  $z$ -Achse ist uniform im Intervall  $[0^\circ, 360^\circ]$ .

### 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 Winkel und Abstände lassen sich die Dartfelder identifizieren, in denen die Dartpfeile eingetroffen sind. Auf diese Weise lässt sich für jeden Dartpfeil ermitteln, in welchem Feld dieser eingetroffen ist und welche Punktzahl durch ihn erzielt wurde.

## 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 Erscheinungsbilder auf. Dieser Unterabschnitt ist dafür vorgesehen, die Umsetzungen der komplexen Parameter genauer darzustellen. Es wird begonnen mit der Betrachtung der Kamerapositionierung in Unterabschnitt 2.3.4.1. Das Setzen der Brennweite der Kamera wird in Unterabschnitt 2.3.4.2 beschrieben und die Wahl von Seitenverhältnis und Auflösung des exportierten Bildes wird in Unterabschnitt 2.3.4.3 beschrieben. Danach folgt in Unterabschnitt 2.3.4.4 die Wahl des Fokuspunkts und abschließend wird in Unterabschnitt 2.3.4.5 die Umsetzung von verwackelten Bildern dargestellt.

### 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**  $\phi_h$ : Öffnungswinkel des Kegels zu den Seiten der Dartscheibe
- **Vertikaler Winkel**  $\phi_v$ : Öffnungswinkel des Kegels in die Höhe
- **Kameraabstand** ( $d_{\min}, d_{\max}$ ): Minimaler und maximaler Abstand der Kamera von der Dartscheibe

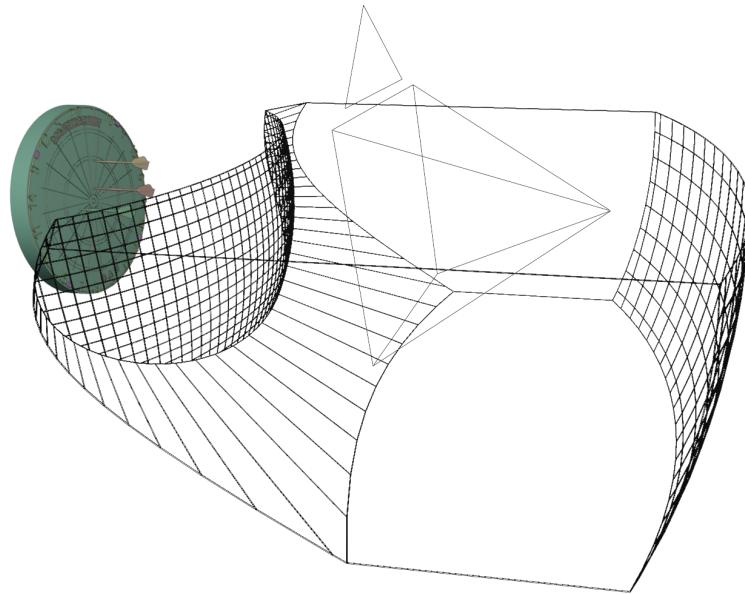


Abbildung 2.7: Darstellung des Kamerabereichs, der zur Erstellung der Daten verwendet wurde.

- **Kamerahöhe** ( $y_{\min}, y_{\max}$ ): Minimale und maximale Höhe der Kamera im Raum<sup>4</sup>
- **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  $l$  der Kamera in Abhängigkeit ihrer Distanz zur Dartscheibe gesetzt. Die Spanne der Brennweite reicht von 18mm bis 60mm. 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 verwandelter Brennweite und Abstand zur Dartscheibe gewonnen unter Beibehaltung der Variabilität der Daten und Einfluss von Zufallswerten.

---

<sup>4</sup>Es wird von einem realen Raum ausgegangen, in dem der Mittelpunkt der Dartscheibe auf einer Höhe von 2,07m angebracht ist.

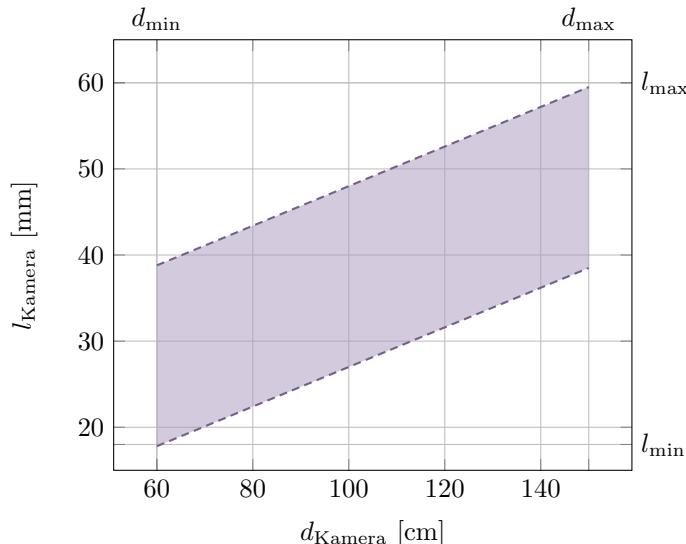


Abbildung 2.8: Abhängigkeit der Brennweiten von dem Abstand zur Dartscheibe. Ober- und Untergrenzen sind durch Strichlinien angegeben, der farblich hervorgehobene Bereich stellt die Spanne möglicher Brennweiten für jeweilige Entfernung dar.

### 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  $2/3$  der Aufnahmen vertikal und in  $1/3$  der Aufnahmen horizontal. Die Auflösung der Kamera wird uniform im Intervall  $[1000 \text{ px}, 4000 \text{ px}]$  gewählt, welches die Pixelzahl entlang der längeren Seite angibt.

### 2.3.4.4 Fokuspunkt

Der Fokuspunkt der Kamera ist in der Szene durch ein eigenes Objekt definiert, sodass die Kamera den Ursprung dieses Objektes fokussiert. Dieses Objekt wird im Umfeld um die Dartscheibe platziert. Ausgehend vom Dartscheibenmittelpunkt wird es normalverteilt mit den Standardabweichungen  $\sigma_x = \sigma_z = \frac{r_D}{3}$  und  $\sigma_y = 2 \text{ cm}$  platziert.  $x$ - 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, jedoch nicht deterministisch.

### 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  $\frac{2 \text{ cm}}{3}$  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 unterschiedlicher gewünschter Erscheinungsbilder sorgt für viele Möglichkeiten zur Anpassung der Farbaufnahme der Kameras in den Szenen. Für einen Cartoon ist beispielsweise ein anderer Umgang mit Farben bei dem Rendern von Bildern erwünscht als für eine cinematiche Szene. Die für diese Thesis verwendeten Farbräume wurden derart gewählt, dass Farben möglichst realistisch dargestellt werden. Dazu wurde als Darstellunggerät und Sequencer sRGB mit einer AgX als Anzeigetransformation gewählt. Trotz häufiger Korrekturen von Handykameras hinsichtlich Kontrasterhöhung der

Aufnahmen wurde sich für einen neutralen Basiskontrast entschieden. 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 im DeepDarts-System verwendet wurden [3]. 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). Die Positionen dieser Punkte im entzerrten Bild sind bekannt, weshalb die Verschiebungen berechnet und die Homographie zur Entzerrung des Bildes abgeleitet werden kann. In der Szene befindet sich ein Objekt, welches aus vier einzelnen Punkten an den Positionen der Orientierungspunkte befindet. Dieses Objekt wird als Maske ausgehend von der finalen Kameraposition und mit den finalen 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.

Diese Art der Identifizierung der Dartscheibenorientierung ist jedoch nicht ideal und zieht Ungenauigkeiten mit sich. Genauer wird auf diese Ungenauigkeiten in der Diskussion in Abschnitt 5.1 eingegangen.

## 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 werden exemplarische Daten aufgezeigt. Anschließend wird ein Überblick über die Rahmenbedingungen der Datenerstellung gegeben, bevor mit einer qualitativen Auswertung der gerenderten Bilder fortgefahrt wird. Darauf folgen Einblicke in die Metainformationen der erstellten Daten. Danach wird auf die Korrektheit der Daten eingegangen und abschließend werden Ungenauigkeiten bei der Erstellung der Daten aufgezeigt.

Eine qualitative Auswertung der Bilddaten ist mangels aussagekräftiger Metriken nicht durchgeführt worden. Eine objektive Bewertung der Realitätsnähe von Bildern ist äußerst komplex und die Aussagekraft nicht eindeutig.

### 2.4.1 Beispiel-Render

In Abbildung 2.9 sind Bilder aus den für diese Arbeit erstellten Daten dargestellt. 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 6 exemplarische Daten, 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 Trainingsdaten des DeepDarts-Systems. Ebenfalls ist eine Variation der Hintergründe und Beleuchtungen offensichtlich. 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 Designs und Positionierungen in den Bildern vorhanden. Während in einigen Bildern alle Dartpfeile vorhanden sind, steckt in einem der dargestellten Bilder kein Dartpfeil in der Dartscheibe.

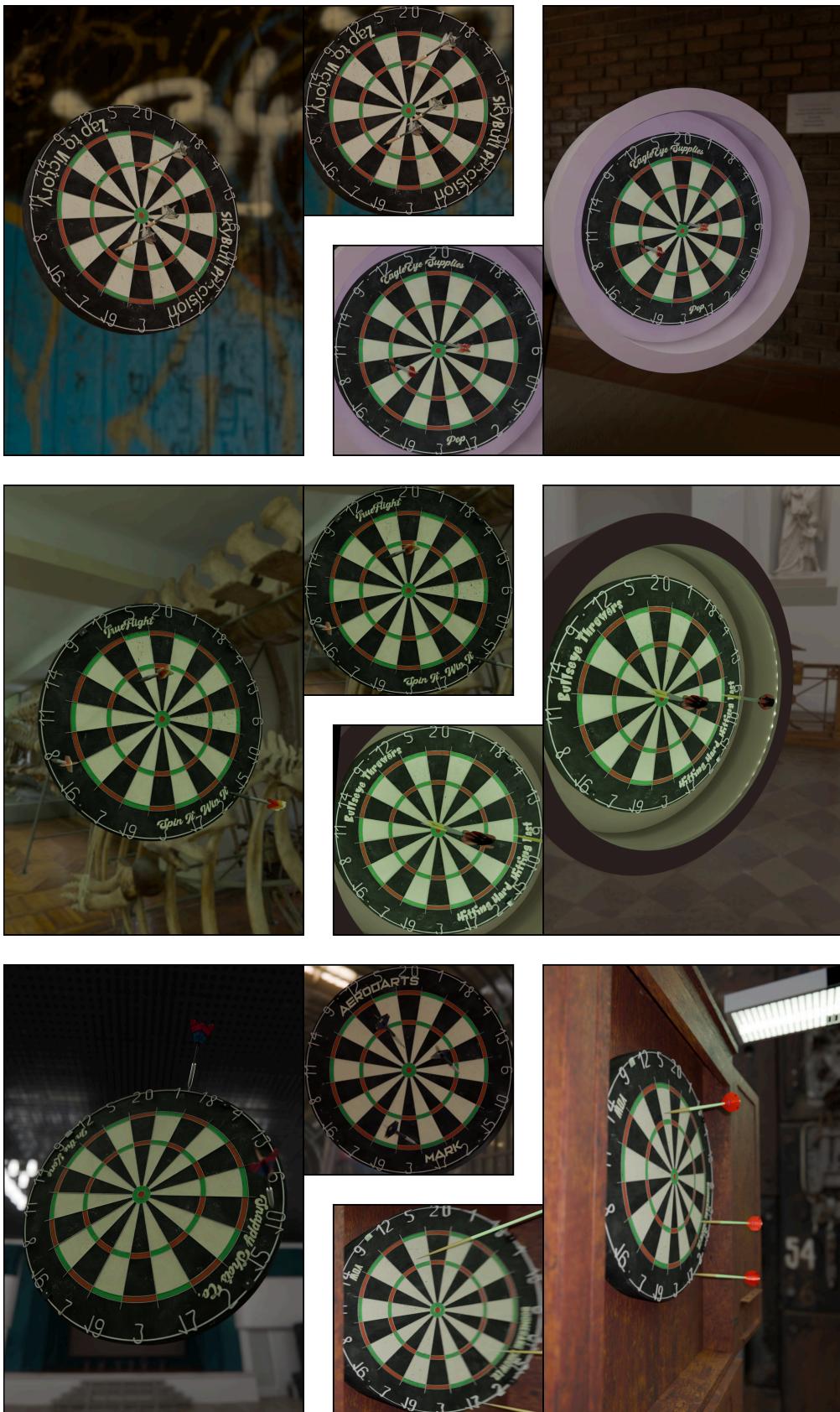


Abbildung 2.9: Gerenderte Bilder der Datenerstellung. Große Bilder an den Seiten sind Render-Outputs, kleine quadratische Bilder der mittleren Spalte sind dazugehörige normalisierte Bilder der Render.

Unterschiedliche Seitenverhältnisse sind aus Gründen der Übersichtlichkeit bewusst nicht dargestellt worden, jedoch existiert eine uniforme Verteilung aller vordefinierter Seitenverhältnisse in den erstellten Daten.

Die entzerrten 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 Winkel<sup>5</sup> zwischen Dartscheibe und Kamera, desto stärker ist die Verzerrung der Dartpfeile durch den Effekt der Normalisierung. Ebenfalls anzumerken ist das Abschneiden der Dartpfeilenden, sofern sich diese über die Ränder der Dartscheibe hinaus erstrecken.

## 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 24 576 Trainingsdaten, 256 Validierungsdaten und 2048 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-Texturen hinzukommen, die extern dynamisch in die Blender-Szene eingebunden werden.

## 2.4.3 Qualitative Auswertung: Subjektiver Unterschied zwischen generierten und echten Bildern

Da eine quantitative Auswertung im Bezug auf das Erscheinungsbild der gerenderten Bilder schwer umsetzbar ist, wird stattdessen eine qualitative Betrachtung der Daten durchgeführt. Dabei wurde auf Merkmale geachtet, die eine Unterscheidung echter 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 wurden und keine realen Dartscheiben abbilden. Ein zentraler Punkt dieser Beobachtung ist die Umgebung der Dartscheibe. Es wurde sich bewusst gegen die Einbindung eines statischen Hintergrundes entschieden, um die unmittelbare Umgebung der Dartscheibe variabel zu halten. Diese Design-Entscheidung schlägt sich hingegen darin nieder, dass die Dartscheiben in Situationen dargestellt sind, die in realen Aufnahmen nicht existieren. Dartscheiben schwanken durch die Verwendung Environment-Texturen in unüblichen Szenarien im Raum ohne sichtliche Befestigung. Für den menschlichen Betrachter ist dieser Umstand sehr auffällig und sorgt für eine direkte Identifikation als gerendertes Bild. Für den Einsatzbereich des Trainings und der Entzerrung ist dieser Aspekt jedoch nicht von Relevanz, da die Semantik eines möglicherweise fehlenden thematischen Kontexts um die Dartscheibe außerhalb des Interessenbereichs der algorithmischen Normalisierung sowie des Trainings eines neuronalen Netzes liegt. Für diese Systeme ist lediglich die Existenz variierender Informationen relevant, die strukturelle Ähnlichkeiten zu realen Daten aufweist. In Bildern, in denen unmittelbare Hintergründe der Dartscheibe beispielsweise durch Einblendung eines Dartschranks 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 Dartschranks 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

---

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

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 echten Aufnahmen abdecken. Die Verteilung der Dartpfeile sowie ihre Orientierung bei Eintreffen auf der Dartscheibe wirken ebenfalls realistisch und kohärent zueinander.

#### 2.4.4 Quantitative Metadatenauswertung

Eine quantitative Auswertung der Daten hinsichtlich ihres Erscheinungsbilds wurde aus bereits genannten Gründen nicht vollzogen. Jedoch existieren vielerlei Metadaten, die einer Auswertung unterzogen wurden. Dieser Unterabschnitt liefert einen Einblick in die Aufstellung der generierten Daten und die Resultate der in Abschnitt 2.2 beschriebenen Techniken. In Unterabschnitt 2.4.4.1 wird Einblick in ausgewählte intrinsische und extrinsische Parameter der Kamera gegeben, gefolgt von einer Übersicht über die Beleuchtungen und den Objekten, die um die Dartscheibe platziert werden können, sowie die Anzahl der Dartpfeile je Bild in Unterabschnitt 2.4.4.2. Zuletzt wird ein Blick auf die Verteilung der Dartpfeile über die Dartscheibe geworfen, indem die getroffenen Felder in Unterabschnitt 2.4.4.3 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. Für die Berechnung der Winkel wurde die Normale der Dartscheibe, die wie die Dartpfeile senkrecht auf der Dartscheibe steht, betrachtet, invertiert und ihr Winkel zur z-Achse der Kamera berechnet. Resultierend ist ein Winkel von  $0^\circ$  eine Frontalaufnahme der Dartscheibe während ein Winkel von  $90^\circ$  eine Aufnahme von der Seite der Dartscheibe darstellt. Durch die Berechnung sind ausschließlich positive Werte der Winkel möglich.

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

**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  $\sim 1\text{ m}$  und die maximale Distanz  $\sim 160\text{ cm}$ . Diese Verteilungen spiegeln nach subjektivem Empfinden Verteilungen wider, die in realen Szenarien zu erwarten sind. Geringere Aufnahmen sowie Aufnahmen mit größerer Entfernung als die verzeichneten wurden 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 sinngemäße Verarbeitung der Daten möglich 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  $\sim 10\text{ mm}$ , die maximale Brennweite  $\sim 60\text{ mm}$ . Die mittlere Brennweite der Daten beträgt  $\sim 38\text{ mm}$  und ist damit wertgleich zu dem oberen Brennweiten-Grenzwert minimaler Abstände sowie zu der unteren Grenze maximaler Abstände zur Dartscheibe.

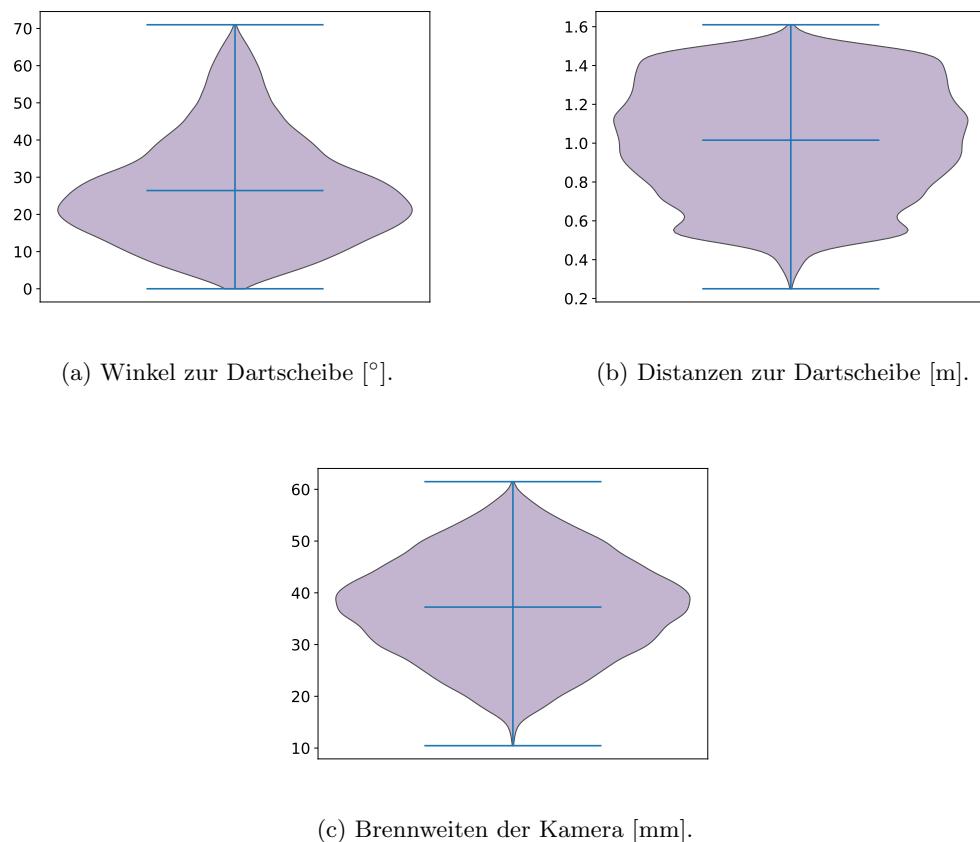
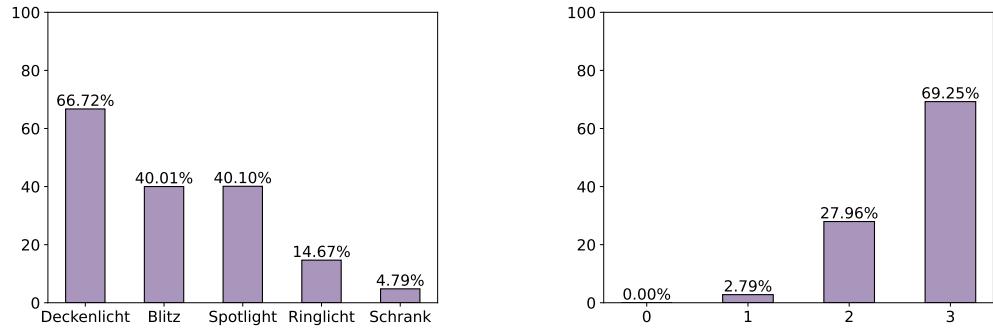


Abbildung 2.10: Verteilungen der Kameraparameter in den Trainingsdaten. Breite der Violinengraphen geben relative Häufigkeiten an, Mittel- und Extremwerte sind dunkelblau gekennzeichnet. Abbildung 2.10a Kamerawinkel. Abbildung 2.10b Kameradistanzen. Abbildung 2.10c Kamerabrennweiten.



