Institut für Informatik VII
Robotik und Telematik

Masterarbeit

Erkennung und Nachverfolgung von Balltrajektorien bei 2D-Fernsehaufnahmen im Tennis

Matthias Pröstler

März 2017

Erstgutachter: Prof. Dr. Andreas Nüchter
Zweitgutachter: Prof. Dr. Dietmar Seipel

Zusammenfassung

Im Sport wird immer weiter versucht seine Leistungsgrenze anzuheben. Dafür werden mehr und mehr computergestützte Hilfsmittel verwendet. Insbesondere im Tennis kann eine Taktikanalyse dem Trainer und dem Athleten bei der Spielvorbereitung des nächsten Gegners helfen. Diese ist allerdings bei manueller Durchführung sehr zeitintensiv und erfordert viel Fachwissen, weswegen eine automatische Ballerkennung bei 2D-Fernsehaufnahmen sehr interessant ist.

Infolgedessen beschäftigte sich diese Masterarbeit mit der automatischen Ballerkennung von 2D-Fernsehaufnahmen im Tennis, die Ballpositionen für eine Taktikanalyse liefern soll. Es wurde analysiert, inwiefern eine derartige Balldetektion bei Verwendung von Fernsehaufnahmen möglich ist. Insbesondere sollte der Einfluss einiger essentiellen Parameter, wie die Farbe des Spielfelduntergrundes, die Art der Kameraführung und die Qualität des Videos, auf die Ballerkennung untersucht werden.

Die Ergebnisse der Auswertung zeigten, dass eine Ballerkennung vor allem bei geeigneten Bedingungen bezüglich der eben genannten beeinflussenden Parameter durchaus vielversprechend ist. Unter diesen günstigen Umständen konnten 59.57% der untersuchten Ballwechsel eines Tennisspiels vollständig erkannt werden, sodass diese für die Taktikanalyse verwendet werden können. Die Analysen machten sichtbar, dass die Ballerkennung eine Precision von bis zu 88.25% und einen Recall von bis zu 80.64% erreichen konnte. Die Ergebnisse dieser Arbeiten stellten dar, dass eine auotmatische Ballerkennung bei 2D-Fernsehaufnahmen im Tennis durchaus umsetzbar ist, jedoch offenbarten sich auch Schwierigkeiten, welche für einen zuverlässigen Einsatz der Ballerkennung für Taktikanalysen berücksichtigt werden sollten. Vor allem sollte die Spielfelderkennung für alle Spielfeldumgebungen, aber auch die Identifikation des Balles in Spielernähe optimiert werden, um einen sinnvollen Einsatz der automatischen Ballerkennung für Taktikanalysen zu gewährleisten.

Inhaltsverzeichnis

Inhaltsverzeichnis I

Abbildungsverzeichnis IV

Tabellenverzeichnis V

Listings VI

1. Einleitung 1
- 1.1. Ziel der Arbeit . . . 2
- 1.2. Aufbau der Arbeit . . . 3

2. Theoretischer Hintergrund 4
- 2.1. Grundlagen der Bilderkennung . . . 4
  - 2.1.1. Darstellung von Bildern . . . 5
  - 2.1.2. Darstellung von Videos . . . 5
  - 2.1.3. Detektion . . . 7
  - 2.1.4. Identifikation . . . 12
- 2.2. Darstellungsarten und perspektivische Projektion . . . 12
- 2.3. Kalman Filter . . . 19
- 2.4. Clusterverfahren . . . 23
  - 2.4.1. Partitionierende Clusterverfahren . . . 23
  - 2.4.2. Hierarchische Clusterverfahren . . . 24
  - 2.4.3. Dichtebasierte Clusterverfahren . . . 25
- 2.5. Entwicklungsumgebung OpenCV . . . 28
  - 2.5.1. API Konzepte . . . 28
  - 2.5.2. Bibliotheken . . . 29

3. Forschungsstand und Hypothesen 34

4. Implementierung 37
- 4.1. Einlesen des Videos . . . 38
- 4.2. Erkennung des Spielfeldes . . . 39
  - 4.2.1. Erkennung von Konturen . . . 39
  - 4.2.2. Identifikation der Konturen . . . 45
  - 4.2.3. Optimierung der Spielfelderkennung . . . 46

4.2.4. Ermittlung der restlichen Spielfeldlinien . . . 47
4.3. Erkennung des Balles . . . 47
4.3.1. Festlegen des relevanten Suchbereiches . . . 49
4.3.2. Gruppieren von bewegten Bildpunkten . . . 54
4.3.3. Suche nach der neuen Ballposition . . . 56
4.4. Speicherung der Flugkurve des Balles . . . 62

5. Evaluierung **66**
5.1. Optimierung der Parameter . . . 67
5.1.1. Analyse des Canny Edge Detectors: min. und max. Thresholdwerte . . . 68
5.1.2. Analyse der Spielfelderkennung . . . 72
5.1.3. Ballerkennung: Analyse des Suchbereiches . . . 76
5.1.4. Ballerkennung: Analyse der Ballgrößen- und Distanzbestimmung . . . 80
5.1.5. Ballerkennung: Analyse der Parameter für die Ballidentifikation . . . 83
5.2. Auswertung der Ergebnisse . . . 86
5.2.1. Analyse der Spielfelder . . . 89
5.2.2. Analyse der Kameraführung . . . 94
5.2.3. Analyse der Videoqualität . . . 96
5.2.4. Analyse der erkannten Ballwechsel . . . 97
5.2.5. Analyse der Ausführungsgeschwindigkeit . . . 99

6. Zusammenfassung und Diskussion **101**
6.1. Zusammenfassung der Arbeit . . . 101
6.2. Diskussion der Ergebnisse . . . 104

A. Anhang **107**
A.1. Diagramme . . . 107
A.1.1. Analyse der Spielfelderkennung . . . 107
A.1.2. Ballerkennung: Analyse des Suchbereiches . . . 111
A.1.3. Analyse der Ballgröße und Distanzbestimmung . . . 137
A.1.4. Analyse der Parameter für die Ballidentifikation . . . 140