(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.

#### 2.4.4.2 Beleuchtung und Objekte in der Szene

In der Szene werden Objekte zufällig ein- und ausgeblendet, basierend auf unterschiedlichen Wahrscheinlichkeitsverteilungen und Umgebungsbedingungen. Diese in den Daten aufgetretenen dynamischen Objekte sind in Abbildung 2.11a dargestellt. Hinsichtlich der Beleuchtungsobjekte ist das Deckenlicht mit  $\sim 67\%$  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  $\sim 40\%$  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 wurde daher gering gehalten. Mit  $\sim 15\%$  Vorkommen des Ringlichts und  $\sim 5\%$  Vorkommen des Dartschranks sind ihre Existenz von allen dynamischen Objekten am seltensten.

Neben den dynamischen Objekten wurde 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  $\sim 69\%$  der Daten sind alle 3 Dartpfeile vorhanden, in  $\sim 28\%$  sind 2 Dartpfeile auf der Dartscheibe verteilt und in  $\sim 3\%$  aller Trainingsdaten ist lediglich 1 Dartpfeil vorhanden. Es wurden 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 wurde eine Heatmap verwendet, die zu erwartenden, realistischen Verteilungen der Dartpfeile auf der Dartscheibe nachempfunden ist. Zusätzlich wurden spezifisch Daten erstellt, in denen die Pfeile ausschließlich auf die Bereiche der Double- und Triple-Felder sowie dem Bull fokussiert sind. In Abbildung 2.12 ist eine Auswertung der getroffenen Felder aller Pfeile in den Trainingsdaten visualisiert. Die Grafik teilt sich auf in die Felder 1-20 (links) und Bull und Fehlwürfe (rechts). Die Verteilung der Felder 1-20 entsprechen einer zu erwartenden Verteilung, in der Felder geringer Werte seltener getroffen worden als Felder mit hohen Zahlenwerten. Die Felder 20, 14 und 19 wurden am häufigsten getroffen, was typische Spielweisen widerspiegelt<sup>6</sup>.

Das Bull ist in den Daten verhältnismäßig gering vertreten in statistisch einem in 200

<sup>6</sup>Das Bespielen der 14 zieht bei Verfehlten höhere Zahlenwerte mit sich als das Verfehlten von 19 oder 20. Daher präferieren einige Spieler die 14 über 19 oder 20.

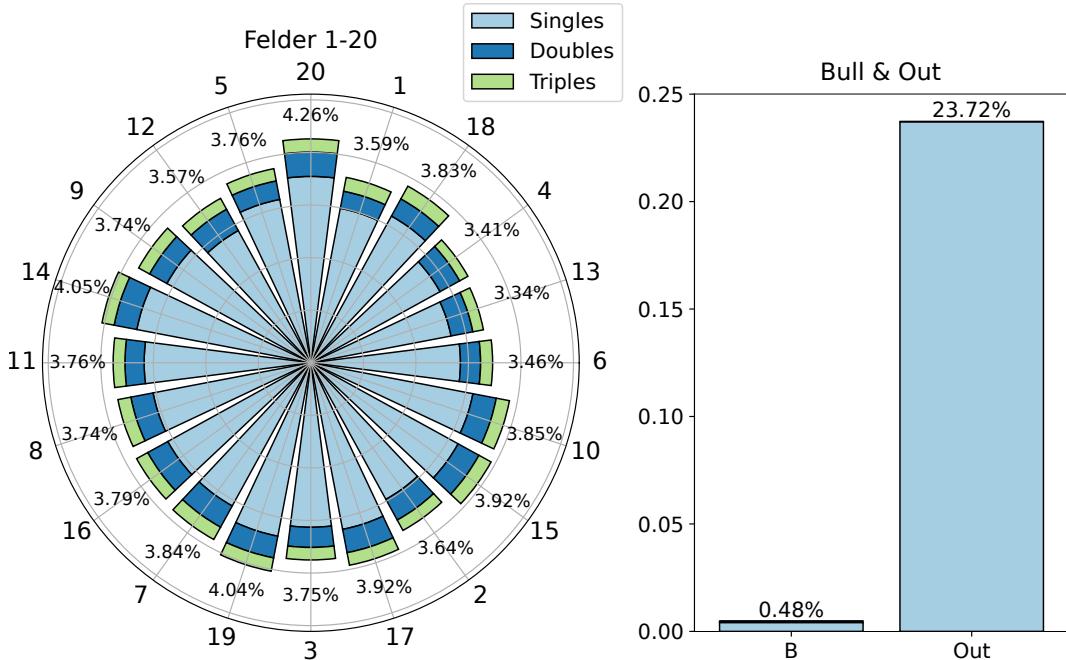


Abbildung 2.12: Relative Verteilung der Dartpfeile je Feld. Links: Felder 1-20; rechts: Bull und Outs.

Bildern. Übermäßig häufig hingegen ist das Verfehlten 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.

#### 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 siegt 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 beeinflusst<sup>7</sup>.

#### 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 und Nachverarbeitung 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

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

ist die Verwendung dieser zur Rückrechnung der Positionen im gerenderten Bild sehr komplex. Stattdessen werden Positionen durch Überschneidungen von Objekten und Rendering dieser als Binärbilder mit den selben Exportparametern 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 von Augmentierung überschattet (vgl. Unterabschnitt 4.2.3.4).

## 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 das DeepDarts-System wurde sich entschieden, die Normalisierung und Vorhersage der Dartpfeilpositionen in einem Schritt von dem neuronalen Netz zu übernehmen. Dieser Ansatz geht jedoch mit einigen Schwachpunkten einher. Wesentliche Nachteile wurden offensichtlich in Hinsicht auf den Verlust von Informationen durch Herunterskalierung des Bildes, um dem Input des neuronalen Netzes gerecht zu werden, und die Identifizierung spezifischer Fixpunkte, die möglicherweise verdeckt sein könnten. Darüber stützt sich eine Abwälzung eines Problems auf ein neuronales Netz auf die Korrektheit und den 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 mit großem Mehraufwand des weiteren Trainierens des Systems verbunden ist.

Eine algorithmische Herangehensweise an ein solches Problem 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 vorkommt und diese Problemquellen können gezielt angegangen werden. Darüber hinaus kann eine algorithmische Normalisierung von Daten ohne den Mehraufwand neuen Trainings eines neuronalen Netzes auf neue Daten erweitert werden. Dazu sind darüber hinaus lediglich wenige Beispiele neuer Daten notwendig, um die Arbeitsweise des Systems anzupassen.

Aus diesen Gründen wurde 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.

Ein Überblick über die Schritte des Algorithmus ist in Abbildung 3.1 dargestellt.

### 3.1 Grundlagen

Bevor in the 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 wichtig sind. Diese Grundlagen setzen mathematische Kenntnisse voraus, wie sie in einem Studium der Informatik erlernt und verstanden werden. Spezifische Algorithmen und Techniken der herkömmlichen

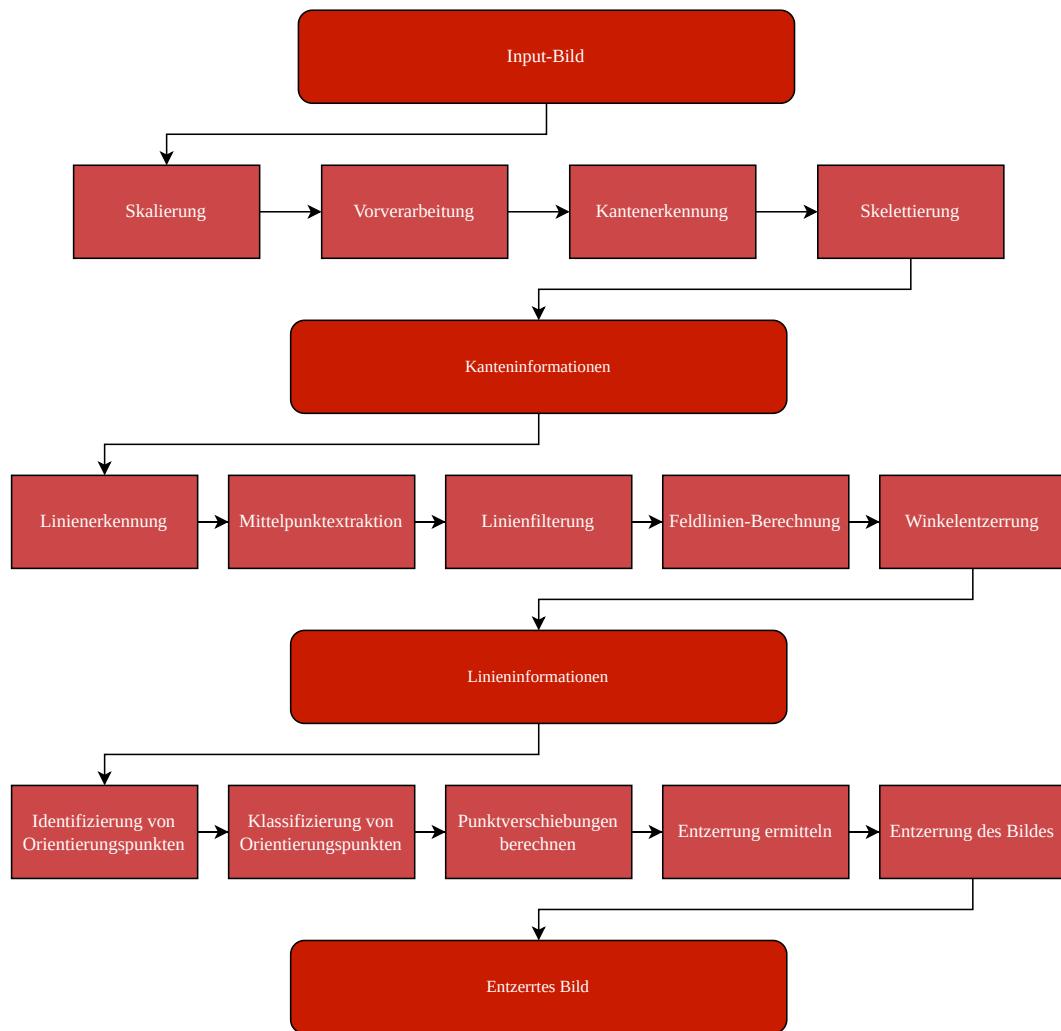


Abbildung 3.1: CV-Pipeline

Computer Vision werden grundlegend erläutert, jedoch nicht in einem Detailgrad, der zum vollständigen Durchdringen der Themen notwendig ist. Diese Sektion ist lediglich dafür vorgesehen, ausreichend Kontext zu den jeweils verwendeten Methoden und Algorithmen zu liefern, um die Anwendung und Arbeitsweise des Vorgehens zu verstehen.

### 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  $(\rho, \theta)$ , in dem  $\rho$  der minimale Abstand der Linie zum Koordinatenursprung ist und  $\theta$  der Winkel der Liniennormalen zur x-Achse. Die Charakterisierung einer Linie durch einen Winkel in Unterabschnitt 3.2.4 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 [40]:

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

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

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 beispielsweise das Identifizieren von Merkmalen anhand von Farben, die in bestimmten Farbräumen 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 Kante und Hintergrund oder in Unterabschnitt 3.2.4.3 zur Filterung von Linien anhand einer Abstandsmetrik.

### 3.1.3 Binning

Als Erweiterung von Thresholding lässt sich Binning verstehen. Unter Binning versteht man die Diskretisierung von Werten in definierte Intervalle, sogenannte Bins oder Buckets. Binning ermöglicht es, 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  $v_1 = v_0 - \epsilon_{>0}$  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 eingesortiert werden, gewichtet mit der Distanz zu der Intervallgrenze.

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

1	0	-1
1	0	-1
1	0	-1

(a) Horizontaler Kantenfilter

1	2	1
2	4	2
1	2	1

(b) Gaußscher Glättungsfilter

1	0	-1
2	0	-2
1	0	-1

(c) Horizontaler Sobel-Filter

Abbildung 3.2: Unterschiedliche 2D-Kernel der Größe  $3 \times 3$ .

### 3.1.4 Faltung

Die Faltung, auch Convolution genannt, ist eine mathematische Operation, die ihren Ursprung in der Signalverarbeitung hat. Die Faltung nimmt zwei Funktionen  $f$  und  $g$  und berechnet eine resultierende Funktion  $h = (f * g)$ . Die Definition einer Faltung ist [41, 37]:

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

In Hinsicht auf 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 [42]:

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

Dabei ist  $I \in \mathbb{N}^2$  ein Eingabebild mit einem Kanal,  $x$  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 unter mehrerer Kanäle anwenden. Für eine Faltung eines Bildes mit  $c_0$  Kanälen, einer Kernel-Größe von  $k_x \times k_y$  und einer Ausgabe in  $c_1$  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

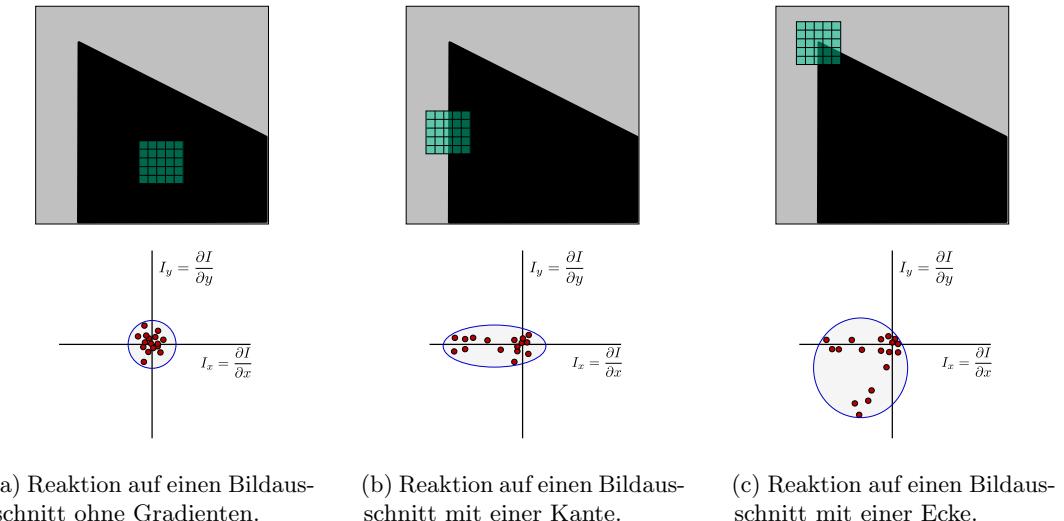
Nachdem die Prinzipien der Faltung bekannt sind, wird in diesem Abschnitt auf die konkrete Anwendung der Faltung in der Bildverarbeitung eingegangen.

Anhängig von den Werten eines Kernels werden unterschiedliche Charakteristiken eines Bildes hervorgehoben. So können mit einem Kantenerkennungsfilter hochfrequente Bestandteile in einem Bild hervorgehoben werden während ein Bild mit einem Glättungsfilter weichgezeichnet wird und hochfrequente Anteile herausgefiltert werden. Der Sobel-Filter ist ein in der Computer Vision etablierter Filter zur robusten Kantenerkennung[43, 37]. Der Sobel-Filter kombiniert die Prinzipien der Glättung und Kantenerkennung, indem das Bild 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 3.2 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 generell in Kapitel 4 als wichtiger Bestandteil von Schichten neuronaler Netze.

### 3.1.6 Harris Corner Detection

Die Harris Corner Detection ist ein Algorithmus, der 1988 von C. Harris und M. Stephens vorgestellt wurde, und ist ein etablierter Algorithmus in der Computer Vision [44, 37]. Ziel des Algorithmus ist es, Ecken in einem Eingabebild zu identifizieren und auszugeben. In dieser Sektion wird auf die grundlegende Arbeitsweise des Algorithmus eingegangen, um ein Verständnis seiner Arbeitsweise zu erlangen.



(a) Reaktion auf einen Bildausschnitt ohne Gradienten.  
(b) Reaktion auf einen Bildausschnitt mit einer Kante.  
(c) Reaktion auf einen Bildausschnitt mit einer Ecke.

Abbildung 3.3: Visualisierung der Reaktionen von Kantenfiltern auf unterschiedliche Bildstrukturen und die Repräsentation ihrer Gradienten [45]. Die Haupt- und Nebenachsen sind in Abbildung 3.3a jeweils kurz, in Abbildung 3.3b ist die Hauptachse lang während die Nebenachse kurz ist und in Abbildung 3.3b sind sowohl Haupt- als auch Nebenachse lang.

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. Diese werden in ein Koordinatensystem übertragen, in dem die Magnitude der Reaktion festgehalten wird. Die Richtungen und Ausprägungen der Kanten bestimmen die Positionen in dem Koordinatensystem. Cluster an Punkten in diesem Koordinatensystem deuten auf eine Häufung ähnlicher Kanten hin.

Bei einem Bildausschnitt ohne Kanten ist ein Cluster um den Ursprung zu erwarten, bei einem Bildausschnitt mit einer Kante ist ein weiterer Cluster zu erwarten. Sind mehrere Kanten in einem Bildausschnitt vorhanden, existieren mehrere Cluster, bei dem jeder Cluster durch eine Kante in einer bestimmten Richtung hervorgerufen wird. In dem Fall einer Ecke sind mehrere Cluster zu erwarten, die es zu identifizieren gilt. Die Identifikation basiert auf Flächenmomenten und dem Einpassen einer Ellipse an Daten. Die Details dieser Implementierung sind für einen Überblick nicht relevant; was in diesem Schritt wichtig ist, ist die Form und Größe der Ellipse, spezifisch ihre Hauptachse  $\lambda_1$  und Nebenachse  $\lambda_2$ . Existieren keine Kanten in dem betrachteten Bildausschnitt, ist ein zentraler Cluster mit geringer Größe im Koordinatenursprung zu erwarten,  $\lambda_1$  und  $\lambda_2$  sind klein. Bei einer Kante sind zwei Cluster zu erwarten, die von einer Ellipse mit langer Haupt- und kurzer Nebenachse umfasst werden:  $\lambda_1 \gg \lambda_2$  bzw.  $\lambda_2 \gg \lambda_1$ . Sind mehrere Kanten in unterschiedlichen Winkeln vorhanden, existieren mehrere Cluster, die sich durch eine große Ellipse umfassen lassen. Die Haupt- und Nebenachsen sind daher groß. Kantenreaktionen und resultierende Ellipsen sind in Abbildung 3.3 visualisiert.

$\lambda_1$  und  $\lambda_2$  werden mit der folgenden Formel zu einem Wert  $R$  zusammengefasst:

$$R = \lambda_1 \lambda_2 - \kappa_{[0.04, 0.06]} (\lambda_1 + \lambda_2)^2$$

$\kappa$  ist dabei ein empirisch bestimmter Wert zur Regulierung der Funktion.  $R$  wird in Kombination mit Thresholding genutzt, um einen Punkt in einem Bild als Ecke zu klassifizieren. Bei der Klassifizierung von Bildpixeln kommt es dazu, dass Cluster von Pixeln als Ecken erkannt werden. Aus diesen Clustern wird mittels Non-Maximum-Suppression derjenige Pixel hervorgehoben, der die größte Antwort auf die Eckenerkennung liefert. 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 [46, 46]. In diesem Abschnitt wird die Erkennung von Linien mit der Hough-Transformation beschrieben, wie sie in Unterabschnitt 3.2.4.1 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  $\rho$  und  $\theta$ . Linien im Image-Space sind mathematisch beschrieben als Geraden der polaren Form:

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

Dabei sind  $\rho$  und  $\theta$  Winkel und Abstand einer Linie zum Koordinatenursprung, wie in Unterabschnitt 3.1.1 eingeführt. Ein Punkt  $(x, y)$  im Image-Space ist im Parameter-Space als Sinuswelle dargestellt, die alle Linien beschreibt, die durch den Punkt  $(x, y)$  im Image-Space verlaufen. Ein Punkt  $(\rho, \theta)$  im Parameter-Space beschreibt eine Linie im Image-Space.

Input der Hough-Transformation ist ein durch Thresholding vorverarbeitetes Bild, in dem typischerweise Kanten oder Ecken extrahiert wurden. Die extrahierten Pixel werden im Parameter-Space akkumuliert, in dem sie sich in Form von Sinuswellen überschneiden. Liegen demnach mehrere Punkte im Image-Space in einer Linie, so überschneiden sich ihre korrespondierenden Sinus-Darstellungen im Parameter-Space in einem Punkt. Dieser Punkt liegt an den Koordinaten  $(\rho, \theta)$  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 Computer Vision basieren auf der Grundlage von Transformationsmatrizen [47, 48, 37].

#### 3.1.8.1 Homogene Koordinaten

Punkte im 2D-Raum besitzen eine  $x$ - und eine  $y$ -Koordinate, sodass  $P = (x, y)$ . 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. 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 3-dimensionale Einbettung des 2-dimensionalen Raumes und ermöglichen Transformationen von Koordinaten und als Erweiterung dessen auch die Transformation von Bildern durch Transformationsmatrizen  $M$ . Diese sind  $3 \times 3$  Matrizen, die durch Multiplikation mit homogenen Punkten Transformationen auf diesen anwenden:

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

Die unterschiedlichen Einträge der Transformationsmatrizen bestimmen die Art der Transformation. Durch Aneinanderreihung mehrerer Transformationsmatrizen aneinander 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:

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

### 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 [37]. Es gibt unterschiedliche Arten affiner Transformationen, die in diesem Unterabschnitt vorgestellt werden.

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

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

**Skalierung** Die Skalierungsmaatrix zeichnet sich durch ihre horizontale Skalierung  $c_x$  und vertikale Skalierung  $c_y$  aus und ist wie folgt aufgebaut:

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

**Scherung** Bei der Scherung werden Punkte parallel zur  $x$ -Achse mit  $c_x$  und parallel zur  $y$ -Achse mit  $c_y$  gescherzt:

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

**Rotation** Die Rotation erfolgt anhand eines Winkels  $\alpha$ , der Punkte in mathematisch positiver Richtung um den Koordinatenursprung rotiert. Sie zugehörigen Rotationsmatrizen nutzen diesen Winkel zur Konstruktion einer Rotationsmatrix. Anzumerken ist dabei, dass eine Rotationsmatrix als Kombination aus Skalierungen  $s_x = s_y = \cos(\alpha)$  und Scherungen  $-c_x = c_y = \sin(\alpha)$  konstruiert werden kann:

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

### 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 2 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 genutzt, um Orientierungspunkte der Eingabebilder auf bekannte Positionen in normalisierten und entzerrten Bildern zu mappen.

### 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  $(\rho, \theta)$  beschrieben. Sie sind definiert als logarithmischer Abstand und Winkel zu einem spezifischen Punkt  $(c_x, c_y)$  [49]. 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 [50, 37]. Zu den am weitesten verbreiteten Farbräumen zählen unter anderem der RGB-Farbraum und der HSV-Farbraum. Im RGB-Farbraum werden Farbinformationen analog zum menschlichen Auge 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 geschieht in Unterabschnitt 3.2.5.2.

### 3.1.11 Structural Similarity (SSIM)

Für die Aufgabe der Ähnlichkeitsbestimmung von Bildern wurde 2004 von Wang u. a. eine Metrik entwickelt, die über das triviale Vergleichen von Farbinformationen hinaus geht [51, 37]. 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 visuell geringer Änderungen, die jedoch starke Auswirkungen auf die zugrundeliegenden Daten besitzen. Ein Beispiel ist das Weichzeichnen eines Bildes oder das Hinzufügen von Rauschen. Diese Operationen ändern die Pixelwerte der Eingabebilder stark, jedoch sind sie visuell unauffällig.

### 3.1.12 RANSAC

Nicht-ideale Daten beinhalten häufig Outlier. Outlier sind Datenpunkte, die nicht der zu erwartenden oder gewünschten Datenstruktur folgen, sondern als fehlerhafte Aufnahmen in den Daten vorhanden sind. Triviale Least-Squares-Approximationen 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 [52, 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 der Approximation 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 genutzt, um eine Entzerrung der Dartscheibe auf Grundlage von Orientierungspunkten 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 Computer Vision 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. Darauf folgt in Unterabschnitt 3.2.5 der Schritt der Orientierung, 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.

### 3.2.1 Warum Computer Vision?

Normalisierung von Daten für ein neuronales Netz kann auf unterschiedliche Arten umgesetzt werden. Bei der Herangehensweise von McNally u. a. für DeepDarts werden Normalisierung von Dartscheibe und Lokalisierung von Dartpfeilen in einem einzigen Durchlauf von einem neuronalen Netz ausgeführt. Dazu werden Orientierungspunkte auf der Dartscheibe identifiziert, deren Positionen in der entzerrten Darstellung der Dartscheibe bekannt sind. Auf diese Weise kann eine Homographie zur Normalisierung abgeleitet werden. In dieser Arbeit wird auf herkömmliche Techniken der Computer Vision zurückgegriffen, um etwaige Nachteile eines neuronalen Netzes gezielt anzugehen. Der wichtigste Aspekt der algorithmischen Normalisierung ist die Nachvollziehbarkeit der Arbeitsweise und Wartung bzw. Anpassung des Systems an neue Gegebenheiten. Wohingegen ein neuronales Netz eine Black-Box ist, deren Arbeitsweise nicht bekannt ist, und die lediglich durch aufwändiges Training neu

eingestellt werden kann, kann bei einem Algorithmus nachvollzogen werden, wo er scheitert und er kann gezielt erweitert oder adaptiert werden.

Ebenfalls war die Verwendung von Computer Vision aufgrund der auffälligen Geometrie einer Dartscheibe naheliegend, da sie Ähnlichkeiten mit Schachbrettern aufweist, welche in der Computer Vision zur Identifizierung von Kameraparametern verwendet werden [37]. Da die Nutzung bekannter Geometrien eine zentrale Arbeitsweise der Computer Vision 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 wurde der in diesem Abschnitt beschriebene Algorithmus entwickelt.

### 3.2.2 Vorverarbeitung

Die Algorithmen der Computer Vision 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, was für eine Abwägung zwischen Geschwindigkeit und Genauigkeit sorgt. In dieser Arbeit wurde sich für eine schrittweise Verkleinerung der Eingabebilder mit Abmessungen  $(w, h)$ , entsprechend Breite und Höhe, entschieden, bis  $\max(w, h) < s_{\max} = 1600\text{px}$ . Dabei werden Eingabebilder jeweils um den Faktor 2 verkleinert, um Artefakte durch Interpolation zu minimieren. Der Wert von  $s_{\max}$  wurde 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 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 sehr viel Rauschen, das nicht für die Normalisierung benötigt wird. Mit der Kantenverarbeitung wird der Umfang an Informationen stark reduziert auf die wichtigen Charakteristiken des Bildes.

#### 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 wurden. 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 Bereiche im Bild voneinander trennen, in denen zudem wenig Kanten erwartet werden. Auf Grundlage dieser Beobachtungen wurde sich für einen untypisch großen Sobel-Kernel mit einer Größe von  $15 \times 15$  Pixeln entschieden, dargestellt in Abbildung 3.4. Dieser Kernel sorgt für eine gezielte Erkennung der geschriebenen Eigenschaften in Bildern.

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 dieses Bild wird der beschriebene Sobel-Kernel in vertikaler und horizontaler Richtung angewendet, um Filterreaktionen von Intensitätsänderungen entlang beider Richtungen zu erlangen. Diese werden miteinander kombiniert und durch Thresholding

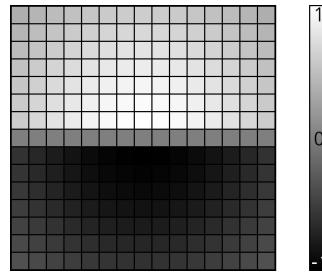


Abbildung 3.4: Vertikaler Sobel-Kernel der Größe  $15 \times 15$  zur Identifizierung großer und unifomer Kanten in einem Bild. Helle Pixel stehen für positive, dunkle Pixel für negative Werte.

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 mehrere Pixel breite Kanten. Diese breiten Kanten werden mittels Skelettierung auf ihre zentrale Kante reduziert [53]. Bei der Skelettierung werden die existierenden Kanten iterativ verringert, bis eine zentrale Kante erzielt wurde. Dazu wird das Konzept der Erosion verwendet, bei Cluster von Pixeln in Binärbildern entlang ihrer Kontur verkleinert werden. Bildlich lässt sich das veranschaulichen mit einer in Wasser liegenden Insel, die durch Erosion an Höhe über dem Meeresspiegel verliert und sich dadurch von außen nach innen verkleinert. Nach der Skelettierung des Kantenbildes verbleibt eine minimale Darstellung der extrahierten Kanten, in der diese auf ihre wesentlichen Züge heruntergebrochen wurden. Der verbliebene Informationsgehalt des Bildes wurde dadurch auf das für die kommenden Schritte wesentliche reduziert.

Der Prozess der Kantenerkennung ist in Abbildung 3.5 dargestellt. Das verwendete Bild stammt aus dem für DeepDarts verwendeten Datensatz und wurde ebenfalls im Paper des Systems zur Veranschaulichung von dessen Arbeitsweise genutzt. Im Sinne der Vergleichbarkeit der Systeme wurde sich daher dazu entschieden, die Arbeitsweise dieses Algorithmus anhand des selben Bildes zu veranschaulichen.

### 3.2.4 Linienverarbeitung

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 Linienverarbeitung 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 Linienverarbeitung 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 im ersten Schritt Linien identifiziert 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:  $L_{\text{points}} = \{(p_{i,0}, p_{i,1}) \mid i \in [0, n_{\text{lines}} - 1]\}$ , wobei  $n_{\text{lines}}$  die Anzahl der gefundenen Liniensegmente ist. In Abbildung 3.6 (1) werden erkannte Linien anhand eines Beispielbildes dargestellt. Jeder Linie wurde zur Visualisierung eine zufällige Farbe zugeordnet. Zu erkennen ist, dass neben den zu erwartenden langen Linien auch viele sehr kurze Linien erkannt werden. Der Grund für eine Häufung vieler kurzer Linien liegt in der diskretisierten Darstellung von Pixeln und Ungenauigkeiten durch Verwackelungen, ungerade

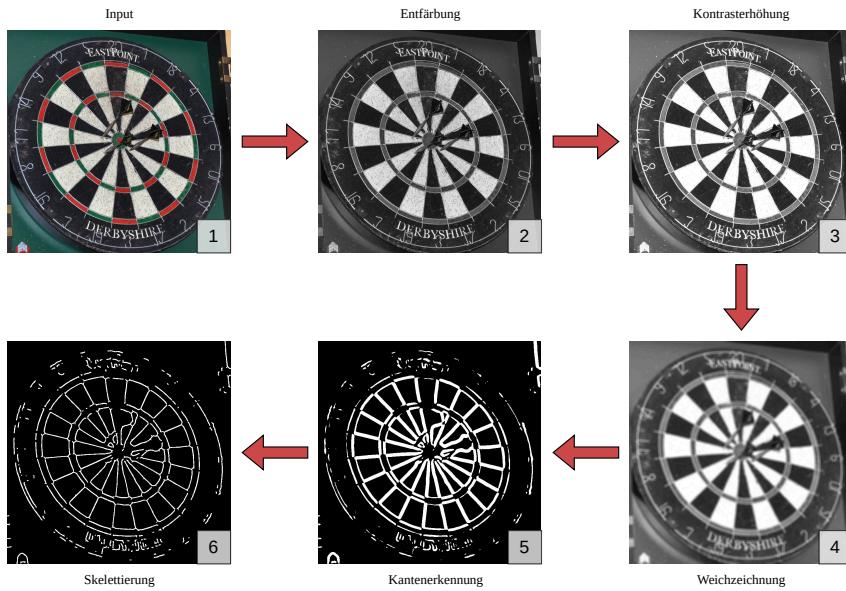


Abbildung 3.5: Schritte der Kantenverarbeitung. (1) Input-Bild aus dem DeepDarts-Datensatz [4]. (2) Umwandlung des Bildes in Graustufen. (3) Kontrasterhöhung des Bildes zur Hervorhebung der Unterschiede schwarzer und weißer Felder. (4) Weichzeichnung zur Verminderung von Störungen. (5) Filterung durch Sobel-Filter, gefolgt von Thresholding. (6) Skelettiertes Kantenbild.

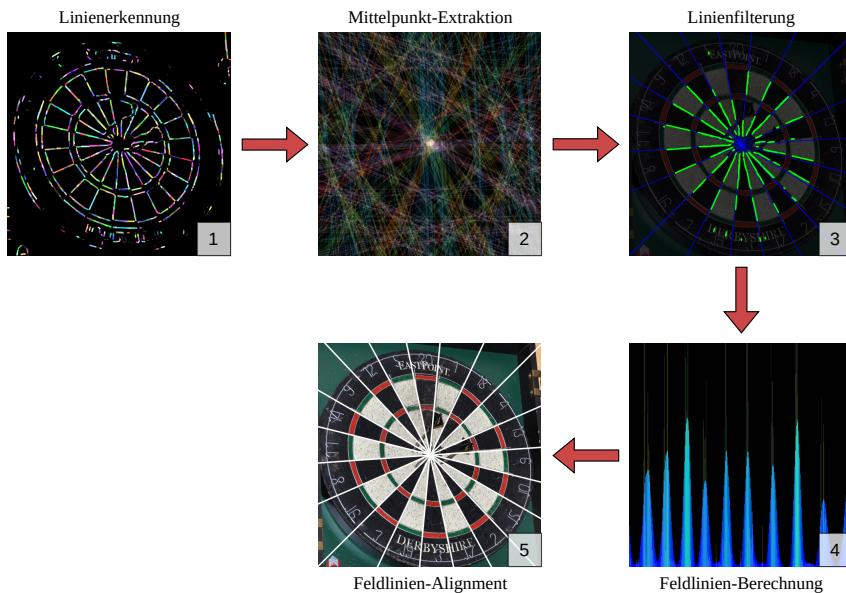


Abbildung 3.6: Veranschaulichung der Schritte der Linienverarbeitung. (1) Identifizierung von Linien im Kantenbild. Jede Linie ist zur Visualisierung in einer zufälligen Farbe dargestellt. (2) Extraktion des Mittelpunktes anhand unterschiedlicher Linienwinkel. Jede Klasse von Winkelwinkeln ist in einer zufälligen Farbe dargestellt. (3) Filterung der Linien anhand des Mittelpunktes. Verbleibende Linien sind grün hervorgehoben; Feldlinienwinkel  $\phi_i$  in blau. (4) Akkumulation der Winkel von Pixeln in gefilterten Linien. Braune Balken sind ungefilterte, blaue Balken gefilterte Werte. (5) Entzerrte Feldlinien. Alle Winkel  $\phi_i = 18^\circ$  sind weiß hervorgehoben.

Feldgrenzen oder Verzerrungen der Kameralinse. Bei dem Prozess der Linienerkennung kann jedenfalls nicht davon ausgegangen werden, dass Linien exakt erkannt werden. Trotz dessen tragen zu kurze Linien mit hoher Wahrscheinlichkeit wenig relevante Informationen, sodass Linien, die kürzer als 5 Pixel sind, 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 der Länge der Linie enthalten. Dieser Aspekt wird in dem kommenden Unterabschnitt zum Vorteil genutzt.

### 3.2.4.2 Mittelpunktextraktion

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

Unter Berücksichtigung dieser Beobachtung geschieht ein Binning der Linien  $L_{\text{polar}}$  anhand ihrer Winkel  $\theta_i$  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  $P_{\text{max}} = (x_{\text{max}}, y_{\text{max}})$  mit dem höchsten Wert dadurch aus, dass durch ihn die meisten Linien verschiedener Richtungen verlaufen. Diese Eigenschaft ist durch die Art der Filterung 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  $P_{\text{max}}$  der Mittelpunkt der Dartscheibe  $m_{\text{Dart}} = (m_x, m_y)$  identifiziert wurde.

Hinsichtlich der Robustheit dieses Algorithmus ist der Fall hervorzuheben, dass Feldlinien durch u. a. perspektivische Verzerrungen oder fehlerhafte Kanten- und Linienerkennung möglicherweise von den zu erwartenden Winkelintervallen abweichen können und nicht in den ihnen zugewiesenen Bins eingeordnet werden. Es kann dadurch zur Einordnung mehrerer Feldlinien in gleiche Bins und folglich 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 wurden 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 der Linien  $L_{\text{points}}$  und  $L_{\text{polar}}$  umfassen neben den für die Entzerrung relevanten Feldlinien weitere Linien, die nicht relevant für die Geometrie der Dartscheibe in dem Bild 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 ist definiert durch [54]:

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

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:

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

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

Auf diese Weise werden diejenigen Linien  $\tilde{L}_{\text{polar}}$  und  $\tilde{L}_{\text{points}}$  ermittelt, die auf den Mittelpunkt der Dartscheibe gerichtet sind und voraussichtlich 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-Brechnung

Zur Identifizierung der Winkel  $\phi_i$  der Feldlinien wird eine adaptierte Hough-Transformation auf die gefilterten Linien  $\tilde{L}_{\text{polar}}$  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  $5^\circ$  – entsprechend 10 Bins – geglättet. In dem resultierenden Akkumulator werden die 10 größten Peaks  $\phi_i$  durch Non-Maximum-Suppression 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 Lage der Feldlinien bekannt und entzerrt ist.

Um diese Entzerrung vorzunehmen, wird eine Minimierung vorgenommen, in der eine affine Transformation gesucht wird, die diese Winkel bestmöglich aneinander anpasst und auf einen Winkelabstand von  $18^\circ$  angleicht. Diese Optimierung beginnt bei einer Startlinie und entzerrt alle restlichen Linien iterativ und wird für jede der 10 Linien als Startlinie ausgeführt. Als finale Transformation wird der Mittelwert aller optimierten Transformationen verwendet. Im folgenden wird die allgemeine Transformationssequenz angegeben.

Die erste Teiltransformation ist die Translation des Mittelpunktes  $M$  in den Koordinatenursprung  $O = (0, 0)$ . Dieser Schritt ist relevant, da atomare affine Transformationen



Abbildung 3.7: Ellipsoide Dartscheibe trotz entzerrter Feldlinienwinkel.

um  $O$  zentriert sind. Darauf folgt die vertikale Ausrichtung der Startlinie  $L_s$  durch eine Rotation um  $-\phi_s$ , sodass  $\phi'_s = 0^\circ$  erzielt wird. Gefolgt wird diese Rotation von der horizontalen Ausrichtung der Orthogonalen  $L_o = L_{(i+5) \bmod 10}$  durch eine Scherung entlang der Vertikalen. Wichtig bei diesem Schritt ist, dass die vertikale Scherung den Winkel von  $\phi'_s$  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  $\phi'_s$  seinen Zielwinkel besitzt. Zuletzt wird eine Translation des Koordinatenursprungs auf  $M$  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  $\phi_i$  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 dabei, dass trotz Angleichung der Winkel  $\phi_i$  keine Normalisierung der Dartscheibe erreicht ist. Um die Normalisierung zu vollenden, muss die Dartscheibe von einer elliptischen in eine runde Form gebracht und korrekt skaliert werden. Diese Schritte geschehen in dem Verarbeitungsabschnitt der Orientierung.

### 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 Umsetzung des DeepDarts-Systems, in welchem Orientierungspunkte identifiziert werden, deren Positionen in einem ideal entzerrten Bild bekannt sind. Anhand dieser Punktverschiebungen kann folglich eine Homographie abgeleitet werden, die eine finale Transformation des Bildes vornimmt, um diese Punkte auf ihre Zielpositionen zu überführen und die Dartscheibe final zu entzerrn. Im Gegensatz zu einer handvoll fest definierter Orientierungspunkte, wie es im DeepDarts-System festgelegt wurde, werden für dieses System beliebig viele Orientierungspunkte identifiziert, wodurch

eine wesentlich robustere Entzerrung ermöglicht ist.