Literaturverzeichnis **IV**

Abbildungsverzeichnis

2.1. Sampling bei einer Videosequenz [13, S.8] 6
2.2. Gegenüberstellung eines Bildes mit Rauschen und nach Bearbeitung von diesem durch Smoothing mit einer $5\,x\,5$ Matrix eines Durchschnittfilters [25, S.31] 8
2.3. Gegenüberstellung eines Bildes auf dem der Canny Algorithmus angewandt wurde und die anschließende Dilatation von diesem 10
2.4. Beispiel einer Dilatation $X \oplus B$ [15, S.662] 11
2.5. Beispiel einer Erosion $X \ominus B$ [15, S.663] 12
2.6. Skizze einer Lochkamera und eine vereinfachte Darstellung einer Projektion [18, S.10] 13
2.7. Übersicht der Darstellungstypen von planaren Projektionen, nach [21] 14
2.8. Darstellung verschiedener Projektionen 15
2.9. Darstellung einer perspektivischen Projektion [2, S.201] 16
2.10. Skizze einer perspektivischen Projektion in seitlicher Ansicht [18, S.11] 16
2.11. Darstellung von zwei Projektionsebenen mit gleichem Projektionszentrum, nach [22] 18
2.12. Darstellung verschiedener Cluster mit unterschiedlichen Formen, Größen und Dichten [11, S.3] 23
2.13. Dichte-Erreichbarkeit bei dichtebasiertem Clustering, nach [6, S.419] 27

4.1. Übersicht des Programmablaufes 37
4.2. Differenzbild von zwei aufeinanderfolgenden Bildern eines Videos 38
4.3. Übersicht des Ablaufes bei der Felderkennung 40
4.4. Darstellung von Linien, die durch eine probabilistische Hough Line Transformation erkannt wurden 41
4.5. Darstellung aller erkannten Konturen des Bildes 42
4.6. Ablauf der Konturendetektion und dessen Optimierung 43
4.7. Darstellung der erkannten Kontur des Spielfeldes: kurze Linien (rot) aufgrund des Spielers und des Netzes 44
4.8. Ablauf der Identifikation des Spielfeldes 45
4.9. Ablauf der verbesserten Erkennung des Spielfeldes 47
4.10. Übersicht des Programmablaufes für die Ballerkennung 48
4.11. Suchbereich bei Aufschlagsituation 50
4.12. Beispiel für den eingeschränkten Suchbereich (grün) nach erkannter Ballposition 51
4.13. Übersicht über das Vorgehen den Suchbereich festzulegen 52

4.14. Beispiel für die Vergrößerung des Suchbereiches, wenn der Ball nicht im Suchbereich liegt 53
4.15. Suchbereich bei verlorener Ballposition in der Spielfeldmitte (Netz) 54
4.16. Übersicht über den Ablauf, um die bewegten Bildpunkte zu gruppieren 55
4.17. Übersicht über das Vorgehen bei der Suche der Ballposition 56
4.18. Darstellung der bewegten Bildpunkte als Cluster: lediglich der Arm und Fuß des unteren Spielers bewegt sich 59
4.19. Ballerkennung bei Aufschlagsituation - Aufschläger rechts oben: Ball wird vor dem Aufschlag noch erkannt 62
4.20. Ballerkennung bei Aufschlagsituation - Aufschläger rechts oben 65

5.1. Kantenerkennung in Abhängigkeit von min. und max. Thresholdwerten - Video 1 68
5.2. Kantenerkennung in Abhängigkeit von min. und max. Thresholdwerten - Video 2 70
5.3. Kantenerkennung in Abhängigkeit von min. und max. Thresholdwerten - Video 3 71
5.4. Darstellung einer Verwechselung des Balles mit einem Spieler 72
5.5. Analyse der Spielfelderkennung: ohne Optimierungen nach erfolgreicher Felderkennung 74
5.6. Analyse der Spielfelderkennung: mit Optimierungen anhand der Spielfeldecken . 75
5.7. Video 2: Ballerkennung - Analyse des Suchbereiches (Art = linear, Verschiebung = anhand Ballposition, Wiederholungsfaktor = 0.7) 77
5.8. Ballerkennung: Analyse des Suchbereiches (Art = linear, Verschiebung = anhand Ballposition, Wiederholungsfaktor = 0.7, Suchbereich = 58) 78
5.9. Ballerkennung: Analyse des Suchbereiches (Art = linear, Verschiebung = anhand Kalman, Wiederholungsfaktor = 0.9, Suchbereich = 58) 79
5.10. Ballerkennung: Analyse des Suchbereiches (Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9, Suchbereich=58) 80
5.11. Video 2: Ballerkennung - Analyse der Ballgröße (Historie) und Distanz (Ermittlung durch letzter Ballposition) 82
5.12. Video 3: Ballerkennung - Analyse der Ballgröße (Historie) und Distanz (Ermittlung durch letzter Ballposition) 83
5.13. Video 1: Ballerkennung - Analyse der Parameter für die Ballidentifikation (Berechnungsformel = exponentiell, Faktor für gültigen Bereich = 0.35) 85
5.14. Video 2: Ballerkennung - Analyse der Parameter für die Ballidentifikation (Berechnungsformel = exponentiell, Faktor für gültigen Bereich = 0.35) 86
5.15. Video 3: Ballerkennung - Analyse der Parameter für die Ballidentifikation (Berechnungsformel = exponentiell, Faktor gültiger Bereich = 0.35) 87
5.16. Ballerkennung: Ball wird nach Falschidentifikation wiedergefunden 87
5.17. Ballerkennung - Analyse der Spielfelder: blaues Spielfeld (Darstellung des Ballwechsels unter Angabe der Ausführungsdauer) 90
5.18. Darstellung des ersten Ballwechsels bei blauen Spielfeld 91
5.19. Ballerkennung - Analyse der Spielfelder: Sandplatz (Problem bei der Spielfelderkennung) 93

5.20. Ballerkennung - Analyse der Spielfelder: Rasenplatz (Problem: Spielfeld ist sehr nah am Bildrand und wird daher nicht komplett analysiert) 94
5.21. Ballerkennung - Analyse der Kameraführung: dynamische Aufnahmen 95
5.22. Ballerkennung - Analyse der Videoqualität: niedrige Videoqualität (Darstellung des Ballwechsels unter Angabe der Ausführungsdauer) 96

Tabellenverzeichnis

2.1. Ablauf des Kalman Filtervorgangs, nach [5] 22

5.1. Übersicht der Parameterkonfiguration (für die weitere Auswertung der Ballerkennung) 88
5.2. Ergebnisse der Ballwechselanalyse 99

Listings

2.1. DBSCAN Algorithmus 1: dbscan(O, ϵ, minPts) 26
2.2. DBSCAN Algorithmus 2: erweitereCluster(O, o, clusterID, ϵ, minPts) 26
2.3. API Konzept: Namespace 28