Die wesentlichen Schritte der Orientierung sind in Abbildung 3.8 dargestellt und werden in den folgenden Unterkapiteln genauer erläutert.

### 3.2.5.1 Identifizierung von Orientierungspunkten

Die konkreten Orientierungspunkte werden nach dem Vorbild des DeepDarts-Systems ausgewählt und befinden sich an Eckpunkten zwischen Dartfeldern. Diese Punkte sind besonders markant und durch ihre Position zwischen den Feldern klar zu identifizieren. Es werden die Kreuzpunkte auf der Innen- und Außenseite 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 die 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 dadurch 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 daher 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 verbliebenen Eckpunkt in dem log-polaren Bild der Dartscheibe wird eine Surrounding betrachtet. Als Surrounding wird die unmittelbare, quadratische Umgebung um einen potenziellen Orientierungspunkt bezeichnet. Durch sie ist eine Einordnung möglich, ob es sich bei einer Ecke um einen Orientierungspunkt handelt oder nicht. Für jede Surrounding wird eine Farbraumtransformation in einen Farbraum mit dem Namen CrYV. Dieser Farbraum setzt sich aus unterschiedlichen Farbräumen 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.

In dem CrYV-Farbraum werden die mittleren Farbwerte der Ecken aller Surroundings klassifiziert. Durch diese lässt sich einerseits herausfinden, ob eine Ecke ein Outlier ist, und andererseits, in welcher Orientierung sich ein Eckpunkt befindet. Dazu werden die Ecken in 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 herausgefiltert, die mit großer Wahrscheinlichkeit keine Orientierungspunkte sind.

In einem folgenden Schritt wird eine mittlere Surrounding aller verbliebenen Eckpunkte als Median aller normalisierter 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 in Kauf zu nehmen.

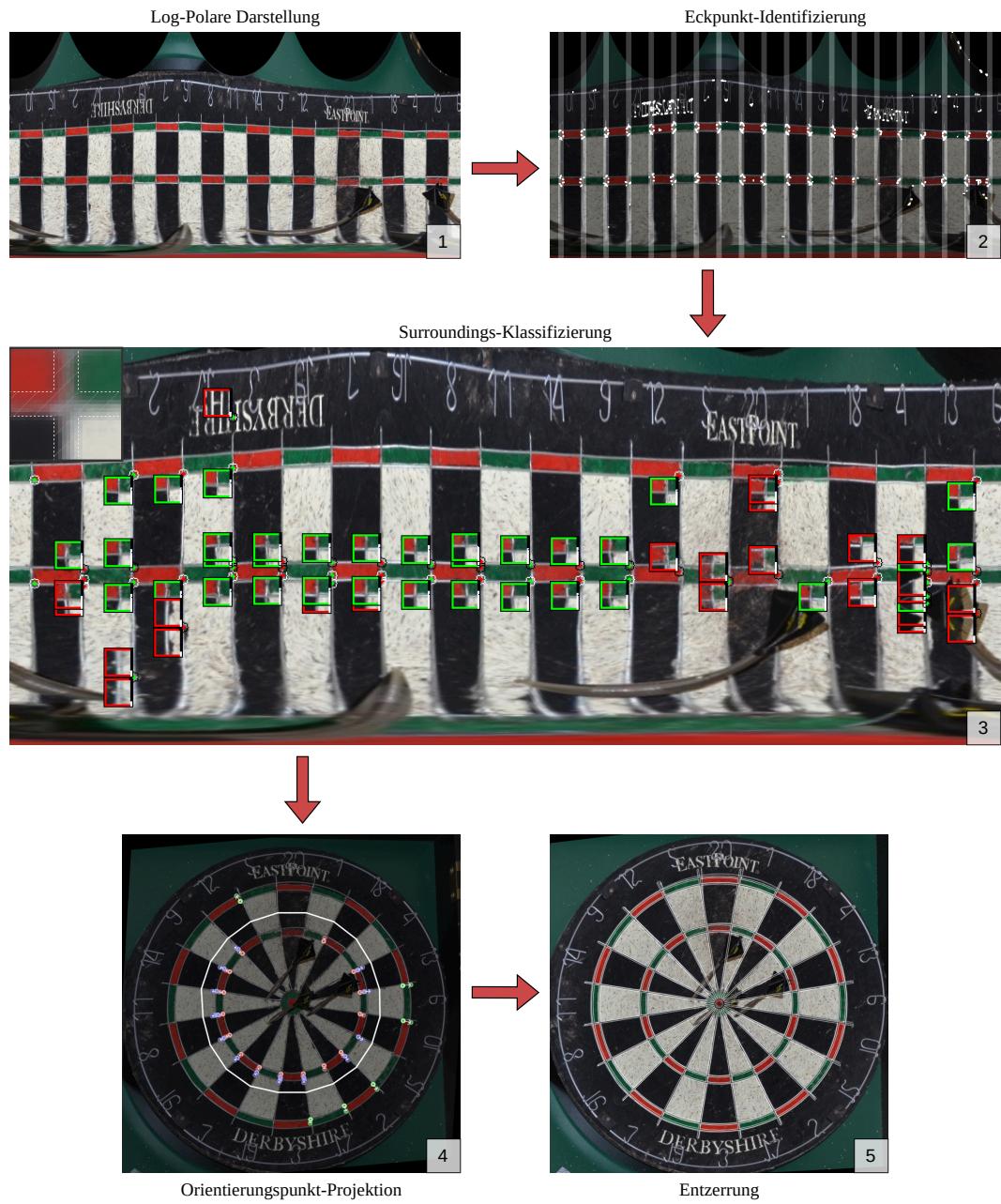


Abbildung 3.8: Schritte zur Orientierung der Dartscheibe. (1) Log-polare Darstellung der Dartscheibe. (2) Identifizierte Eckpunkte. Hervorgehobene Streifen veranschaulichen die erwarteten Bereiche der Feldlinien. (3) Identifizierung und Klassifizierung von Surroundings. Positiv klassifizierte Surroundings 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.

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 Punktverschiebungen berechnen

Durch die Position und die Beschaffenheit 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 Surrounding bekannt, ob er sich auf der Innen- oder Außenseite eines Rings befindet. Durch diese Informationen lässt sich für jeden 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 Interpolierung identifiziert.

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

### 3.2.5.4 Entzerrung

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 wurden oder die sich nicht an Positionen von Orientierungspunkten befinden. Bei der Ableitung einer Entzerrungshomographie 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 jedem Durchlauf werden zufällig 5 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.

## 3.2.6 Zusammenführen aller Komponenten

An diesem Punkt wurde die gesamte CV-Pipeline durchlaufen und es wurden mehrere Transformationsmatrizen für verschiedene Zwischenschritte der Normalisierung erstellt. 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. Mapping der Orientierungspunkte auf Ziel-Positionen nach Unterabschnitt 3.2.5 (Orientierung):  $M_{project}$

Die Aneinanderreihung dieser Transformationen führt zu der finalen Transformation  $M_{final}$ . Mathematisch setzt sich diese wie folgt zusammen:

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

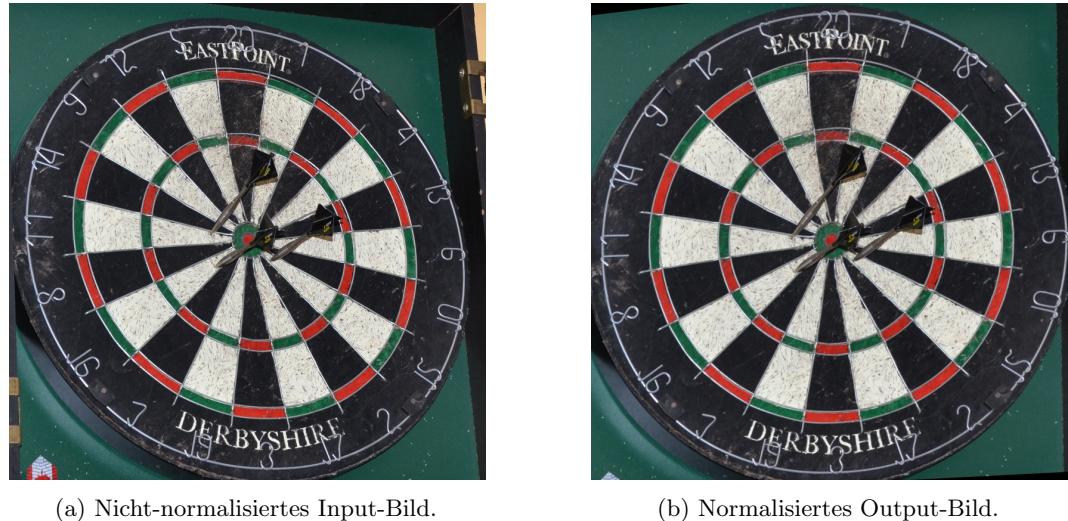


Abbildung 3.9: Entzerrung eines Beispielbildes aus dem DeepDarts-Datensatz [4]. Abbildung 3.9a zeigt das Input-Bild, Abbildung 3.9b zeigt das normalisierte Output-Bild nach der Verarbeitung durch die CV-Pipeline.

Nach der Anwendung dieser Transformation auf das Eingabebild wird ein finales Zuschneiden des Bildes auf die Eingabegröße des neuronalen Netzes vorgenommen, was die Normalisierung des Bildes abschließt.

Der Effekt der Entzerrung anhand des Beispielbildes ist in Abbildung 3.9 dargestellt.

### 3.3 Implementierung

CV-Implementierung beschreiben.

#### 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 verläuft.

Für die Bewältigung dieser Aufgabe wird eine Adaption der Hough-Transformation implementiert. 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  $\varphi_i$  zum Mittelpunkt bestimmt:

$$\varphi_i = \arctan2(y_i - m_{\text{Dart, y}}, x_i - m_{\text{Dart, x}})$$

Für die Implementierung dieser Berechnung wurde 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 kompilierter Funktionen ist eine schnelle Ausführung aufwändiger Berechnungen trotz der Ausführung mit Python möglich.

#### 3.3.2 Farben-Identifizierung

Für die Identifizierung von Orientierungspunkten werden die Farben der Umgebungen der Kandidaten der 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 auszumachen, ob eine gewisse Farbgebung vorhanden ist.

**Klassifizierung schwarzer und weißer Bereiche** Die durchschnittliche 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 (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:

$$\text{is\_bw}(C_p, C_r, T_C) = \begin{cases} 1, & \text{wenn } \sum_{i=0}^2 |C_r[i] - C_p[i]| < T_C \\ 0 & \text{sonst} \end{cases}$$

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:

$$\text{is\_color}(C_p, T_C, t_{\text{red}}, t_{\text{green}}) = \begin{cases} 1, & \text{wenn } \min(|t_{\text{red}} - C_p[0]|, |t_{\text{green}} - C_p[0]|) < T_C, \\ 0, & \text{sonst} \end{cases}$$

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

### 3.3.3 Klassifizierung von Surroundings

top/bottom, left/right: black/white/color

Kombination der Ecken -  $\downarrow$  Art der Surrounding (innen / außen von Ring)

Abgleich mit mittlerer Surrounding

Surroundings-  
Implementierung

## 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 stattgefunden hat. 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.

Datenquelle	Gerenderte Bilder	DeepDarts-d1 Validierung	DeepDarts-d1 Test	DeepDarts-d2 Validierung	DeepDarts-d2 Test
Anzahl Bilder	2048	1000	2000	70	150
Automatische Annotation	✓	✗	✗	✗	✗

Tabelle 3.1: Datenquellen für die Auswertung der Dartscheibenentzerrungen.

### 3.4.1 Metriken

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

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

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  $\Lambda$  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  $P_i \in [N_{\text{OP}}]$  aller Orientierungspunkte im Zielfeld 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:

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

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  $\tilde{H} = \hat{H}$ , da  $\hat{H} \times \text{inv}(\tilde{H}) = \text{Id}$ , die Identitätsmatrix.

### 3.4.2 Verwendete Daten

- Gen-Daten + DD-Daten - keine negativen Sample, da lediglich Ergebnisse auf positiven Daten relevant sind.
- es geht darum, Dartscheiben zu entzerren, nicht darum, sie zu identifizieren
- dass Dartscheiben in den Bildern vorhanden sind, wird als Voraussetzung für die Verwendung des Systems angesehen

Zur Evaluierung des Algorithmus werden Daten benötigt. Die Daten dieser Auswertung stammen aus 5 unterschiedlichen Quellen und werden 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, indem in jedem Bild eine Dartscheibe vorhanden ist. Aufgabe der Systems ist nicht die Identifizierung von Dartscheiben, sondern die Entzerrung dieser, sodass eine Existenz einer Dartscheibe in den Bildern vorausgesetzt wird.

### 3.4.3 Quantitative Auswertung

Die quantitative Auswertung ist unterteilt in die Abschnitte Geschwindigkeit, Render-Ergebnisse, DeepDarts-Ergebnisse und Zusammenfassung. Die Aufteilung der Ergebnisse in

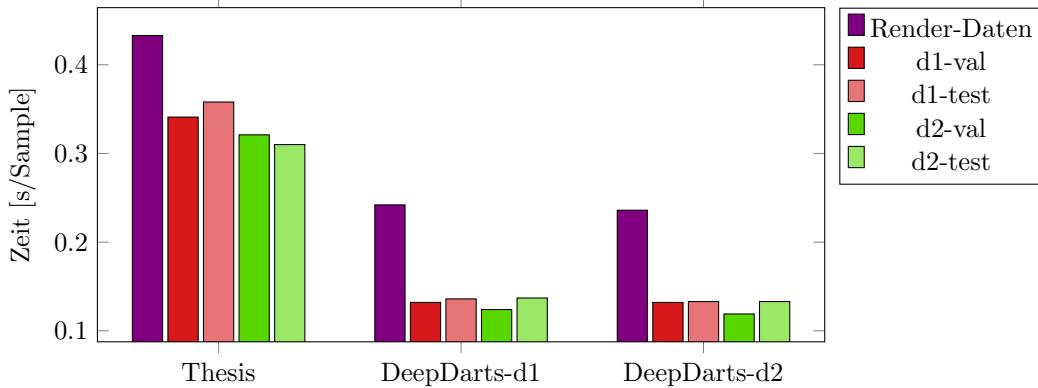


Abbildung 3.10: Dauer der Normalisierung auf unterschiedlichen Datensätzen, gruppiert nach Systemen.

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 Render-Daten wesentlich offener hinsichtlich der Darstellung der Dartscheiben. Diese Vorverarbeitung der DeepDarts-Daten soll zu keiner unvorhergesehenen Verzerrung der Daten führen.

Die Auswertungen sind jeweils unterteilt in das in dieser Thesis erarbeiteten System und das DeepDarts-System, um die Unterschiede der Genauigkeiten darzustellen. Da das DeepDarts-System ein Single-Shot-Neural-Network ist, 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 des DeepDarts-Systems 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 5 Durchläufe der Auswertung vorgenommen, deren Mittelwerte zur Evaluation verwendet wurden.

### 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 Zeiten des Systems dieser Thesis. Die Ausführungszeiten des Systems dieser Thesis 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 des DeepDarts-Ansatzes, 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  $800 \times 800$  px aufweisen. Die gerenderten Daten hingegen sind für diese Auswertung in keinerlei Weise vorverarbeitet und werden den Systemen in den originalen Auflösungen präsentiert. Die Seitenlängen von Bildern der Render-Daten betragen mindestens 434 px, maximal 3998 px und die durchschnittliche Seitenlänge beträgt 2147 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

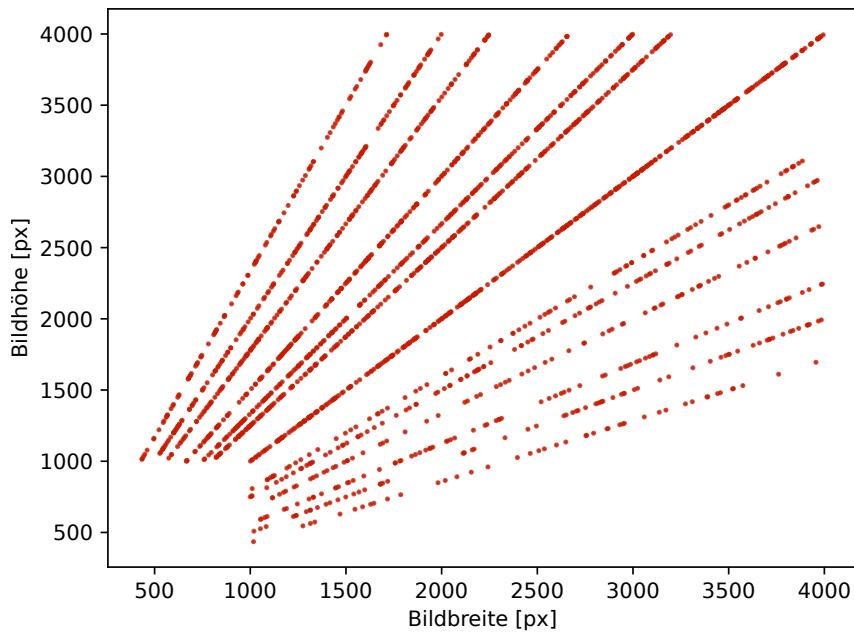


Abbildung 3.11: Verteilung der Seitenverhältnisse der gerenderten Bilder.

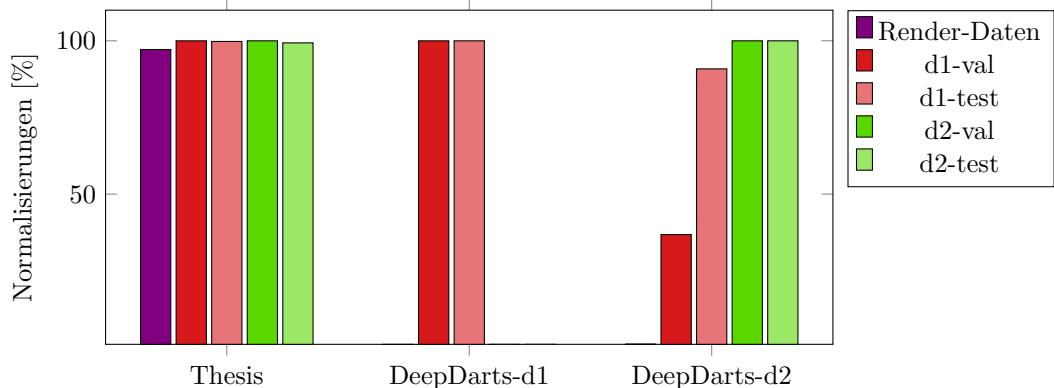


Abbildung 3.12: Auswertung der Fähigkeit unterschiedlicher Systeme, Normalisierungen für Daten zu finden.

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 und 1600 px skaliert werden und damit über den Abmessungen der DeepDarts-Daten liegen.

### 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 hier erarbeitete System 60 mögliche Punkte erkennen kann.

Die Ergebnisse dieser Auswertung sind in Abbildung 3.12 in Form von Kuchendiagrammen 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,

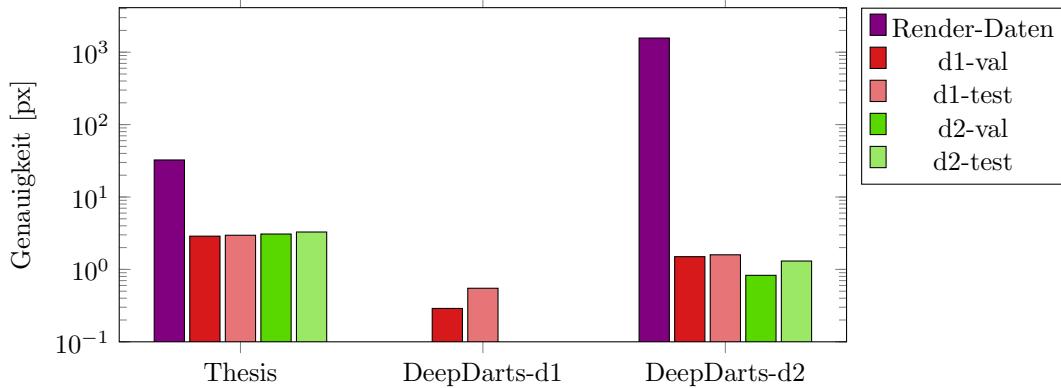


Abbildung 3.13: Genauigkeiten der Normalisierungen auf unterschiedlichen Datensätzen, gruppiert nach Systemen. Sofern keine Normalisierung möglich war, existiert kein Balken.

in mindestens 97% der Bilder eine Normalisierung zu ermitteln. Die Performance ist dabei weitestgehend unabhängig von dem Ursprung der Daten. Demgegenüber steht die Performance der DeepDarts-Systeme  $d_1$  und  $d_2$ . Während  $d_1$  zwar Auswertungen von 100% auf den eigenen Validierungs- und Test-Daten erzielt, ist es nicht in der Lage, positive Ergebnisse auf anderen Daten zu erzielen. Die Auswertung von  $d_2$  auf den eigenen Daten liegt ebenfalls bei 100% und zumindest die Findung der Entzerrung der Test-Daten aus  $d_1$  deckt sich mit 83.4% mit den Beobachtungen aus dem eigenen Paper. Jedoch ist die Auswertung auf den Validierungs-Daten von  $d_1$  mit 36.8% nicht annähernd auf diesem Niveau und auf den gerenderten Daten konnten lediglich für 2.2% der Daten normalisiert werden.

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 der in Unterabschnitt 3.4.1 eingeführten Metrik durchgeführt und die Ergebnisse sind in Abbildung 3.13 dargestellt. Es ist zu erkennen, dass signifikante Unterschiede zwischen den Systemen vorherrschen. DeepDarts' System  $d_2$  erzielt auf dem ihm zugewiesenen Validierungs- und Test-Daten ausgesprochen gute Ergebnisse mit durchschnittlich 0.419 px Abweichung während es auf anderen Daten keine Ergebnisse erzielen kann.  $d_2$  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 1.545 px Abweichung auf den  $d_1$ -Daten und 1.067 px auf den  $d_2$ -Daten erzielt werden. Weit davon abweichend ist jedoch die Auswertung auf den gerenderten Daten, auf denen lediglich eine mittlere Abweichung von 1568.744 px erzielt wurde. Diese Beobachtung lässt darauf schließen, dass eine zuverlässige Normalisierung nicht mit diesem System möglich war, da die Bilder lediglich eine Größe von  $800 \times 800$  px besaßen, was wesentlich geringer ist als die mittlere Abweichung.

Das in dieser Thesis erarbeitete System war jedoch in der Lage, auf allen Datensätzen Normalisierungen zu finden. Darüber hinaus bewegen sich die mittleren Verschiebungen über die Datensätze in etwa ähnlichen Wertebereichen: die Render-Daten konnten mit einer mittleren Verschiebung von 17.234 px normalisiert werden, die DeepDarts-Daten mit 3.281 px. Die höhere Genauigkeit auf den DeepDarts-Datensätzen stammt von der Vorverarbeitung der Daten, sodass diese eine feste Größe besitzen, bei der die Dartscheiben nicht während der Vorverarbeitung dieses Systems skaliert werden. 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 unter der Verwendung von Python-Code mit gelegentlicher Einbindung von Frameworks, die auf kompliziertem Maschinencode arbeiten, entwickelt wurde, verwendet DeepDarts ein neuronales Netz, welches nahezu vollständig kompiliert ist und keinerlei Kontrollfluss wie Verzweigungen und Schleifen verwendet. 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. DeepDarts ist nicht in der Lage, Bilder zu normalisieren, welche nicht aus den für das Training verwendeten Daten stammen. Die Wahl der Testdaten ist dabei zu hinterfragen, da diese für eine Vermeidung eines solchen Overfittings vorgesehen sind.



# Kapitel 4

## Identifizierung von Dartpfeilen mit neuronalen Netzen

Einleitende Sätze zu Kapitel 4.

### 4.1 Grundlagen

Zum Verständnis dieses Kapitels wird analog zu Kapitel 3 mit Grundlagen zu Konzepten und Begrifflichkeiten begonnen. Diese ermöglichen ein grundlegendes Verständnis, um die in diesem Kapitel eingesetzten Techniken zu verstehen 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 erläutert Unterabschnitt 4.1.5.

#### 4.1.1 Was sind neuronale Netze?

Neuronale Netze sind Kern eines spezifischen Bereichs des Machine Learnings, in dem sich auf das Erlernen von Eigenschaften 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 in Gehirnen 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 Kern neuronaler Netze war 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 auf Grundlage von Kerneln funktioniert. Auf dieser Arbeitsweise fußen die Convolutional-Schichten. Die Parameter dieser Schichten bestimmen die Ausprägung eines Kerbels, der auf eingehende Bilddaten angewandt wird. Durch vielfache Hintereinanderreihung von Convolutional-Schichten können inkrementell komplexere Strukturen in Bildern identifiziert und abstrakt festgehalten werden [55]. Neuronale Netze, die auf Convolutional-Schichten aufbauen, werden als Convolutional Neural Networks, kurz 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<sup>1</sup> [56, 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 Netzwerken geschieht für diese Arten der Fragen typischerweise in Form von Vektoren, die diese Kategorien durch One-Hot-Encoding (oder 1-of-n-Encoding) darstellen [57]. Dabei ist jeder Kategorie ein Eintrag im Vektor zugeordnet; die Größe der Zahlenwerte geben die Ausgaben des Netzes für die jeweiligen Kategorien an.

Konträr zur Klassifikation diskreter Gegebenheiten ist die Vorhersage kontinuierlicher Werte als Ausgabe eines neuronalen Netzes möglich, die 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. Der Hintergrund dieser Normalisierung liegt in der Arbeitsweise der Netzwerkschichten und wird üblicherweise in einem Nachverarbeitungsschritt nach der Vorhersage des neuronalen Netzes wieder umgekehrt.

In dieser Thesis werden sowohl binäre als 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 seinen Ausgaben auf unterschiedliche Arten verlaufen. In dieser Arbeit wird Supervised Learning verwendet, bei welchem Eingabedaten mit ihren zugehörigen Ausgaben gegeben sind. Das Netzwerk erlernt auf der Grundlage dieser Daten Parameter, durch die die Ausgaben zu den jeweiligen Eingaben ableiten. Weitere Methoden zum Training neuronaler Netze sind Unsupervised Learning und Reinforcement Learning. Bei Unsupervised Learning liegen lediglich Eingabedaten vor und die Ausgaben werden von dem Netzwerk identifiziert. Diese Art des Lernens wird beispielsweise bei Clustering von Datenpunkten oder Findung von Wort-Embeddings verwendet. Reinforcement Learning wird genutzt, um einem System das Agieren in einer Umgebung zu ermöglichen und basiert auf Belohnung gewünschter Ereignisse und Bestrafung nicht gewünschter Ereignisse [56]. Im Kontext dieser Arbeit lediglich Supervised Learning angewendet, um das neuronale Netz zu trainieren.

---

<sup>1</sup>Neben Klassifikation und Regression sind weitere Arten von Ausgaben möglich, beispielsweise Embeddings von Autoencodern. Für den Kontext dieser Arbeit sind diese Arten der Ausgaben jedoch nicht relevant, weshalb sich auf die gängigen Ausgaben klassischer neuronaler Netze beschränkt wird.

Supervised Training basiert auf der Korrektur von Fehlern getätigter Vorhersagen des neuronalen Netzes. Als Forward Pass eines neuronalen Netzes wird die Vorhersage von Daten bezeichnet, durch die eine Ausgabe des Netzes erzeugt wird. Der Fehler von Vorhersagen wird durch eine Metrik gemessen, die hinsichtlich der Parameter des Netzwerks differenzierbar ist. Diese Metrik wird als Loss-Funktion<sup>2</sup> 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 iterativ Parameter der Netzwerkschichten angepasst, um die Vorhersage für die gegebenen Daten derart zu korrigieren, dass ein folgender Forward Pass auf den selben Daten einen geringeren Loss mit sich zieht [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 [56].

**Trainingsdaten** Dreh- und Angelpunkt des Trainings neuronaler Netze sind die Trainingsdaten. Sie werden genutzt, um die Parameter des zu trainierenden neuronalen Netzes anzupassen, indem getroffene Vorhersagen bewertet werden. Mit dieser Fehlerbewertung durch die Loss-Funktion werden die Parameter während der Backpropagation angepasst. Trainingsdaten haben üblicherweise die größte Kardinalität aller für das Training und die Evaluation verwendeten Datensätze.

Um ein effektives Training eines neuronalen Netzes zu gewährleiten, 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 ebenso wie die 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 wurden. Durch diese Daten 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 Verzerrung der Generalisierbarkeit zu vermeiden [56].

Die Wahl der Validierungsdaten unterliegt den gleichen Voraussetzungen wie den Trainingsdaten, um Verzerrungen in die Einblicke der Netzwerk-Performance zu vermeiden. 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.

**Testdaten** Nach dem Training eines neuronalen Netzes werden Testdaten genutzt, um die Netzwerk-Performance zu evaluieren. Während Trainings- und Validierungsdaten Einfluss

---

<sup>2</sup>Alternativ wird diese Funktion auch als Cost- oder Error-Funktion bezeichnet. In dieser Arbeit wird die Terminologie der Loss-Funktion verwendet.

auf den Verlauf des Trainings nehmen, werden Testdaten genutzt, um einen unabhängigen Einblick in die Netzwerkperformance nach Beendigung des Trainings zu gewinnen [56]. 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 einen universellen Überblick 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 bewusst von den Validierungs- und Testdaten ab, spricht man von Out-of-distribution(OOD)-Training. Das neuronale Netz wird auf Daten trainiert, die daher einer anderen Verteilung entsprechen als der zu erwartenden Daten für die Inferenz des Netzes.

**Under- und Overfitting** Die Validierungsdaten eines Trainings werden verwendet, um den Erfolg eines Trainings zu beurteilen. Während Optimierungsalgorithmen darauf ausgelegt sind, den Trainings-Loss zu minimieren, ist es möglich, dass der Validierungs-Loss von diesem abweicht. Befindet sich der Wert des Validierungs-Loss signifikant über dem Trainings-Loss, spricht man von Underfitting [56, 37]. In dieser Situation ist das neuronale Netz nicht in der Lage, das Gelernte auf neue Daten anzuwenden, da es noch nicht ausreichend trainiert wurde. 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 Was ist Augmentierung?

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 vorhandenen Daten kann eine Technik, die als Augmentierung bekannt ist, genutzt werden [37]. Als Augmentierung wird eine Manipulation der Trainingsdaten beschrieben, 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.

#### 4.1.5 Die YOLOv8-Architektur

YOLOv8 ist eine Netzwerkarchitektur, die 2023 durch Ultralytics außerhalb des Rahmens einer gesonderten wissenschaftlichen Aufarbeitung veröffentlicht wurde<sup>3</sup>. Mit YOLOv8 wurde eine Netzwerkarchitektur vorgestellt, die Objekterkennung durch Anchor-Free-Erkennungsmechanismen in Echtzeit und in hoher Genauigkeit ermöglicht [58]. Sie wurde als Verbesserung der YOLOv5-Architektur vorgestellt und ist für den Einsatz in Edge-Devices optimiert [59].

Der Aufbau von YOLOv8 folgt einem Design, welches in Backbone, Neck und Head unterteilt ist. Als Backbone wird eine erweiterte Version des CSPDarknet-Backbones

---

<sup>3</sup>Siehe: <https://github.com/ultralytics/ultralytics/issues/2572>

verwendet während der Neck auf dem Path Aggregation Network (PANet) basiert. Mit dem Aufbau von YOLOv8 wird das Fully-Convolutional-Paradigma neuronaler Netze verfolgt, welches die Verarbeitung von Bildern beliebiger Dimensionen 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 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 Non-Maximum-Suppression (NMS) nachverarbeitet.

Der Einsatzbereich von YOLOv8 spezifiziert sich auf Echtzeitanwendungen, in welchen robuste und akkurate Objekterkennung notwendig 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 und der Objekterkennung in Drohnen [60, 61].

## 4.2 Methodik

**Einleitende Sätze zur Methodik.**

### 4.2.1 Die YOLOv8\*-Architektur

Die Vorhersagen der Dartpfeilpositionen in normalisierten Bildern geschieht durch ein neuronales Netz der YOLOv8\*-Architektur. Diese Architektur basiert auf YOLOv8 [62], 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

Mit YOLOv8 wurde eine Netzwerkarchitektur vorgestellt, die in der Lage ist, Objekterkennung durch Single-Shot-Vorhersagen effizient und mit hoher Genauigkeit auszuführen [62]. Die vorgestellte Netzwerkarchitektur ist in der Lage, in Echtzeit ausgeführt zu werden und selbst in ressourcenbegrenzten Umgebungen hohe Genauigkeiten zu erlangen.

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 der Anzahl der 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.

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 u.a. 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 von Architektur und Aufgabe wurden strukturelle Änderungen im Netzwerkaufbau unternommen. Die Adaptierte

Netzwerkarchitektur wird im Folgenden als YOLOv8\* bezeichnet, um eine Differenzierung zwischen 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 signifikanten Vorteil. Das DeepDarts-System projizierte quadratische Bounding Boxes auf die Dartpfeilspitzen, um Outputdaten zum Training des Netzwerks zu generieren. Diese Bounding Boxes wurden im Verlauf des Trainings verkleinert, bis sie eine minimale Größe erreichten [3]. Die letztendliche Positions berechnung der identifizierten Dartpfeile bezog die Bounding Boxes nicht mit ein, da der Ankerpunkt der quadratischen Bounding Box die Position des Dartpfeils angibt. Die YOLOv8\*-Architektur wurde an ihren Einsatz angepasst, indem die Verwendung von Bounding Boxes aus der Architektur eliminiert wurde.

**Multi-Scale-Output** In der YOLOv8-Architektur wird ein Multi-Scale-Output zur Identifizierung von Objekten unterschiedlicher Größen eingesetzt. Objekte werden in 3 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 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  $800 \times 800$  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 3 Würfen besteht, ist davon auszugehen, dass eine maximale Anzahl von 3 Dartpfeilen in den Bildern zu identifizieren sind. Dabei kann jedoch nicht ausgeschlossen werden, dass sich die Dartpfeilspitzen in unterschiedlichen Regionen befinden. Je Region wird daher eine feste Anzahl von 3 möglichen Dartpfeilpositionen vorhergesagt. Existiert lediglich ein Dartpfeil, so sind zwei mögliche Outputs genutzt.

**Transition-Block** 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 den Fully-Convolutional-Approach der YOLO-Netzwerke durch Einbindung von Dense-Schichten. Die Eigenschaft von Fully-Connected-CNNs, Bilder beliebiger Eingabegrößen verarbeiten zu können, geht mit der Einschränkung einher, keinen direkten globalen Kontext des Bildes einzufangen. Da die Eingabegrößen der Bilder der in dieser Thesis erarbeiteten Systems durch die Vorverarbeitung vorgegeben sind, liefert die Verarbeitung beliebiger Eingabegrößen keinen Mehrwert und ermöglicht damit die Lockerung dieses Paradigmas. Folglich ist eine Einbindung von Fully-Connected-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.

#### 4.2.1.3 Aufbau der Architektur

Der Aufbau des 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 wurde ein gesonderter Netzwerkabschnitt zugeteilt, in welchem die Eliminierung des Multi-Scale-Outputs sowie die gesonderte Handhabung der Outputs manifestiert ist. Der Detection-Bereich ist 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.

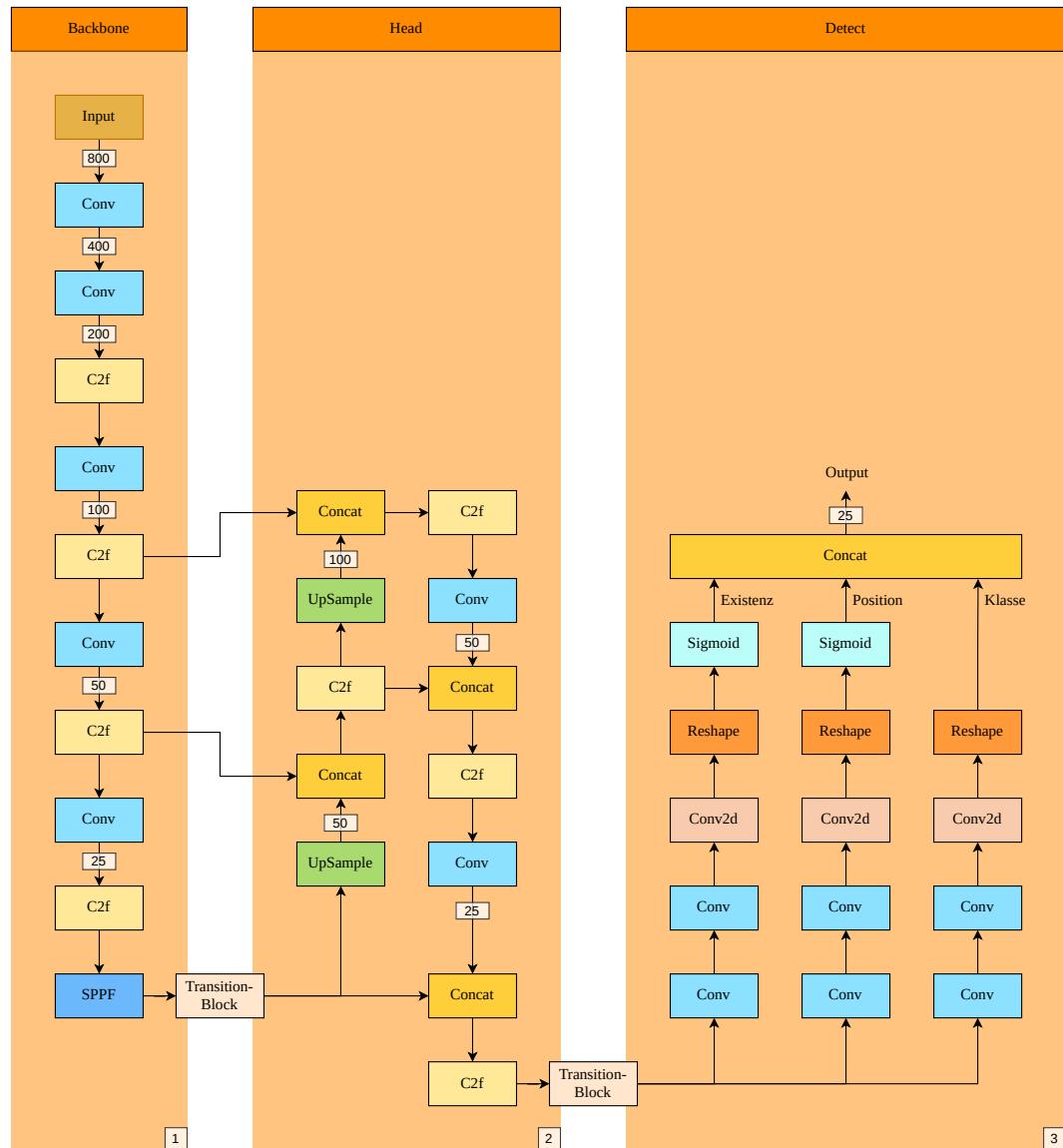


Abbildung 4.1: YOLOv8\*-Architektur. (1) Bottleneck; Extraktion von Features. (2) Head; Kombination von Features. (3) Detect; Deutung von Features.

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.

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

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 wurde 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 eines Dense-Layers. 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 in Vergleich zu DeepDarts komplexeren Daten wurde sich für die Verwendung von YOLOv8\*-s entschieden, welche mit etwa 10,5 Millionen Parametern etwa 40% mehr Parameter als YOLOv4-tiny besitzt.

## 4.2.2 Loss-Funktionen

Die Loss-Funktion, die für das Training der YOLOv8\*-Architektur verwendet wurde, wurde 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 zum Trainieren der YOLOv8\*-Architektur folgt dem Aufbau dieser. 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  $\mathcal{L}_{\text{xst}}$**  Für den Existenz-Loss  $\mathcal{L}_{\text{xst}}$  wird der Focal-Loss der Existenz-Einträge aller Einträge aller Regionen gebildet [63]. Die Verwendung des Focal-Loss für diesen Loss liegt in der spärlichen Menge positiver Einträge in Output-Tensoren. Bei einer Auflösung von  $25 \times 25$  Regionen und 3 Vorhersagen je Region besitzt der Output-Tensor  $n_{\text{entries}} = 25 \times 25 \times 3 = 1875$  Einträge. Zu erwarten sind  $n_{\text{positive}} \leq 3$  positive Einträge, sodass der maximal zu erwartende Prozentsatz positiver Einträge  $p_{\text{positive}} = \frac{n_{\text{positive}}}{n_{\text{entries}}} \leq \frac{1}{625} = 0.16\%$ . Der Focal-Loss gewichtet positive und negative Klassen, sodass ein gezieltes Erlernen trotz signifikanten Klassenungleichgewichts.