4.1. Speicherung der Flugkurve des Balles in einer XML-Datei 63

1. Einleitung

In vielen Sportarten steht durch ihre heutige Medienpräsenz zunehmend mehr Geld zur Verfügung. Dies wird zum einen durch Fernsehgelder zum anderen aber auch durch lukrative Werbemöglichkeiten für Unternehmen in Form von Sponsoring ermöglicht. Der Sport wird dadurch weiter professionalisiert und es stehen höhere Preisgelder zur Verfügung. Der Anreiz und auch der Druck in seiner Sportart erfolgreich zu sein steigt dadurch jedoch ebenfalls für den Sportler an. Dies führt zu Veränderungen und kontinuierlichen Optimierungen des Trainingsbetriebs, um den Sportler an seine Leistungsgrenze zu bringen. Die Informatik wird dabei im ansteigenden Maße als Hilfsmittel eingesetzt. Durch die fortschreitende Technik können Bewegungs- und Taktikanalysen von Spielern immer besser dargestellt werden. Diese Analysen sind allerdings bei manueller Durchführung sehr zeitintensiv, da laut Kovacs et al. [8] Tennisspiele zwischen einer und fünf Stunden andauern können, und benötigen Fachwissen des Trainers. Es muss wesentlich mehr Zeit als die absolute Spieldauer investiert werden, um beispielsweise eine gute Taktikanalyse zu erstellen. Hierbei ist es nicht ausreichend nur das Spiel zu betrachten. Die Spielsequenzen müssen meist verlangsamt oder mehrfach angesehen werden, um die wichtigsten Anhaltspunkte für eine Bewegungs- oder Taktikanalyse herauszufinden. Aus diesen Gründen ist ein automatisches Verfahren zur Videoanalyse von großem Interesse.

Diese Masterarbeit beschäftigt sich mit der automatischen Ballerkennung bei Tennisspielen, sodass der Ballwechsel anhand von 2D-Fernsehaufnahmen ersichtlich wird und für weitere Analysen zur Verfügung steht. Ein Programm zur Taktikanalyse von Ballwechseln im Tennis existiert bereits in einer ersten Version am Lehrstuhl für Algorithmen, Komplexität und wissensbasierte Systeme an der Universität Würzburg und wurde von Herrn Prof. Dr. Seipel entwickelt. Dieses basiert auf Prolog und kann durch logische Verknüpfungen Beziehungen zwischen verschiedenen Schlägen herstellen. Es ermöglicht das Schlagverhalten von Spielern zu untersuchen, sowie Stärken und Schwächen der Spieler aufzudecken. Solch eine Taktikanalyse ist für die Spielvorbereitung sowohl für den Athleten als auch für den Trainer sehr hilfreich, um sich auf den entsprechenden Gegner vorzubereiten. Das Programm von Herrn Prof. Dr. Seipel benötigt zur weiteren Optimierung möglichst viele Ballpositionsdaten von Tennisspielen. Diese mussten bisher manuell erfasst und in die Datenbank eingetragen werden. Für eine bessere Anwendbarkeit des Programms sollen die Daten der Ballwechsel automatisiert aus Videodateien extrahiert werden, um Spiele leichter analysieren zu können.

1.1. Ziel der Arbeit

Diese Masterarbeit hat das Ziel ein Programm zu entwickeln, welches anhand von 2D-Fernsehaufnahmen Ballwechsel im Tennis automatisch erkennt und diese abspeichert, sodass sie in eine Taktikanalyse-Datenbank importiert werden können. Dabei soll grundsätzlich untersucht werden, ob solch eine automatische Ballerkennung auf Tennisfeldern mit hohem Kontrast zwischen Spielfeldbelag und Tennisball möglich ist. Den vermutlich höchsten Kontrast bietet ein blauer Hartplatz in Kombination mit einem gelben Tennisball, weshalb diese Situation zuerst betrachtet werden soll. Im Anschluss daran soll untersucht werden, wie sich die Ballerkennung bei davon abweichenden Spielfeldbelägen verhält. Zu klären ist, wie sicher sich die Ballposition dabei bestimmen lässt. Es ist allerdings nicht Teil dieser Arbeit die Ballerkennung für jeden Spielfelduntergrund zu gewährleisten.

Primäres Ziel soll die Implementierung der Ballerkennung bei statischen Videosequenzen in Vogelperspektive[1] sein. Das Spielfeld muss daher zum einen von oben und hinter dem Feld betrachtet werden, um das ganze Spielfeld immer im Blick behalten zu können. Auch können dadurch andere Videosequenzen wie Werbepausen, Wiederholungen eines Ballwechsels und Nahaufnahmen erkannt werden, die sich durch andere Perspektiven auf das Spielfeld auszeichnen. Zum anderen sollen Videoaufnahmen mit Zoom-Effekten sowie solche mit Schwenkbewegung der Kamera vorerst vernachlässigt werden. Es soll aufgezeigt werden, wie zuverlässig der Ballwechsel bei statischen Videosequenzen erkannt werden kann. Im späteren Verlauf soll analysiert werden, inwieweit die Implementierung auch für dynamische Videosequenzen eingesetzt werden kann.

Im Allgemeinen sind für die Taktikanalyse der Prologdatenbank vor allem die Kontaktpunkte des Balles mit dem Boden und der Treffpunkt des Balles mit dem Schläger des jeweiligen Spielers wichtig. Folglich liegt das Hauptaugenmerk bei der Ballerkennung in der Registrierung dieser Punkte. Als Toleranzbereich wird dabei festgelegt, dass der Ball nicht in jedem Bildframe exakt erkannt werden muss, lediglich die grundsätzliche Flugkurve des Balles soll nachverfolgt werden können. Vernachlässigt werden kann die exakte Position des Balles beim Schlagzeitpunkt und die Flugkurve des Balles, wenn sich dieser direkt vor dem Spieler befindet, da in diesen Situationen der Ball mit dem Spieler verschmilzt und nicht ohne weiteres erkannt werden kann. Es ist für die Taktikanalyse nicht gravierend, wenn die Erkennung der Ballposition nicht auf den Zentimeter genau stattfindet. Wurde der Ball erkannt, sollen die Position und der entsprechende Zeitpunkt abgespeichert werden. Die Koordinaten werden dabei in Pixelwerten (x und y) angegeben. Ziel ist es den Verlauf des Ballwechsels nachverfolgen zu können.