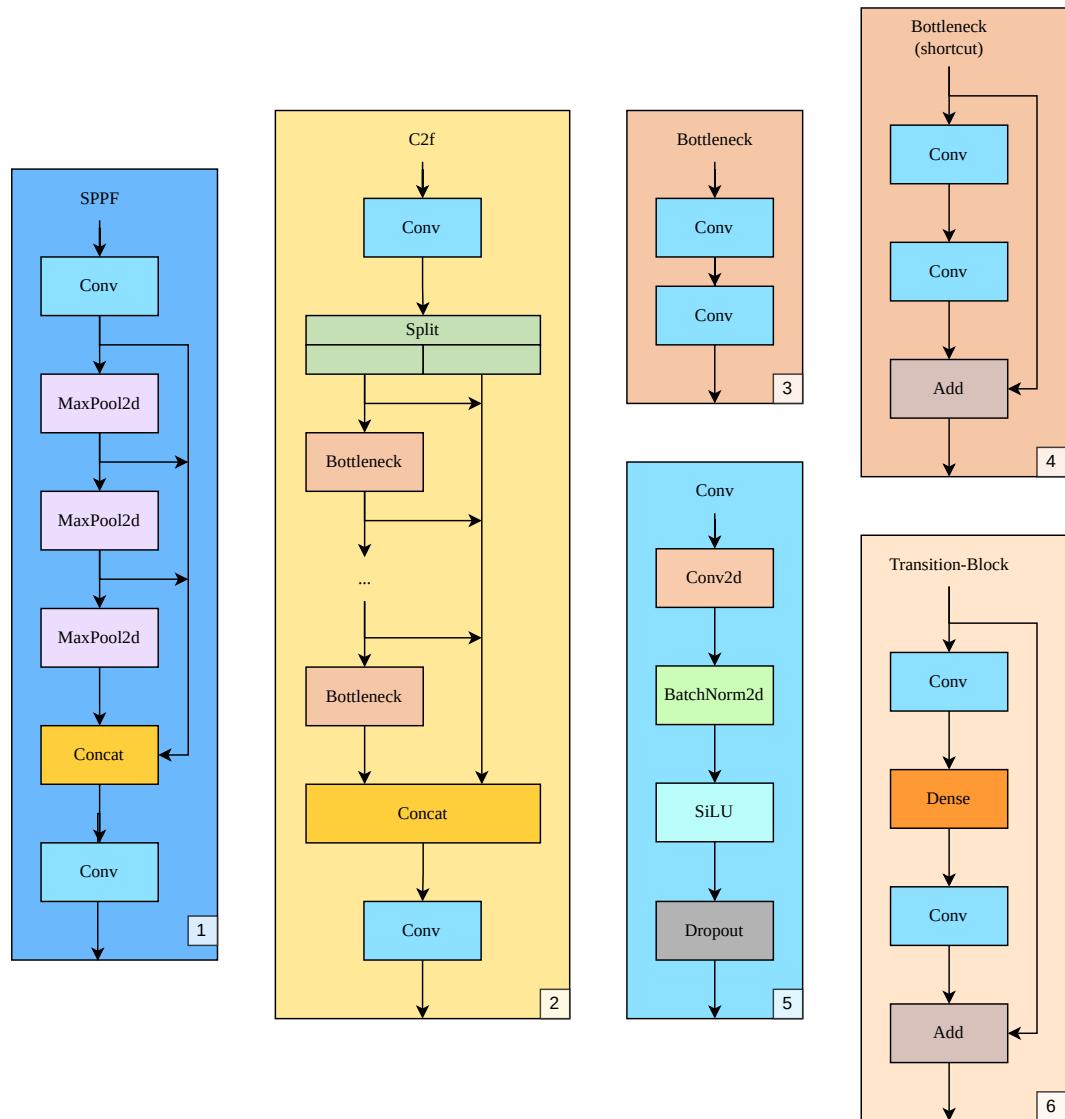


Abbildung 4.2: Netzwerkbestandteile der YOLOv8\*-Architektur. (1) SPPF-Block; dieser zeichnet sich durch Hintereinanderreihung von MaxPool2d-Schichten aus. (2) C2f-Block; dieser spaltet die Feature Maps auf und wiederholt Bottleneck-Blöcke an. (3, 4) Bottleneck-Block ohne und mit Shortcut; wiederholte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

**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 kategorische Focal-Kreuzentropie (Categorical Focal Cross-Entropy) [63] 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).

**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, beträgt der Loss-Wert 0.

**Gewichtung** Durch die Verwendung unterschiedlicher Losses für Existenz, Klasse und Position resultieren unterschiedliche Bildbereiche der Loss-Werte. Die Loss-Gewichte  $\omega_{xst}$ ,  $\omega_{cls}$  und  $\omega_{pos}$  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 wurden die gewichtete  $\omega_{xst} = 400$ ,  $\omega_{cls} = 2000$  und  $\omega_{pos} = 0.5$  verwendet.

#### 4.2.2.2 Hintergründe und Zielsetzung

Der Hintergrund der Aufteilung des Losses in Existenz, Klasse und Position liegt in der Netzwerkarchitektur. Diese ist auf die Vorhersage von Existenz, Klasse und Position ausgelegt, um Dartpfeile für eine Punktzahlbestimmung zu lokalisieren. Durch 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 die Überschattung eines Teil-Losses durch einen anderen mit weitaus größerem Wert entgegengewirkt. Ein Overfitting eines Bereichs ist dadurch wahrscheinlicher in dem kombinierten Loss widergespiegelt, sodass auf diesen dynamisch während des Trainings eingegangen werden kann.

#### 4.2.2.3 Abweichung von DIoU-Loss

Bei dem Training von YOLO-Architekturen ist die die Verwendung vom IoU-Losses oder Adaptionen wie CIoU, DIoU oder GIoU [64], eine übliche Praxis [65, 62, 66]. Diese Erweiterungen des IoU-Losses stützen sich grundlegend auf der Annahme der Existenz von Bounding Boxes. Da diese in der YOLOv8\*-Architektur eliminiert wurden, 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 möglich wäre. 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 echte 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 und im Detail beschrieben.

#### 4.2.3.1 Trainingsdaten

Die Trainingsdaten bilden die Basis des Trainings eines neuronalen Netzes. Für das Training des DeepDarts-Systems wurden wenig diverse Daten verwendet, wodurch die Performance des Systems beeinträchtigt wurde, wie bereits in Abschnitt 3.4 dargestellt wurde. Um diesem Phänomen der einseitigen und wenig diversen Daten entgegenzuwirken, wurde in dieser Thesis auf die Nutzung eigener, synthetisch generierter Daten gesetzt, die durch Salting echter Daten angereichert wurden. Die Datenerstellung erfolgte nach dem in Kapitel 2 beschriebenen Prinzip. Zum Salting wurden einerseits Daten aus dem DeepDarts-Datensatz verwendet, sowie manuell aufgenommene Daten aus einem Lokal.

Die überwiegende Mehrheit der Trainingsdaten wird durch die generierten Daten ausgemacht mit dem Ziel, durch diese ein grundlegendes Verständnis der zu lösenden Aufgabe der Klassifikation von Existenz und Feldfarbe sowie der Regression von Positionen der Dartpfeilspitzen zu erlangen. Das Salting durch eine geringe Menge echter Daten dient der Festigung der erlernten Grundprinzipien und der Adaption auf echte Daten zur Minimierung des Risikos des Overfittings. Daten, die zum Salting verwendet werden, besitzen zur Regulierung von Kardinalitätsunterschieden eine höhere Gewichtung als generierte Daten.

Supervised Training setzt die Existenz sowohl von Inputs als auch korrekten 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 mit dem Farbformat BGR und Abmessungen von  $800 \times 800 \text{ px}^4$ .

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

#### 4.2.3.2 Validierungsdaten

Zusätzlich zu den Trainingsdaten werden für das Supervised 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 wurde 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 den Trainingsdaten 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.

#### 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 Verteilungsungleichheit der Klassen schwarzer und weißer Felder im Vergleich zu roten und grünen Feldern. Um diesem Klassenungleichgewicht entgegenzuwirken, wurde ein Oversampling roter und grüner Felder sowie ihrer unmittelbaren Umgebungen durchgeführt.

---

<sup>4</sup>Da es sich bei dem trainierten Netzwerk um ein Fully Convolutional Neural Network handelt, ist eine Festlegung auf konkrete Bilddimensionen nicht notwendig. Zur Vereinheitlichung der Daten und zur Normalisierung der Eingaben wurde jedoch eine feste Größe verwendet.

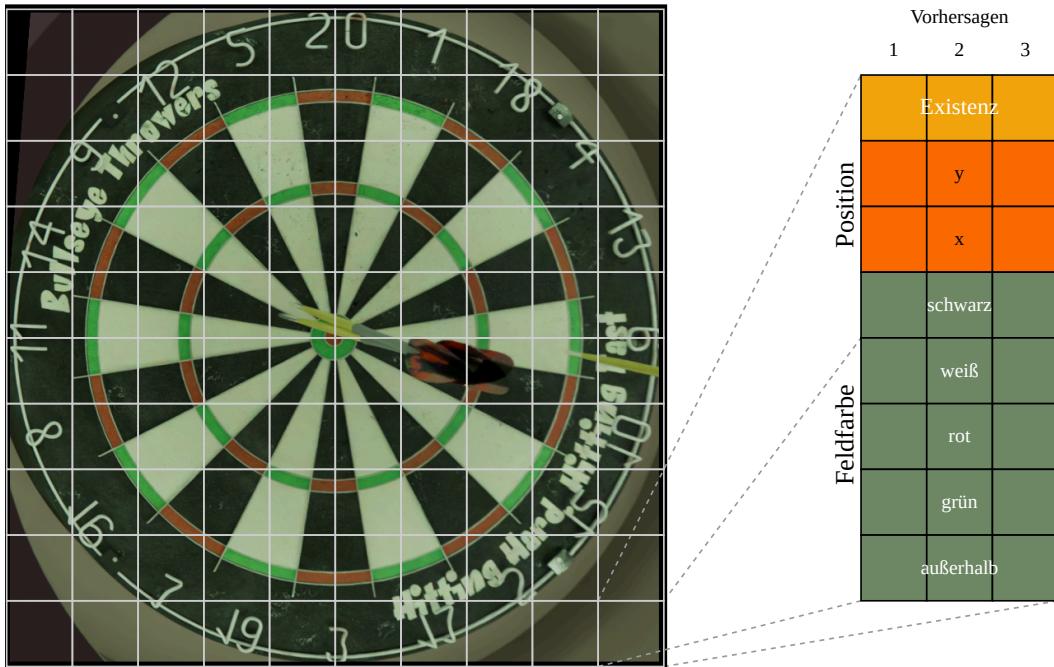


Abbildung 4.3: Schematische Darstellung des Output-Datenformats. Die Anzahl der Output-Zellen wurde hinsichtlich der Übersichtlichkeit auf  $10 \times 10$  Regionen begrenzt.

Durch das Oversampling ist ein übermäßiger Prozentsatz unterrepräsentierter Klassen in den Daten vorhanden, der zu einem ausgeglichenen Lernen aller Klassen führt [67]. Ohne Oversampling flächenmäßig kleiner Felder der Dartscheibe in den Trainingsdaten resultierte die Vorhersage von Double-, Triple- oder Bull-Pfeilen 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 [68]. Für diese Thesis wurden 24 576 Daten erstellt, davon 20 480 reguläre Daten und 4096 Oversampling-Daten. Der Prozentsatz des Oversamplings beläuft sich damit auf 16.67% der generierten Daten.

Die Heatmaps zum Generieren der für das Oversampling genutzten Wahrscheinlichkeitsverteilungen sind in Abbildung 2.6 dargestellt; Details zur Datenerstellung mittels dieser Heatmaps wurden in Unterabschnitt 2.3.3 (Generierung von Dartpfeil-Positionen) erläutert.

#### 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. Durch Augmentierung werden neue Daten aus bereits vorhandenen Daten abgeleitet, indem die Parameter der Daten manipuliert werden. Durch diese Technik wird eine größere Bandbreite möglicher Parameter und resultierend mögliche Eingaben in das System abgedeckt [69, 70]. Durch Verwendung von Datenaugmentierung ist eine generelle Verbesserung der Netzwerkperformance zu erwarten [71].

Für die Augmentierung von Bilddaten existieren unterschiedliche Herangehensweisen [72]. Konkret wurden 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** Unter Pixel-Manipulation versteht sich die Änderung von Pixel-Werten der Eingabebilder, die keine Beeinflussung der Output-Daten mit sich ziehen. Diese Manipulation ist in 5 Unterschritte aufgeteilt, die sukzessiv auf die Input-Daten angewandt werden. Die Manipulation beginnt mit einer Aufhellung oder Verdunklung

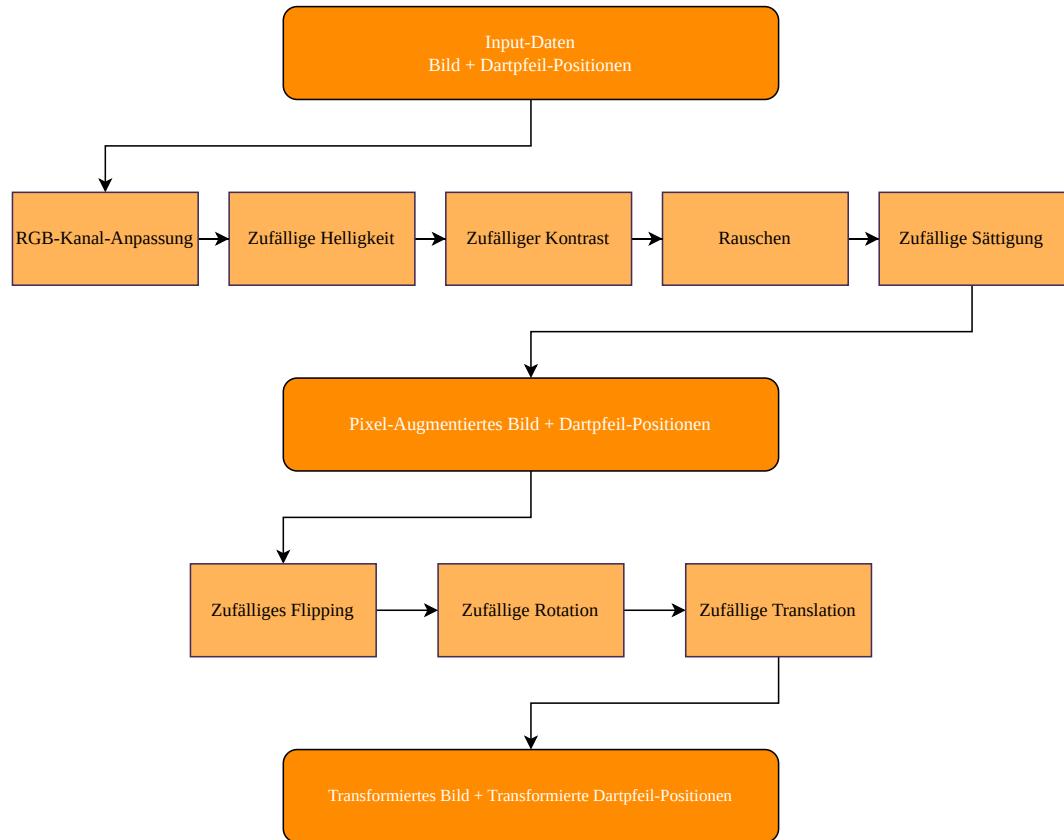


Abbildung 4.4: Pipeline der Daten-Augmentierung.

einzelner Farbkanäle der Bilddaten, gefolgt von zufälliger Helligkeits- und Kontrastanpassung. Danach werden die Bilddaten mit einem normalverteilten Gaußschem Rauschen angereichert und zuletzt wird die Sättigung des resultierenden Bildes zufällig gesetzt.

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 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 Transformation besteht aus drei Schritten: Flipping, Rotation und Translation. Bei dem Flipping wird das Bild zufällig horizontal und vertikal gespiegelt. Die Rotation rotiert das Bild um Ganzzahlige Vielfache von  $18^\circ$  um den Mittelpunkt. Diese Rotationsinkremente entsprechen den Feldlinienwinkel der Dartscheibe und sorgen dafür, dass die Wertigkeit der Dartfelder rotiert, jedoch die Position der Feldlinien erhalten bleibt, da diese durch die in Kapitel 3 vorgestellte Vorverarbeitung vorgegeben ist. 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 Parameterwerte mit sich zieht. Für das Training in dieser Thesis wurde eine adaptive Learning Rate verwendet, die dynamisch auf den Verlauf des Trainings reagiert und für eine gezielte Konvergenz sorgt [73].

Es wurde ein Learning Rate Schedule gewählt, der die Learning Rate zu Beginn des Trainings ansteigen lässt und sie heruntersetzt, sobald über eine vorbestimmte Dauer keine Verbesserung des Validation-Losses geschehen ist:

$$\text{lr}(e) = \begin{cases} \frac{\text{lr}_0}{e+1}, & \text{wenn } e < e_{\text{warmup}}, \\ \min(\text{lr}_{\text{min}}, \text{lr}(e-1) \cdot f_{\text{lr}}), & \text{wenn } \min(\text{val\_loss}) \notin \{\text{val\_loss}_i \mid i \in [0, e - w_{\text{lr}}]\}, \\ \text{lr}(e-1), & \text{ansonsten.} \end{cases}$$

Es wurden Parameterbelegungen  $\text{lr}_0 = 0.001$ ,  $\text{lr}_{\text{min}} = 10^{-6}$ ,  $e_{\text{warmup}} = 8$ ,  $f_{\text{lr}} = 0.1^{0.25}$  und  $w_{\text{lr}} = 50$  gewählt. Der Wert von  $f$  wurde derart gesetzt, dass die Learning Rate nach 4-facher Verringerung um einen Faktor 10 gemindert wurde. Mit den gewählten Werten wird die Anzahl der Verringerungen der Learning Rate im Verlaufe eines gesamten Trainings auf 12 festgelegt, sofern das Training nicht zuvor abgebrochen wird.

#### 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 [74, 75, 76, 77, 78].

Daten zum Training, Graphen etc - EDIT: Das ist teils in Implementierung schon. Überdenken!  
Einleitende Sätze zur Implementierung

### 4.3 Implementierung

#### 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 wurde TensorFlow als Framework für neuronale Netze verwendet, welches nicht reibungslos mit PyTorch vereinbar ist. Die Frameworks arbeiten auf unterschiedliche Arten, wodurch eine Übersetzung der verwendeten Schichten und vortrainierten Gewichten 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 nicht trivial möglich. Auf Grundlage der offiziellen Dokumentation sowie des Quelltexts und Konfigurationsdateien wurde die YOLOv8-Architektur in TensorFlow übersetzt und implementiert. Atomare Bestandteile wie Convolution-Schichten, Batch-Normalisierung, Pooling-Operationen und Aktivierungsschichten sind analog von PyTorch zu TensorFlow zu übertragen. Schichten, die nicht als Schicht in TensorFlow verfügbar sind – beispielsweise die Split-Operation – werden als TensorFlow-Operationen in dem Netzwerk ausgeführt.

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

Nachdem die grundlegende Architektur in TensorFlow verfügbar war, konnte sie durch eigene Adaptionen erweitert werden. Eine wesentliche Änderung ist das Hinzufügen von Dropout-Schichten, welche nicht in der Dokumentation erwähnt wurden, jedoch die Stabilität des Trainings erhöhen und die Wahrscheinlichkeit des Overfittings senken. Das Hinzufügen dieser Schichten konnte durch den modularen Aufbau der Architektur einfach vollzogen werden.

Eine weitere Adaption ist das Hinzufügen des Transition-Blocks, welcher eine residuale Fully-Connected-Schicht ist. In diesem Block wird der Eingabetensor durch einen Tensor moduliert, der den globalen Kontext des Eingabetensors durch eine Fully-Connected-Schicht einfängt. Die Implementierung dieses Blocks reiht sich in die Implementierungen der restlichen Blöcke ein, indem auf die Grundbausteine zugreift. Lediglich die Fully-Connected-Schicht wurde als Grundbaustein ergänzt, da diese aufgrund des Fully-Convolutional-Paradigmas nicht in der YOLOv8-Architektur vorkommt.