[1] Vogelperspektive: Ansicht hinter und überhalb des Spielfeldes, sodass eine Sicht von oben auf das Spielfeld möglich ist. Bei Tennisspielen ist die Kamera dabei normalerweise hinter der Grundlinie, sowie in der Mitte des Spielfeldes positioniert.

1.2. Aufbau der Arbeit

Diese Arbeit ist in sechs Kapitel gegliedert. Im Anschluss an diese Einleitung, in der das Thema vorgestellt und die Ziele, sowie der Aufbau erläutert werden, wird in Kapitel 2 auf grundlegendes Hintergrundwissen und allgemeine Theorien, die die weitere Arbeit betreffen, eingegangen.

In Kapitel 3 wird der aktuelle Forschungsstand dargestellt. Darin wird auf themenverwandte Arbeiten eingegangen, deren Ergebnisse vorgestellt und die Schwierigkeiten, auf die die Verfasser dabei gestoßen sind, erläutert. Aus diesen Ergebnissen und Problemen werden anschließend Hypothesen aufgestellt, auf die im weiteren Verlauf der Arbeit eingegangen wird.

Das vierte Kapitel enthält die eigentliche Umsetzung und Implementierung dieser Arbeit. Auf der einen Seite werden die Ideen und Konzepte beschrieben, die genutzt werden können, um Ballwechsel in Fernsehaufnahmen zu erkennen. Auf der anderen Seite wird dabei genauer auf Besonderheiten bei der Umsetzung eingegangen, sowie auftretende Probleme geschildert.

Im darauffolgenden Kapitel 5 wird auf die Ergebnisse dieser Arbeit eingegangen und anhand von Statistiken veranschaulicht. Darüber hinaus werden die Hypothesen aus Kapitel 3 wieder aufgegriffen und ihre Umsetzbarkeit erörtert.

Das letzte Kapitel fasst die Ergebnisse zusammen, interpretiert sie und gibt einen Ausblick auf mögliche Strategien zur weiteren Optimierung.

2. Theoretischer Hintergrund

In diesem Kapitel werden für diese Arbeit wichtige Hintergründe und Grundlagenwissen erläutert, sodass die Terminologie und weitere Beschreibungen in dieser Arbeit verständlich sind. Da sich diese Arbeit mit der Ballerkennung bei 2D Fernsehaufnahmen von Tennisspielen beschäftigt, liegt das Hauptaugenmerk auf der Bildverarbeitung. Die Grundlagen dazu sollen neben der perspektivischen Geometrie beschrieben werden, damit ersichtlich wird, wie Videos vom Computer dargestellt und verarbeitet werden können. Ebenso wird das dafür verwendete Framework OpenCV[1] vorgestellt, das bereits viele Funktionalitäten bereitstellt, um Bilder zu untersuchen. Im Rahmen dieser Arbeit wird mit Hilfe verschiedener Verfahren der Ball erkannt und dessen Flugkurve nachverflogt. Dafür sind zum einen Clusterverfahren notwendig, sowie der Kalman Filter, die es erlauben die Objekte in Form von Bildpunkten zu erkennen und nachzuverfolgen. Das Basiswissen dazu soll in den nächsten Abschnitten vorgestellt werden.

2.1. Grundlagen der Bilderkennung

Die Bilderkennung beschäftigt sich mit der Verarbeitung von Bildern, sowie dem Verstehen des Dargestellten. Dies beschreibt Sonka und Hlavac als Computer Vision [15]. Dort werden zwei grundlegende Phasen unterschieden:

- Detektion
- Identifikation

Bevor auf diese Phasen genauer eingegangen wird, soll kurz auf die Aussage von Sonka und Hlavac Bezug genommen werden, dass Computer Vision ein schwieriges Thema in der Informatik ist [15, S.3]. Sie beschreiben in ihrem Buch, dass Bilder alle 3D Informationen der realen Welt in 2D abspeichern, wodurch ein enormer Informationsverlust auftritt. Weiterhin können kleine Störungen in den Bildaufnahmen genauso wie Objekte, die aufgrund verschiedener Lichtverhältnisse oft nicht vom Hintergrund zu unterscheiden sind, die Bilderkennung erschweren. Eine weitere Aussage, die hier von Sonka und Hlavac aufgegriffen werden soll, ist dass besonders bei Videos sehr große Datenmengen analysiert werden müssen, um die notwendigen Informationen zu extrahieren. Sofern die Bilder und die darauf gespeicherten Umrisse der Objekte erkannt wurden, stellt sich die Frage, um was für ein Objekt es sich überhaupt handelt. Dafür können laut [15, S.

[1] OpenCV: Abkürzung für Open Computer Vision Library. Sie ist eine Open Source Bildverarbeitungsbibliothek auf die in Kapitel 2.5 genauer eingegangen wird.

381] Verfahren der Künstlichen Intelligenz helfen, um mit verschiedenen Methoden die erkannten Umrisse zu klassifizieren. Um diese Probleme besser nachvollziehen zu können, soll in den nächsten Abschnitten zuerst die Art und Weise, in der Bilder und Videos von Computern dargestellt werden, grob beschrieben werden, bevor anschließend die Detektion und Identifikation in der Bildverarbeitung erläutert wird.

2.1.1. Darstellung von Bildern

Bilder bestehen aus vielen Bildpunkten (Pixel), die im zweidimensionalen Raum durch x und y Koordinaten dargestellt werden. Die Qualität der Bilder wird mit einer Auflösung beschrieben. Typische Bildauflösungen sind SD (Standard Definition) mit einer Breite x Höhe Verhältnis von $720\,x\,576$ Pixel, 720p HD (High Definition) mit $1280\,x\,720$ und 1080p HD mit $1920\,x\,1080$ Pixel. [13, S.19]

Beim Erstellen von Bildern können Sensoren der Kamera diese als kontinuierliche Funktion $f(x, y)$ darstellen. Computer stellen Bilder in Rastern dar, wie beispielsweise Matrizen. Durch das Sampling Verfahren, auf das hier nicht weiter eingegangen wird, werden die kontinuierlichen Bildsignale auf eine Matrix abgebildet. Diese entspricht den zuvor genannten Auflösungen. Weiterhin wird das kontinuierliche Signal durch Quantisierung auf sein digitales Pendant abgebildet, sodass pro Pixel ein digitaler Wert des Bildrasters abgespeichert werden kann. Ein solcher Pixel besteht, je nach verwendetem Farbmodell, aus 8- bis 32-bit wodurch die Farbe oder Intensität des Bildes dargestellt wird. Bei Verwendung des RGB Farbmodells besteht ein Pixel aus 24-bit, die drei Farbkanäle - rot, grün und blau - mit je 8-bit wiederspiegeln. Bei Graustufenbildern stellt der Wert des Pixels die Helligkeit des Bildes, also die Intensität, dar. [15, S.14f] [14, S.9]

2.1.2. Darstellung von Videos

Videos bestehen aus einer Aneinanderreihung von einzelnen Bildern. Wie bei Bildern wird durch Sampling ein analoges Videosignal in ein digitales umgewandelt. Dabei ist, wie in Abbildung 2.1 zu sehen ist, die räumliche (spatial) Abtastung für die Zuweisung der Bildpunkte innerhalb eines Bildes zuständig. Bei der zeitlichen (temporal) Abtastung werden dem Video aufeinanderfolgende Bilder, die auch Frames genannt werden, zugewiesen. Die Anzahl der Frames per Second (FPS) bestimmt, ob das Video flüssig wahrgenommen wird. Üblicherweise werden Videos mit 25 oder 30 FPS wiedergegeben [13, S.7ff]. Dadurch kann ein Video flüssig dargestellt werden, was allerdings einen enormen Speicheraufwand mit sich bringt. Bei 30 FPS und einer 1080p HD Auflösung von $1920\,x\,1080$ Pixeln sowie einer Farbtiefe von 8Bit (RGB) ergibt sich ein Speicherbedarf von $177.98 MByte$ pro Sekunde. Dies ist ein enormer Speicheraufwand, weswegen die Rohdaten dieser Videos komprimiert werden, um die Dateigröße zu reduzieren.

Dafür werden **Video CODECs** (enCOder/DECoder) verwendet. Sie ermöglichen die Komprimierung des Videos mit einem Encoder und die anschließende Wiedergabe mit Hilfe eines Decoders. Bei der Komprimierung werden beispielsweise redundante Daten im Video entfernt. Solch überflüssige Informationen könnten der gleiche Hintergrund in aufeinanderfolgenden Bildframes

oder auch im selben Bildframe nebeneinanderliegende Bildpunkte sein, da diese mit hoher Wahrscheinlichkeit ähnlich sind. Hier gibt es je nach CODEC unterschiedliche Ansätze überflüssige Informationen zu entfernen. Der Decoder stellt aus dem komprimierten Video das Ausgangsvideo wieder her. Sofern das rekonstruierte Video mit dem Ursprungsvideo übereinstimmt, handelt es sich um eine verlustfreie Komprimierung. Kann das Ursprungsvideo hingegen nicht mehr vollständig hergestellt werden, ist die Komprimierung verlustbehaftet. Verlustbehaftete Komprimierungen erreichen allerdings eine bessere Komprimierung als eine verlustfreie Komprimierung, wodurch sie, trotzt Einbußen bei der Bildqualität, Anwendung finden. CODECs haben das Ziel Videos möglichst gut zu komprimieren ohne dabei die Qualität zu vermindern. Diese Ziele stehen normalerweise in Konflikt zueinander. [13, S. 25f]

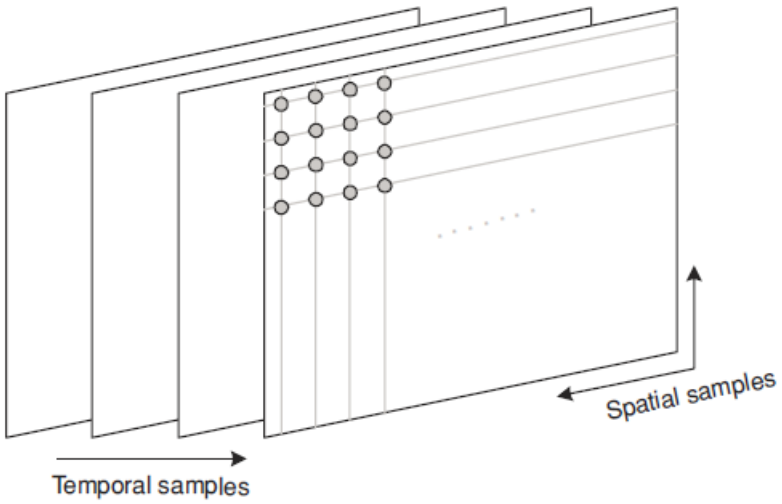


Abbildung 2.1.: Sampling bei einer Videosequenz [13, S.8]

Nachdem diese Arbeit Fernsehaufnahmen von Tennisspielen untersuchen soll, werden im Folgenden die typischen Fernsehformate beschrieben. Dabei soll lediglich auf die zwei relevantesten Standards **National Television Systems Committee** (NTSC) und **Phase Alternating Line** (PAL) eingegangen und dessen Eigenschaften kurz beschrieben werden.

Der analoge schwarz/weiß NTSC Standard wurde von 1941 bis 2009 vorwiegend in Amerika verwendet. Er wurde mit 525 horizontalen Bildzeilen definiert, wovon lediglich 480 beziehungsweise 486 verwendet wurden. Da in Amerika die Stromversorgung mit einer Frequenz von $60Hz$ arbeitet, wurden für die flüssige Darstellung der Videos 60 Halbbilder pro Sekunde festgelegt, was 30 verschachtelte Frames pro Sekunde wiederspiegelt. Dies bedeutet, dass die Informationen von 30 Bildframes aufgetrennt werden, sodass diese in 60 Frames pro Sekunde übertragen werden können. Als 1964 das Farbfernsehen aufkam, wurden dem NTSC Standard Farben hinzugefügt, allerdings stand dafür nicht mehr viel Platz zur Verfügung, sodass die Farbauflösung von NTSC nur sehr schwach ist. Weiterhin wurde in diesem Zusammenhang die Framerate um 0.1 Prozent auf 29.97 FPS vermindert.

Seit 1967 ist PAL der farbkodierte Standard in Europa. Ähnlich wie in Amerika wurden die

Frameraten an die Frequenz der Elektrizität gekoppelt. Diese liegt in Europa bei $50Hz$, weswegen der PAL Standard mit 50 Halbbilder pro Sekunde und damit 25 Frames pro Sekunde läuft. Er verwendet hingegen 625 horizontale Bildzeilen von denen 576 verwendet werden. Dadurch kann PAL genauso viele ($576*25 = 14.440$) Informationen pro Sekunden wie NTSC ($480*30 = 14.440$) übertragen. Aufgrund dieser Unterschiede in den Frameraten und der Anzahl der horizontalen Bildzeilen sind die beiden Standards nicht kompatibel zueinander und können nur mit Verlusten in das jeweils andere umgewandelt werden.