### 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.

#### 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 16 verwendet mit dem AdamW-Optimizer und einer adaptiven Learning Rate, wie in Unterabschnitt 4.2.3.5 erläutert. Für das Training wurden 24 960 Trainings- und 544 Validierungsdaten verwendet. Die Trainingsdaten setzen sich zusammen aus 24 576 generierten Daten, 256 Daten des DeepDarts-Trainings-Datensatzes und 128 Daten, die privat aufgenommen wurden. Die Validierungsdaten sind zusammengesetzt aus 256 synthetischen Daten, 128 Daten des DeepDarts-Validierungs-Datensatzes sowie 160 manuell aufgenommen Daten.

#### 4.3.2.2 Augmentierungsparameter

Die Daten wurden vektorisiert als TensorFlow Datasets eingelesen, wodurch ein effizientes und parallelisierte Laden der Daten ermöglicht wird. Teil der Pipeline zum Einlesen der Daten ist die Augmentierung der Trainingsdaten. Die Augmentierung wird dynamisch auf jedes Bild der Trainingsdaten mit zufälligen Parametern angewendet. 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  $b_{\text{img}}$  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 je horizontal und vertikal mit einer Wahrscheinlichkeit von 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  $\sigma_{\text{trans}} = 5 \text{ px}$ .

#### 4.3.2.3 Verlauf des Trainings

**Trainings-Epochen eingehen**

Das Training mit einer unbestimmten Anzahl an Epochen gestartet und dynamisch nach **XXX** Epochen abgebrochen, nachdem keine weitere Verbesserung des Validierungs-Losses verzeichnet werden konnte. Der Verlauf des Trainings ist in Abbildung 4.5 dargestellt. Die Abbildung ist unterteilt in die Teil-Losses der Existenz, Klassen und Positionen sowie den

kombinierten Loss. Die Graphen der Trainings- und Validierungs-Losses fallen gemeinsam, jedoch besteht eine Diskrepanz zwischen diesen Werten. Der Verlauf der Validierungs-Losses deuten weder auf Over- noch Underfitting hin.

## 4.4 Ergebnisse

### 4.4.1 Metriken

Für die Auswertung der Genauigkeit der jeweiligen Systeme werden mehrere Metriken verwendet. Zur Auswertung von DeepDarts wurde PCS verwendet, um die relative Anzahl korrekt ausgewerteter Scores zu bestimmen. Diese Metrik ist jedoch dahingehend fehleranfällig, dass positive Ergebnisse trotz Fehlklassifikationen zustande kommen können. PCS misst die Fähigkeit, die korrekte Punktzahl vorherzusagen, statt der Fähigkeit, die Dartpfeile korrekt zu ermitteln. Um Einblick in dieser Fähigkeiten der Systeme zu gewinnen, werden in dieser Arbeit drei weitere Metriken verwendet: Existenz  $\mu_X$ , Klasse  $\mu_K$  und Position  $\mu_P$ .

Einleitende Sätze: NN-Ergebnisse

#### 4.4.1.1 Existenz-Metrik $\mu_X$

Mit dieser Metrik wird bestimmt, ob die korrekte Anzahl der Dartpfeile bestimmt wurde.  $\mu_X$  ist definiert als:

$$\mu_X = \frac{1}{N} \sum_{i=1}^N 1 - \frac{1}{3} | N_{\text{Dart}, i} - \hat{N}_{\text{Dart}, i} |$$

In dieser Formel stehen  $N_{\text{Dart}, i} \in \mathbb{N}$  und  $\hat{N}_{\text{Dart}, i} \in \mathbb{N}$  für die Anzahl vorhandener und vorhergesagter Dartpfeile je Bild mit Index  $i$ . Anhand des Werts von  $\mu_X$  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 Dartpfeil-Existenz und Dartpfeil-Position wird in dieser Metrik nicht festgehalten.

#### 4.4.1.2 Klassen-Metrik $\mu_K$

$\mu_K$  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:

$$\mu_K = \frac{1}{N} \sum_{i=1}^N \frac{1}{3} N_{K, \text{correct}, i}$$

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

#### 4.4.1.3 Positions-Metrik $\mu_P$

Ziel dieser Metrik ist es, die durchschnittlichen Abweichungen der Dartpfeilspitzen einzufangen. Die Dartpfeilspitzen werden analog zu  $\mu_K$  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 Fähigkeit, mit welcher Präzision Dartpfeilspitzen erkannt werden. Der Wert von  $\mu_P$  ergibt sich aus:

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

$P_{i,d} \in \mathbb{R}^2$  und  $\hat{P}_{i,d} \in \mathbb{R}^2$  sind die gegebenen 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. Diese Eigenschaft der Metrik wird analog zu  $\mu_K$  durch die Auswertung von  $\mu_X$  abgebildet.

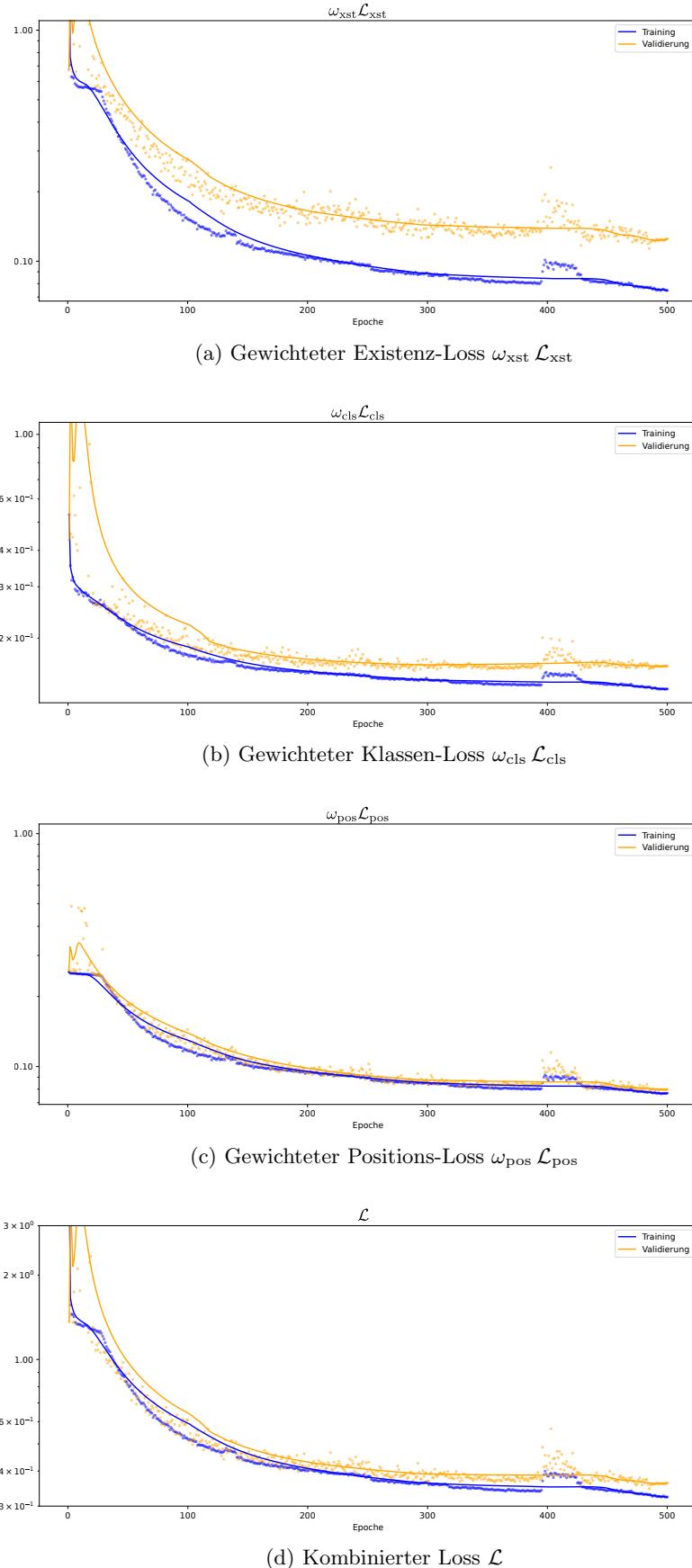
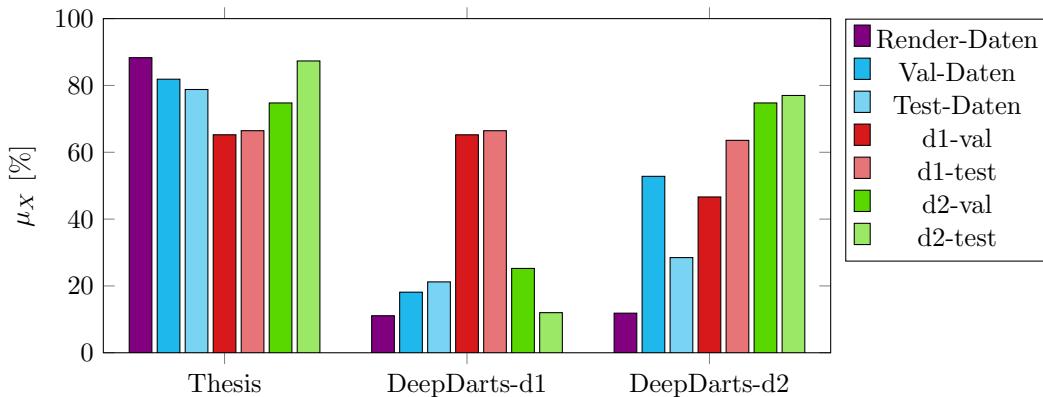


Abbildung 4.5: Trainingsverlauf der YOLOv8\*-Architektur. Die Loss-Werte sind als Punkte dargestellt, welche durch eine Linie der entsprechenden Farbe geglättet dargestellt sind. Trainings-Losses werden in blau dargestellt, Validierungs-Losses in orange.

Datenquelle	Gerenderte Bilder	Reale Bilder Validierung	Test	DeepDarts-d1 Validierung	Test	DeepDarts-d2 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  $\mu_X$  der Systeme auf unterschiedlichen Datenquellen.

#### 4.4.2 Datenquellen und Herangehensweise

Für die Auswertung der Performance des neuronalen Netzes wurden Daten unterschiedlicher Quellen verwendet, aufgelistet in Tabelle 4.2. Analog zur Auswertung der algorithmischen Normalisierung der Bilder wurden die selben gerenderten Bilder sowie die Validierungs- und Test-Daten von DeepDarts einbezogen. Zusätzlich wurden Bilder echter Darts-Runden 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 wurden.

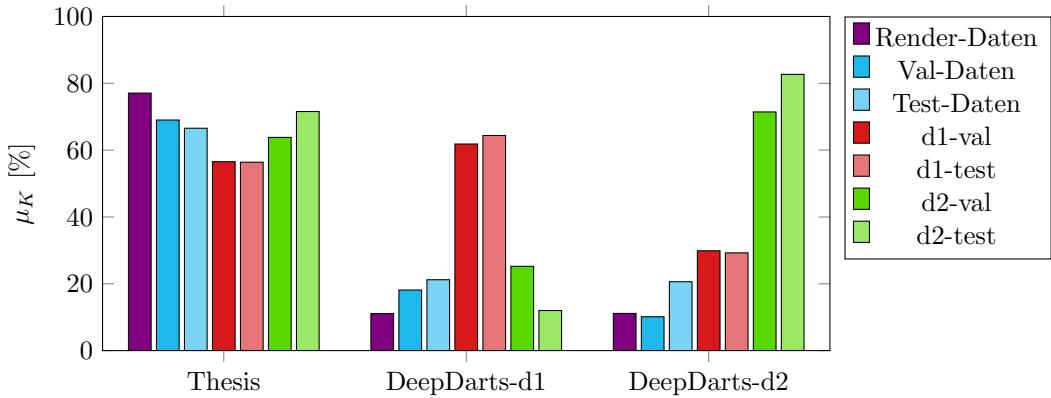
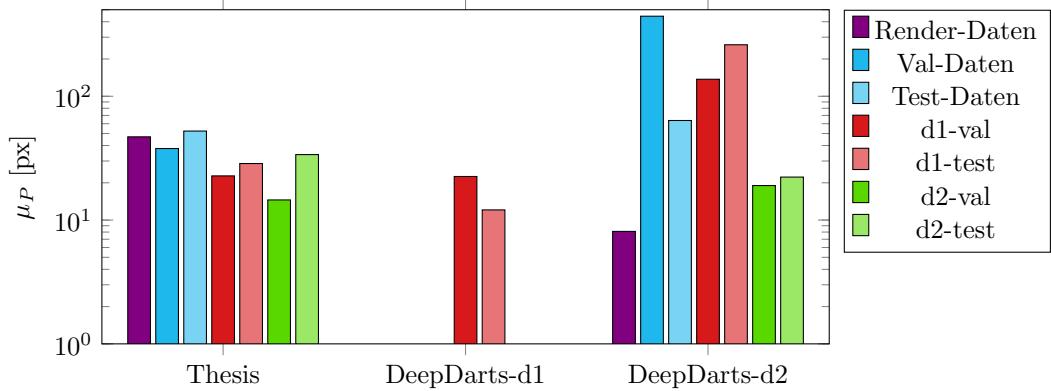
Die in den folgenden Unterabschnitten dargestellten Auswertungen stellen die Ergebnisse des gesamten Systems dar unter Einbezug der Normalisierung. Hintergrund dieses Zusammenschlusses der Verarbeitungsschritte ist die Vergleichbarkeit mit DeepDarts, in welchem die Verarbeitungsschritte miteinander verschmolzen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

#### 4.4.3 Auswertung der Existenz-Metrik $\mu_X$

Die Auswertungen der Existenz-Metrik  $\mu_X$  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 wurde mit 88% auf den gerenderten Daten erzielt während die geringsten Metrik-Werte auf den Validierungs-Daten von DeepDarts-d1 mit 65% erzielt wurden.

Die Auswertung von DeepDarts-d1 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 des Systems waren. Auf diesen Daten wird ein durchschnittlicher Wert von 17% erzielt.

DeepDarts-d2 zeigt eine weitaus bessere Auswertung als DeepDarts-d1, sodass auf DeepDarts-d1-Daten durchschnittlich 56% Existenz erzielt werden konnten und bei DeepDarts-d2-Daten durchschnittlich 76%. Die Auswertungen auf den für diese Auswertung aufgenommenen echten Daten liegen bei 20% und bei den gerenderten Daten sind es lediglich 11%.

Abbildung 4.7: Auswertung von  $\mu_K$  der Systeme auf unterschiedlichen Datenquellen.Abbildung 4.8: Auswertung von  $\mu_P$  der Systeme auf unterschiedlichen Datenquellen. Je geringer die Werte, desto besser die Auswertung.

#### 4.4.4 Auswertung der Klassen-Metrik $\mu_K$

Die Auswertung der unterschiedlichen Systeme hinsichtlich der Metrik  $\mu_K$  ist in Abbildung 4.7 dargestellt. Die Resultate spiegeln die Auswertung von  $\mu_X$  wider indes die relativen Verteilungen ähnlicher Struktur sind. Die Auswertung des Ansatzes dieser Thesis zeigen Werte im Bereich um 70% für die Daten, die für diese Arbeit erstellt wurden mit einer signifikant besseren Auswertung gerenderter Daten im Vergleich zu realen Daten. Auf den DeepDarts-d1-Datensätzen konnten mit 56% die geringsten Resultate erzielt werden während auf den DeepDarts-d2-Daten im Mittel 68% erzielt wurden. Mit diesen Resultaten liegen die Genauigkeiten der Findung von Feldfarben unter den Genauigkeiten der Fähigkeit, die Dartpfeile zu identifizieren.

DeepDarts-d1 zeigt hinsichtlich  $\mu_K$  ähnliche Auswertungen zu  $\mu_X$ : 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 wurden lag im Durchschnitt lediglich eine Klassen-Genauigkeit von 17% vor.

Ähnliche Züge der Evaluation hinsichtlich  $\mu_K$  sind für DeepDarts-d2 zu verzeichnen: Die d2-Datensätze werden mit einer hohen Genauigkeit von durchschnittlich 77% erkannt während weitere Datensätze mit durchschnittlich 20% Genauigkeit ausgewertet sind. Die Fähigkeit, Feldfarben korrekt zu identifizieren, liegt bei DeepDarts-d2 deutlich unter der Fähigkeit, Dartpfeile zu identifizieren. Die Bestimmung korrekter Scores je Dartpfeil ist dadurch stark beeinträchtigt.

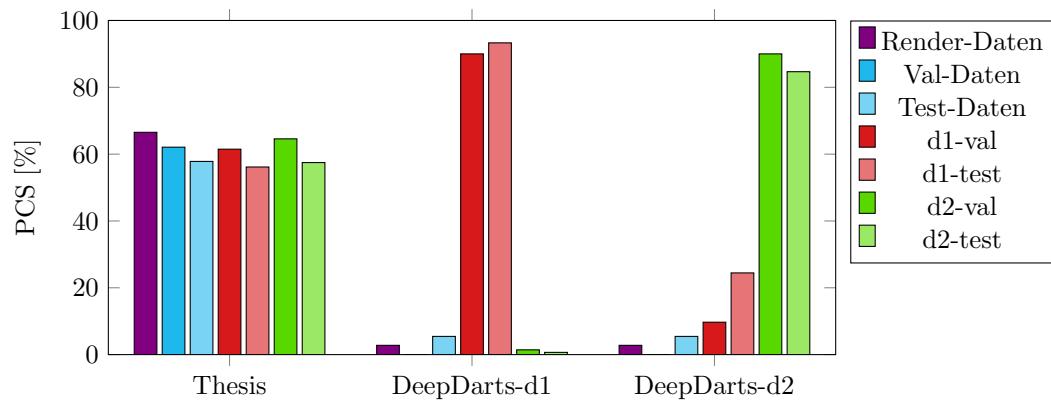


Abbildung 4.9: Auswertung von PCS der Systeme auf unterschiedlichen Datenquellen.

#### 4.4.5 Auswertung der Positions-Metrik $\mu_P$

rtung beschrei-

#### 4.4.6 Auswertung der PCS-Metrik

Es zeichnet sich ein ähnliches Bild wie in Abbildung 3.13 ab, indem eine signifikant höhere Performance

PCS-Auswertung beschrei-  
ben

# Kapitel 5

## Diskussion

In diesem Kapitel werden die Ergebnisse und Beobachtungen der Unterprojekte dieser Thesis aufgefasst. Diese werden miteinander diskutiert und mögliche Schwachstellen sowie Ungereimtheiten werden aufgezeigt. 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 aufgekommen, 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 echte 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 3360 möglichen Dartpfeilen ergibt, jedoch ist der limitierende Faktor die Wiederverwendung existierender Pfeile. Das neuronale Netz erfährt während des Trainings nicht mehr als 7 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 einen Dartpfeil. 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 Umgebungstexturen, verwendet, aus dem für jedes Bild ein zufälliges Bild ausgewählt wurde und in seiner 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 5 unterschiedliche

Lichtquellen stark limitiert. Obgleich diese Lichtquellen in ihrer Zusammensetzung kombiniert und modifiziert werden, kann die Anzahl der Beleuchtungsmöglichkeiten eine Verzerrung bzw. Einseitigkeit in die Daten einfließen lassen.

Zuletzt wird ein statisches Compositing verwendet, in welchem Imperfektionen der Kamera, Kontrast und Rauschen auf das Kamerasbild gelegt wird, 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 generelle 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 echten Daten und synthetischen Daten mit geringer Unsicherheit möglich. Die Aufnahmen der synthetischen sind ohne Probleme als synthetische Aufnahmen zu identifizieren. Zwar ist gezeigt worden, dass diese Daten den Kern der Datenlage echter Aufnahmen einfangen und diese für ein Training eines neuronalen Netzes ausreichen, jedoch ist die Generalisierung der gelernten Charakteristiken und Merkmale auf echte Bilder nicht fehlerfrei möglich. Durch starke Augmentierung der Bilder, beispielsweise Kontrastanpassung und Hinzufügen von Rauschen, werden diese Bilder derart verzerrt, dass die Diskrepanz zwischen augmentierten echten und augmentierten synthetischen Bildern geringer 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 echter 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, mit dem 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 echter Dartscheiben ist dieser Unterschied deutlich zu sehen, indem Dartscheibe und Dartpfeile durch die umliegenden Objekte indirekt beleuchtet wurden.