Analogen Standards wurden in den letzten Jahren zunehmend durch digitale abgelöst, um High Definition Television (HDTV) zu ermöglichen. In den USA ist dies der Standard des Advanced Television Standards Committee (ATSC). In Europa wird der Digital Video Broadcasting (DVB) Standard verwendet. [20, S.77f]

2.1.3. Detektion

Die Bilderkennung beschäftigt sich damit Objekte in Bildern wahrzunehmen und zu identifizieren. Um Objekte erkennen zu können, werden die Bilder häufig aufbereitet und damit eine möglicherweise fehlerhafte Darstellung der Bilder korrigiert oder die relevanten Informationen des Bildes für die entsprechende Bilderkennungsaufgabe stärker herausgehoben. Hierbei gibt es zwei wesentliche lokale Vorgehensweisen zur Verarbeitung der Bilder.

Zum einen das Smoothing, bei dem das Bild geglättet wird, um Störungen im Bild, in Form von kleinen Abweichungen oder fehlerhaften Pixelwerten, zu unterdrücken. Zum anderen gibt es Gradienten[2] Funktionen die Änderungen im Bildverlauf (Kanten) erkennen können und diese deutlicher darstellen. Im Anschluss soll auf einige Methoden zur Detektion eingegangen werden. [15]

Smoothing

Das Smoothing (Glättung) ist eine häufig angewandte Technik, um Rauschen aus Bildern zu entfernen, sodass weniger Störungen bei der Detektion auftreten. Hier wird für jeden Pixel (x, y) der Durchschnitt für die im Umkreis liegenden Pixelwerte berechnet und als neuer Pixelwert angenommen. Dies ermöglicht es, kleine Veränderungen des Bildes und Signalstörungen einzelner Pixelwerte aufzubereiten. Bei größeren Veränderungen ist diese Methode allerdings nicht erfolgreich, da diese zu stark in den Durchschnitt, der im Umkreis liegenden Pixel, mit eingehen und damit den neuen Pixelwert verfälschen. Das Glätten der Werte (x, y) erfolgt nach Matrix h bei einer Filtergröße von $n\,x\,n$. Diese Matrix ist in Gleichung 2.1 beschrieben.

[2] Gradienten: sind lokal zu berechnende Merkmale in Bildern [14, S.9f].

$$h(x,y) = \frac{1}{n^2}\begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} \tag{2.1}$$

In Abbildung 2.2(a) ist ein Bild von Lena dargestellt, das in der Bildverarbeitung häufig zur Verbildlichung von Rauschen benutzt wird. Im rechten Bild wurde das Smoothing mit einer $5\,x\,5$ Matrix angewandt und dadurch das Rauschen abgeschwächt.

(a) verpixeltes Bild (b) geglättetes Bild

Abbildung 2.2.: Gegenüberstellung eines Bildes mit Rauschen und nach Bearbeitung von diesem durch Smoothing mit einer $5\,x\,5$ Matrix eines Durchschnittfilters [25, S.31]

Kantenerkennung

In der Bildverarbeitung wird von Kanten gesprochen, wenn sich der Wert von aufeinanderfolgenden Pixeln rasch verändert. Kanten sind für die Bildwahrnehmung wichtig. Sie werden durch ihre Stärke und Orientierung ausgedrückt. Die Kantenstärke und -orientierung eines Bildpunktes wird dabei durch Gradienten in diesem Punkt beschrieben. Wie Salmen in seiner Dissertation [14] beschreibt sind Gradienten lokal zu berechnende Merkmale in Bildern. Der Gradient kann pro Pixel mit Hilfe der umliegenden Pixel ermittelt werden. In einem Bild I gibt es horizontale g_x und vertikale Gradienten g_y, die sich anhand folgender Formeln berechnen lassen:

$$g_x(x,y) = I(x-1,y) - I(x+1,y) \tag{2.2a}$$
$$g_y(x,y) = I(x,y-1) - I(x,y+1) \tag{2.2b}$$

Die Kantenstärke E wird aus der Länge der Hypotenuse des von den Gradienten g_x und g_y aufgespannten Dreiecks bestimmt:

$$E(x,y) = \sqrt{g_x^2(x,y) + g_y^2(x,y)} \tag{2.3}$$

Die Richtung des Gradienten $\alpha(x,y)$ wird durch folgende Gleichung ermittelt, vorausgesetzt $g_x(x,y) \neq 0$:

$$\alpha(x,y) = \arctan \frac{g_y(x,y)}{g_x(x,y)} \tag{2.4}$$

Anschließend wird der Canny Edge Algorithmus beschrieben, mit dessen Hilfe die Kantenerkennung realisiert werden kann, sowie Methoden die dies unterstützen.

Canny Edge Algorithmus:

Ein spezieller Ansatz zur Kantenerkennug ist der Canny Algorithmus, der auf Gradienten basiert. Angewandt auf ein Bild, liefert er als Ergebnis für jeden Pixel eine 1, wenn dort eine Kante, und eine 0, wenn dort keine Kante ausgemacht wurde. Er gibt folglich ein Binärbild zurück. Der Algorithmus wurde mit dem Ziel entwickelt drei Kriterien zu erfüllen.

- **Erkennung:** Erfassung aller wichtigen Kanten, aber keiner falschen Kanten
- **Lokalität:** Minimierung der Distanz zwischen gefundenen und tatsächlichen Kanten
- **Ansprechverhalten:** Vermeidung von Mehrfacherkennungen von Kanten

Nachfolgend soll der Algorithmus in vereinfachter Form wiedergegeben werden:

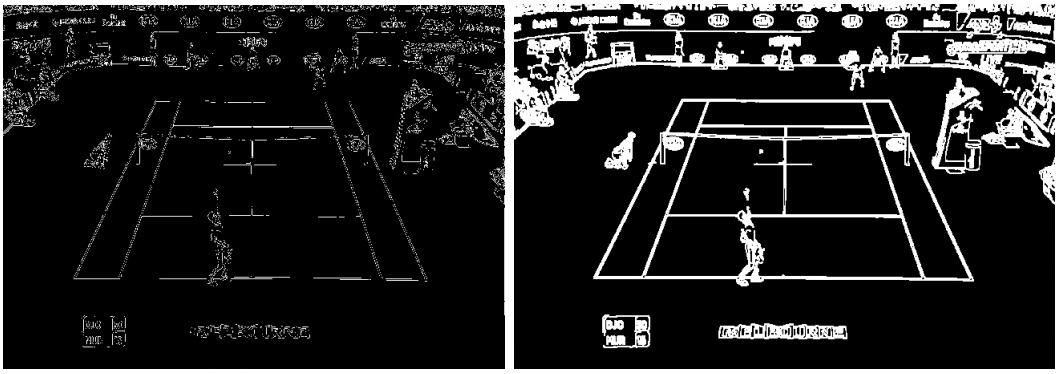
1. **Glättung:** Unterdrückung von Rauschen mit Hilfe des Gaußfilters
2. **Kantendetektion:** Berechnung von Kantenstärken und -richtungen
3. **Unterdrückung von nicht Maxima:** Akzeptanz nur von lokalen Maxima der Kantenstärken
4. **Hysterese:** Unterdrückung von unwichtigen Kanten durch Bilden von Schwellwerten

In Abbildung 2.3 ist ein Bild eines Tennisfeldes abgebildet. Auf dem linken Bild wurde der Canny Algorithmus auf das Ursprungsbild angewandt. Dort sind nur noch die Kanten mit maximaler Kantenstärke und die, die den Schwellwerten entsprechen, zu erkennen. Im rechten Bild wurden nach Anwendung des Canny Algorithmus die Kanten mit einer Dilatation[3] verbessert, indem Lücken im Kantenverlauf geschlossen wurden. Genaueres zur Dilatation folgt auf der nächsten SeiteTODO.

Thresholding:

Ein weiteres Verfahren für die Detektion von Objekten in Bildern ist das Thresholding. Dies ist ein sehr einfaches Verfahren, besonders bei Verwendung von Graustufenbildern. Es überprüft jeden Punkt im Ursprungsbild U darauf, ob ein bestimmter Schwellwert T eingehalten oder

[3]Dilatation: verstärkt die Struktur von Objekten



(a) Bild nach Canny Algorithmus (b) Bild nach anschließender Dilatation

Abbildung 2.3.: Gegenüberstellung eines Bildes auf dem der Canny Algorithmus angewandt wurde und die anschließende Dilatation von diesem

überschritten wird. Bei Überschreitung des Schwellwertes wird im Ausgabebild A der Wert des Punktes auf 1 gesetzt. Wird der Schwellwert unterschritten, ist das Ergebnis an dieser Stelle 0. Dies wird in nachfolgender Formel beschrieben:

$$A(x,y) = \begin{cases} 1, & wenn\ U(x,y) \geq T \\ 0, & wenn\ U(x,y) < T \end{cases} \tag{2.5}$$

Thresholding ist nicht rechenintensiv und daher vergleichsweise schnell. Es wird häufig verwendet, kann allerdings bei globaler Anwendung nicht zwischen Objekten und Hintergrund unterscheiden. Daher gibt es adaptives Thresholding, bei der der Schwellwert je nach Region des Bildes angepasst wird. [15, S.176]

Morphologische Operatoren:

Neben des Thresholding gibt es noch eine weitere unterstützende Methode zur Kantenverbesserung. Dabei handelt es sich um morphologische Operatoren, die dazu dienen die Struktur und Form von Objekten zu analysieren und zu beeinflussen. Diese werden oft eingesetzt, wenn die Form des Objektes eine Rolle spielt, wie zum Beispiel bei der Erkennung von Bewegungen oder der Analyse von Dokumenten. In diesen Fällen werden sie unter anderem für die Bildverarbeitung in Form von Filterung rauschbehafteter Kanten verwendet. Morphologische Operatoren unterstützen aber auch ein besseres Identifizieren von Objekten durch Ausdünnung oder Kräftigung der Kanten der Objekte. Um diese zu realisieren, läuft ein Strukturelement B über das Bild und bearbeitet dieses je nach festgelegter Operation. Typische morphologische Operatoren sind die Dilatation und die Erosion. [15, S.657ff]

Die **Dilatation** verstärkt die Struktur von Objekten, indem eine Vektoraddition $\oplus$ zwischen dem Bild X und dem Strukturelement B durchgeführt wird. Das Ergebnis e der Dilatation $X \otimes B$

ist ein neuer Vektor. Dieser wird durch alle möglichen Kombinationen der beiden Vektoren per Vektoradditionen ermittelt. [15, S. 661f]

$$X \oplus B = \{\vec{e} = \vec{x} + \vec{b}, \quad f\ddot{u}r\ \vec{x} \in X und\ \vec{b} \in B\} \tag{2.6}$$

In Abbildung 2.4 ist ein Beispiel einer Dilatation zu sehen. Die Vektoren besitzen folgende Werte:
$X = \{(1,0),(1,1),(1,2),(2,2),(0,3),(0,4)\}$
$B = \{(0,0),(1,0)\}$
$X \oplus B = \{(1,0),(1,1),(1,2),(2,2),(0,3),(0,4),(2,0),(2,1),(2,2),(3,2),(1,3),(1,4)\}$

Es ist zu erkennen, dass durch Dilatation mit diesem Strukturelement, die Objektstruktur um einen Pixel auf der rechten Seite verstärkt wird.

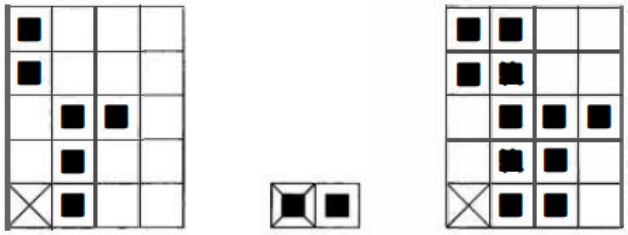


Abbildung 2.4.: Beispiel einer Dilatation $X \oplus B$ [15, S.662]

Im Gegensatz zur Dilatation dünnt die **Erosion** die Objektstruktur durch eine Vektorsubtraktion aus. Sie wird nach folgender Formel berechnet:

$$X \ominus B = \{\vec{e} = \vec{x} + \vec{b}, \quad f\ddot{u}r\ \vec{e} \in X\ und\ f\ddot{u}r\ jedes\ \vec{b} \in B\} \tag{2.7}$$

Auch hierfür ist in Abbildung 2.5 ein Beispiel dargestellt, das durch folgende Vektoren repräsentiert wird.
$X = \{(1,0),(1,1),(1,2),(0,3),(1,3),(2,3),(3,3),(1,4)\}$
$B = \{(0,0),(1,0)\}$
$X \oplus B = \{(0,3),(1,3)(2,3)\}$