### 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 weniger Pixel resultieren kann. Da für die Lokalisierung der Dartscheibe 4 Orientierungspunkte verwendet wurden, welches die minimale Anzahl notwendiger Punkte zur Berechnung einer Entzerrung 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 2 Dimensionen (Fläche) auf 0 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 headless ohne Nutzeroberfläche 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 zeitaufwendiger 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 von Dartscheiben

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

DeepDarts hat ein System vorgestellt, welches durch die Verwendung neuronaler Netze zuverlässige Ergebnisse auf ihren Daten erzielen konnte. Von diesem Ansatz wurde sich aus Gründen der Flexibilität und des vorhandenen Hintergrundwissens im Bereich der herkömmlichen Computer Vision gelöst, indem der Prozess zweigeteilt wurde. Die resultierende Aufteilung in algorithmische Normalisierung und trainierte Dartfeilerkennung zieht sowohl Vorteile als auch Nachteile mit sich.

Da die Schritte der Normalisierung algorithmisch durchgeführt werden und die Funktionsweise dadurch bekannt ist, kann Fehlerfällen gezielt nachgegangen werden und das System kann gezielt erweitert werden. Aus dieser Herkunft stammt jedoch das relevanteste Performance-Problem dieses Ansatzes: die Ausführungszeit. Neuronale Netze verwenden eine Vielzahl abstrakter Informationen parallel, um zu einem Ergebnis zu gelangen wohingegen algorithmische Methoden eine geringe Zahl konkreter Informationen verwenden. Durch die massive Parallelität neuronaler Netze ist die Verarbeitung der Daten effizient während die algorithmische Verarbeitung der Daten weitestgehend in sequenziellen Stufen abläuft. Trotz der geringeren Datenmenge ist die Ausführungszeit durch die sequenzielle Natur der Herangehensweise limitiert. Zudem ist die hauptsächliche Implementierung des Algorithmus dieser Thesis in Python implementiert, welches trotz der Verwendung komplizierter Bibliotheken eine interpretierte Sprache ist und daher weitaus längere Ausführungszeiten aufweist als in kompilierten Programmiersprachen zu erwarten ist. Neuronale Netze werden hingegen weitestgehend kompiliert und parallel ausgeführt.

### 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 geringes Vertrauen in die Resultate und großzügiges Thresholding von Daten. Zur Identifizierung einer Entzerrung der Dartscheibe ist ein Minimum von 3 Punkten

notwendig, welche aus bis zu 60 Kandidaten erkannt werden können. Es sind daher lediglich 5% der möglichen Orientierungspunkte zwingend notwendig. 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, da RANSAC auf zufälligem Samplen von Datenpunkten basiert. Der Algorithmus produziert daher keine eindeutigen Resultate, sodass unterschiedliche Durchläufe auf dem selben Bild zu unterschiedlichen Ergebnissen und im Extremfall sogar teilweise fehlschlagen 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 1600 px iterativ um Faktor 2 verkleinert. Der Hintergrund dieser Skalierung ist der Zeitaufwand der Operationen. Sowohl Genauigkeit als auch Berechnungsdauer der Algorithmen skaliert mit der Größe der Bilder. Um eine zeitliche Obergrenze der Berechnungsdauer zu setzen wurde sich für eine Vorverarbeitung zur Reduktion der Bildgröße entschieden. Der mit dieser einhergehenden Informationsverlust im Bild sorgt potenziell für Ungenauigkeiten bei der Berechnung der Normalisierung.

## 5.3 Diskussion der Dartpfeil-Erkennung durch neuronale 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, der spezifische Änderungen unterzogen wurden, um sie auf die zu lösende Aufgabe anzupassen. 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 bereits trainierten Gewichte. Durch die eigene Implementierung in einem anderen Framework zur Adaption der Architektur ist das Zurückgreifen auf diese vortrainierten Gewichte des offiziellen Modells nicht möglich. YOLOv8 wurde mit Hilfe von PyTorch entwickelt und implementiert während TensorFlow in dieser Arbeit verwendet wurde. Vortrainierte Modelle beherbergen den Vorteil, deutlich mehr Bildern ausgesetzt gewesen zu sein, sodass die Parameter des Netzwerks in derartigem Einklang miteinander sind, dass universelle Strukturen erkannt werden können. 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 zur Adaption auf die eigene Aufgabe vorhandene Objekte in den Parametern des Netzwerks eingebettet und abrufbar 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 echten Daten bereits 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 echte Daten haben kann.

Bei Auswertung der Netzwerkgenauigkeit auf Validierungs- sowie Testdaten aus je unterschiedlichen Quellen konnte eine deutliche Differenz zwischen synthetischen und echten Daten beobachtet werden. Während die synthetischen Daten nahezu perfekt identifiziert werden konnten und in den meisten Fällen lediglich geringe Abweichungen zu vermerken waren verlief die Erkennung der Dartpfeile in echten Bildern in vielen Fällen unter deutlich größerer Unsicherheit. Nicht nur wurden teilweise systematisch Markierungen als wahrscheinliche Kandidaten für Dartpfeile verzeichnet, sondern auch wurden die Farben der Felder teilweise nicht korrekt identifiziert.

Der systematische Fehler durch ein OOD-Training ist in den Auswertungen des Netzwerks manifestiert und nicht von der Hand zu weisen. Trotz dessen ist die Fähigkeit der Übertragung erlernter Charakteristiken synthetischer Daten auf reale Daten gezeigt worden.

### 5.3.3 Vergleich mit DeepDarts

Das für DeepDarts trainierte Modell beläuft sich auf eine Größe von ca. 6 Millionen Parametern; das in dieser Thesis trainierte Netzwerk umfasst etwa 10.5 Millionen Parameter. Die Modellgrößen weichen stark voneinander ab. 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 Anzahlen der Parameter als 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 massives Overfitting lediglich eine untere Schranke im Bezug auf die Parameterzahl stark limitierter Daten ermittelt werden. Die wesentlich diversere Datenlage dieser Thesis erfordert ebenfalls mehr Parameter in einem Netzwerk, um ähnliche Ergebnisse zu erzielen.



# Kapitel 6

## Fazit

- Forschungsfragen beantworten!

Hier steht das Fazit.

- es ist sehr schwer, alle Gegebenheiten in automatisierter Datenerstellung zu erfassen
- - Training auf echten Daten möglicherweise notwendig - erstellte Daten sind ausreichend, um ein neuronales Netz grundlegend auf Datenlage zu trainieren - - Inferenz nicht fehlerfrei möglich - - aber trotzdem zu weiten Teilen gute Ergebnisse - - Ausmerzung von Feinheiten notwendig, um zuverlässig zu funktionieren - im Gegensatz zu DD-System nicht overfitted
- - Inferenz auf neuen Daten grundlegend möglich - robustes Entzerren von Dartscheiben möglich - - kein Training notwendig - - orientiert an grundlegenden Gegebenheiten, nicht an Daten - - robust gegen Verdeckung der Dartscheibe - - klar strukturiert, Fehlerquellen präzise zu erkennen - - gutes System! \*pat pat\*

Fazit ausformulieren



# Kapitel 7

## Ausblick

Diese Arbeit stellt ein System vor, mit welchem ein automatisches Dart-Scoring durch die Verwendung herkömmlicher Computer Vision 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. Dieses Kapitel gibt einen Überblick über mögliche Ausgestaltungen und Verbesserungen dieser Systeme. Es wird begonnen mit dem Ausblick der Datenerstellung in Abschnitt 7.1, gefolgt von dem Ausblick der Pipeline zur Normalisierung in Abschnitt 7.2. Abschließend wird in Abschnitt 7.3 ein Überblick über weitere mögliche Arbeit an der Dartpfeilerkennung gegeben.

### 7.1 Ausblick der Datenerstellung

**Integration von physically-based rendering (PBR)** Ein wesentlicher Bestandteil der Datenerstellung ist die prozedurale Erstellung von Materialien. Diese ist aktuell nicht standardisiert 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 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 echten Aufnahmen zu erwarten sind. Die Aufnahmen können durch das gezielte Hinzufügen weiterer Objekte realistischer werden.

**Aussehen der Dartscheibe** Das Aussehen der Dartscheibe ist konform der Regelwerke der WDF und der Professional Darts Corporation gestaltet [5, 6]. Diese gibt Form und Farben der Dartscheibe mit etwaigen Toleranzen weitestgehend vor. Für die aktuelle Datenerstellung wurden lediglich diese und ähnliche Dartscheiben als Vorlagen in Betracht gezogen, wodurch Möglichkeiten weiterer Dartscheiben ausgeschlossen wurden. 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 Felder.

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 echten Dartscheiben hat sich jedoch gezeigt, dass diese Abstraktion von Logos durch Texte nicht ausreichend ist, um diese im nicht als Dartpfeile klassifizieren. Eine weitergehende Ausgestaltung dieser Bereiche der Dartscheibe ist daher notwendig.

## 7.2 Ausblick der Normalisierung

Der in dieser Thesis vorgestellte Algorithmus zur Lokalisierung und Normalisierung von Dartscheiben in beliebigen Bildern basiert ausschließlich auf Techniken herkömmlicher Computer Vision. 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 unter die Lupe genommen 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 Klassifikation 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 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 relevant sind. 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 zur denkbar. Die CNNs könnten die Aufgabe der Einordnung von Surroundings übernehmen, indem sie auf einer 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.

**Kompilierung des Algorithmus** Zur Implementierung des Algorithmus, wie er in dieser Thesis vorgestellt und ausgewertet wurde, 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-Quelltexts zur Optimierung von Ausführungszeiten [79, 80]. Durch diese Bibliotheken kann potenziell eine Optimierung der Laufzeit mit geringer Abänderung des vorhandenen Quelltextes erzielt werden.

- Kompilierung der CV-Pipeline - - entweder Cython [79] / Numba [80] oder Implementierung in kompilierter Sprache (C/C++/Rust)
- Ellipsen-Erkennung in CV einbauen ([81]) - - Möglichkeit zur Identifizierung von Ellipsen im Bild
- Vergleich mit DD auf unterschiedlichen Plattformen - - aktuell: eine Plattform - - möglich: Ausführung beides Systeme auf einem Mobiltelefon - - Hintergrund: System ist darauf ausgelegt, mobil genutzt zu werden

Ausblick CV

## 7.3 Ausblick der Dartpfeilerkennung

- neues Trainieren des Systems auf mehr echten Daten - - Arbeit spezifisch hinsichtlich Aufnahme und Annotation neuer Daten
  - Quantisierung der Netzwerke - - Verbesserung der Geschwindigkeit - - Verringerung der Ressourcennutzung (insbesondere hinsichtlich mobilem Deployment)
  - KI-Prediction auf Grundlage einer leeren Dartscheibe - - Kalibrierungs-Bild schließen und als Referenz nutzen - - Wenn bekannt ist, dass keine Dartpfeile auf Kalibrierungs-Bild vorhanden sind, ist die Wahrscheinlichkeit von Fehlklassifikationen des Hintergrundes geringer
  - neue Architekturen austesten - - Arbeit mit Fokus spezifisch auf Auswahl des Netzwerks
  - Warm Restarts der Learning Rate: [82]

Ausblick NN

**Eidesstattliche Erklärung**

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel, 25.05.2025

---



# 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
$\kappa \in \mathbb{R}$	Regulierungsfaktor der Harris Corner Detection
$\Lambda : ((\mathbb{R}^{3 \times 3})^2 \mapsto \mathbb{R})$	CV-Metrik zur Identifizierung der Ähnlichkeiten von Homographien
$\lambda_1 \in \mathbb{R}$	Hauptachse einer Ellipse
$\lambda_2 \in \mathbb{R}$	Nebenachse einer Ellipse
$(d_{\min}, d_{\max})$	Minimaler und maximaler Abstand der Kamera von der Dartscheibe
$(y_{\min}, y_{\max})$	Minimale und maximale Höhe der Kamera im Raum
$\mu_K$	Klassen-Metrik zur Bestimmung korrekter Klassen der Dartpfeile je Bild.
$\mu_P$	Positions-Metrik zur Bestimmung der Abweichungen der Dartpfeilpositionen.
$\mu_X$	Existenz-Metrik zur Bestimmung korrekter Anzahl der Dartpfeile je Bild.
$\phi_h$	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 $L_o$
$\phi_v$	Vertikaler Öffnungswinkel des Kamera-Space in die Höhe
$\sigma_{\text{noise}}$	Standardabweichung der Rausch-Augmentierung.
$\sigma_{\text{trans}}$	Standardabweichung der Translations-Augmentierung.
$\text{Id} \in \mathbb{R}^{3 \times 3}$	Identitäts-Transformation
$\text{lr}_0$	Initiale Learning Rate
$\text{lr}_{\min}$	Minimale Learning Rate
$\theta_{i,p} \in \mathbb{R}$	Winkel eines Pixels $p$ auf der Linie $i$ zum Mittelpunkt der Dartscheibe
$\varphi_i$	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
$A_{\text{cont}}$	Augmentierungsgewicht der Kontraständerung.
$A_{\text{rot}}$	Augmentierungsgewicht der Rotation.
$A_{\text{sat}}$	Augmentierungsgewicht der Sättigungsänderung.
$A_{\text{cha}, \{r,g,b\}}$	Augmentierungsgewichte für rot-, grün- unb blau-Kanäle.

$b_{\text{adj}}$	Maximale Helligkeitsänderung der Augmentierung.
$b_{\text{img}}$	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
$d_{\text{Kamera}}$	Abstand der Kamera zur Dartscheibe
$d_{i,p} \in \mathbb{R}$	Abstand einer Pixels $p$ auf einer Linie $i$ zum Mittelpunkt der Dartscheibe
$dx_{\max}$	Maximaler seitlicher Abstand der Kamera zum Dartscheibenmittelpunkt
$e \in \mathbb{N}$	Aktuelle Trainingsepochen
$e_{\text{warmup}} \in \mathbb{N}$	Anzahl an Warmup-Epochen für das Training
$f_{\text{lr}}$	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
$l_{\text{Kamera}}$	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
$M_{\text{scl}} \in \mathbb{R}^{3 \times 3}$	Skalierungsmaatrix
$M_{\text{shr}} \in \mathbb{R}^{3 \times 3}$	Scherungsmaatrix
$M_{\text{trans}} \in \mathbb{R}^{3 \times 3}$	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
$N_{\text{OP}}$	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
$P_{\max} \in \mathbb{R}^{\mathbb{F}}$	Kartesischer Punkt mit maximalen Linienüberschneidungen 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 $i$ .
$R \in \mathbb{R}$	Ausgabewert der Harris Corner Detection zur Bestimmung von Ecken

$s \in \mathbb{N}$	Indizes für Startlinien der Winkelentzerrung
$s_{\max} \in \mathbb{N}$	Maximale Breite bzw. Höhe eines Eingabebildes
$T$	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.
$w_{\text{lr}}$	Fenstergröße des Learning Rate-Schedules
$x, y \in \mathbb{N}$	Variable Koordinaten in einem kartesischen Koordinatensystem



# Abkürzungen

**CV** Computer Vision

**NMS** Non-Maximum-Suppression

**OOD** out-of-distribution

**PANet** Path Aggregation Network

**PBR** physically-based rendering

**PCS** Percent Correct Score

**WDF** World Dart Federation

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



# Literatur

- [1] Auto Darts. *Auto Darts - Official Website*. Accessed: 2024-10-23. 2024. URL: <https://autodarts.io>.
- [2] SCOLIA Technologies. *Scolia - Official Website*. Accessed: 2024-10-23. 2024. URL: <https://scoliadarts.com>.
- [3] William McNally u.a. *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>.
- [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](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](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 u.a. „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 u.a. „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 u.a. *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 u.a. *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 u.a. „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 u.a. „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 u. a. „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 u. a. *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](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](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 u. a. „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 u. a. „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 u. a. „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.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] 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>.
- [40] 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>.
- [41] 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.
- [42] Giovanni Capobianco u. a. „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>.
- [43] Irwin Sobel. „An Isotropic 3x3 Image Gradient Operator“. In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).
- [44] 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>.
- [45] Kris Kitani. *Harris Corners*. Zuletzt aufgerufen: 03.03.2025. URL: [https://www.cs.cmu.edu/~16385/s17/Slides/6.2\\_Harris\\_Corner\\_Detector.pdf](https://www.cs.cmu.edu/~16385/s17/Slides/6.2_Harris_Corner_Detector.pdf).

- [46] P. V. C. Hough. „Method and Means for Recognizing Complex Patterns“. US3069654A. Issued December 18, 1962. URL: <https://patents.google.com/patent/US3069654A/en>.
- [47] Richard Hartley und Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2. Aufl. Cambridge University Press, 2004, S. 37–44.
- [48] Gilbert Strang. *Introduction to Linear Algebra*. 4. Aufl. Wellesley, MA: Wellesley-Cambridge Press, Feb. 2009, S. 375–379.
- [49] Helder Araujo und Jorge Dias. „An Introduction to the Log-Polar Mapping“. In: *Proc. of Second Workshop on Cybernetic Vision* (Jan. 1996).
- [50] 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>.
- [51] Zhou Wang u. a. „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).
- [52] 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>.
- [53] 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>.
- [54] B. Spain. *Analytical Conics*. Dover Books on Mathematics. Dover Publications, 2007, S. 14. ISBN: 9780486457734. URL: <https://books.google.de/books?id=qHFXUAau5vcC>.
- [55] 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 u. a. Bd. 25. Curran Associates, Inc., 2012. URL: [https://proceedings.neurips.cc/paper\\_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf).
- [56] 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>.
- [57] Conor Sheehan, Ben Day und Pietro Liò. *Introducing Curvature to the Label Space*. 2018. arXiv: [1810.09549 \[cs.LG\]](https://arxiv.org/abs/1810.09549). URL: <https://arxiv.org/abs/1810.09549>.
- [58] Muhammad Hussain. „Yolov1 to v8: Unveiling each variant—a comprehensive review of yolo“. In: *IEEE access* 12 (2024), S. 42816–42833.
- [59] Muhammad Hussain. „Yolov5, yolov8 and yolov10: The go-to detectors for real-time vision“. In: *arXiv preprint arXiv:2407.02988* (2024).
- [60] 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.
- [61] 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.
- [62] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: [2408.15857 \[cs.CV\]](https://arxiv.org/abs/2408.15857). URL: <https://arxiv.org/abs/2408.15857>.
- [63] Tsung-Yi Lin u. a. „Focal loss for dense object detection“. In: *Proceedings of the IEEE international conference on computer vision*. 2017, S. 2980–2988.

- [64] Zhaohui Zheng u. a. „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.
- [65] Joseph Redmon u. a. „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.
- [66] Peiyuan Jiang u. a. „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>.
- [67] Ruizhi Zhang u. a. „A density-based oversampling approach for class imbalance and data overlap“. In: *Computers & Industrial Engineering* 186 (2023), S. 109747.
- [68] Ahmad S Tarawneh u. a. „Stop oversampling for class imbalance learning: A review“. In: *IEEE Access* 10 (2022), S. 47643–47660.
- [69] 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. URL: <https://doi.org/10.1186/s40537-019-0197-0>.
- [70] 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.
- [71] Jason Wang, Luis Perez u. a. „The effectiveness of data augmentation in image classification using deep learning“. In: *Convolutional Neural Networks Vis. Recognit* 11.2017 (2017), S. 1–8.
- [72] Agnieszka Mikolajczyk 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.
- [73] Qianqian Tong, Guannan Liang und Jinbo Bi. „Calibrating the adaptive learning rate to improve convergence of ADAM“. In: *Neurocomputing* 481 (2022), S. 333–356.
- [74] Ilya Loshchilov und Frank Hutter. „Decoupled weight decay regularization“. In: *arXiv preprint arXiv:1711.05101* (2017).
- [75] Ricardo Llugsí u. a. „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.
- [76] Sri Nurdiani u. a. „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>.
- [77] Juntang Zhuang u. a. „Adabelief optimizer: Adapting stepsizes by the belief in observed gradients“. In: *Advances in neural information processing systems* 33 (2020), S. 18795–18806.
- [78] Nefeli Gkouti u. a. „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).
- [79] Stefan Behnel u. a. „Cython: The best of both worlds“. In: *Computing in Science & Engineering* 13.2 (2011), S. 31–39.

- [80] 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>.
- [81] Modesto Medina-Melendrez u. a. „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>.
- [82] Ilya Loshchilov und Frank Hutter. „Sgdr: Stochastic gradient descent with warm restarts“. In: *arXiv preprint arXiv:1608.03983* (2016).