Das **morphologische Opening und Closing** beschreibt Kombinationen aus Erosion und Dilatation. Diese stellen keine inversen Transformationen dar, sodass durch Anwendung der Erosion auf einem dilatierten Bild das Ursprungsbild nicht wiederhergestellt wird. Sie haben beide besondere Eigenschaften Bilder zu beeinflussen. [15, S. 665f]

Das morphologische Closing verbindet Objekte die nahe beieinanderliegen beziehungsweise füllt Löcher, je nachdem wie groß das Strukturelement B ist. Das morphologische Closing ist definiert als:

$$X \bullet B = (X \oplus B) \ominus B \tag{2.8}$$

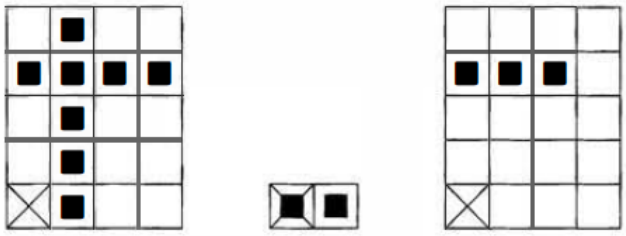


Abbildung 2.5.: Beispiel einer Erosion $X \ominus B$ [15, S.663]

Das morphologische Opening hingegen führt zu einer Vereinfachung des Ausgangsbildes und gibt es mit weniger Details zurück. Es ist folgendermaßen definiert:

$$X \circ B = (X \ominus B) \oplus B \tag{2.9}$$

Diese Operatoren können alle dazu verwendet werden, um die Objekte in Bildern besser zu erkennen. Aus diesem Grund werden sie im weiteren Verlauf dieser Arbeit zur Kantendetektion eingesetzt.

2.1.4. Identifikation

Die Identifikation von Bildern beschäftigt sich damit, den durch die Detektion wahrgenommenen Objekten eine Bedeutung zu zuweisen. Dies ist eine schwere Aufgabe, da sie sehr vom Anwendungsfall abhängt. Grundsätzlich wird versucht aus den erkannten Objekten Merkmale zu extrahieren, die durch Verfahren der Künstlichen Intelligenz klassifiziert werden können. Durch diese Einteilung kann den Objekten eine Bedeutung zugewiesen werden. Hierbei werden beispielsweise Verfahren wie neuronale Netze, einfache Klassifizierung oder statistische Verfahren angewandt. [15, S.450]

2.2. Darstellungsarten und perspektivische Projektion

Die digitale Fotographie hat die Aufgabe 3D Objekte in ein 2D Bild umzuwandeln. Dies bedeutet immer auch einen Informationsverlust und kann auf unterschiedliche Arten realisiert werden. Grundsätzlich wird eine Abbildung von Bildpunkten eines höher dimensionierten Raumes (z.B. 3D) zu einem niedriger dimensionierten Raum (z.B. 2D) **Projektion** genannt. Das darzustellende Objekt wird durch gedachte **Projektionsstrahlen** (engl. projectors) auf eine Ebene abgebildet. Diese wird **Projektions- oder Bildebene** genannt und enthält das projizierte Bild. Die Projektionsstrahlen sind Abbildungslinien, die sich alle im **Projektionszentrum** (engl. center of projection COP) schneiden. Dieser Punkt wird auch **Kamerapunkt oder Augpunkt** genannt. [21]

Ein sehr einfaches Modell für die Abbildung von Bildern ist die **Lochkamera** (engl. pinhole camera). Sie ist in Abbildung 2.6 als Skizze dargestellt. Jeder Punkt des Objektes wird durch

ein kleines Loch auf die Bildebene abgebildet. Das Loch wird dabei unendlich klein gewählt, sodass ausgehend von einem Bildpunkt nur ein Projektionsstrahl das Loch durchqueren kann und dadurch ein scharfes Bild dargestellt wird. Dies ist allerdings auch ein Nachteil der Lochkamera, da durch das kleine Loch nur wenig Licht in die Kamera fällt. Weiterhin kann die Kamera das Bild nicht in Größe und Winkel verändern, sondern nur so darstellen, wie die Lichtstrahlen durch das Loch scheinen. Aus diesem Grund besitzen alle Kameras Linsen, die genau diese Schwachstellen beheben. Sie fangen zum einen mehr Licht auf und können durch einen Fokus auf die entsprechende Kameraposition angepasst werden. In der rechten Abbildung von 2.6 ist zu sehen, dass das Bild allerdings auf dem Kopf stehend angezeigt wird. Daher wird für die Darstellung der Projektionen vereinfacht angenommen, dass die Bildebene vor dem Projektionszentrum liegt und so das Bild nicht spiegelverkehrt projiziert wird. [2]

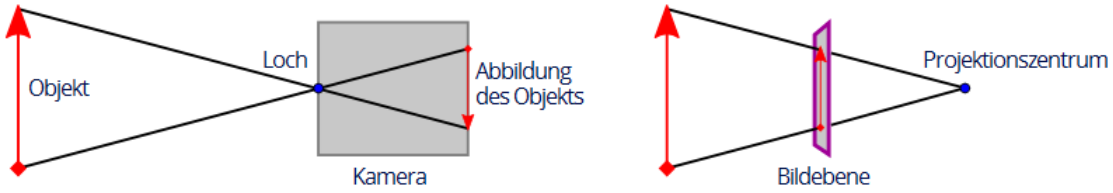


Abbildung 2.6.: Skizze einer Lochkamera und eine vereinfachte Darstellung einer Projektion [18, S.10]

Die **Projektion** der Bilder kann dabei sehr unterschiedlich ausfallen. Sie beschreibt wie Objekte aus einem höher dimensionierten Raum auf einen niedriger dimensionierten Raum dargestellt werden sollen. Hierfür gibt es je nach Art der Darstellung verschiedene Projektionsvorschriften. Im Allgemeinen wird von **planaren Projektionen** gesprochen, da alle Projektionen durch Projektionslinien auf eine Fläche (engl. plane) abgebildet werden. In Abbildung 2.7 ist eine Übersicht möglicher Darstellungstypen zu finden. Anschließend wird kurz auf die Darstellungsarten eingegangen, um einen Überblick über die Darstellung von Bildern zu verschaffen. Lediglich die perspektivische Projektion soll genauer dargestellt werden, da diese in Fernsehaufnahmen üblich und damit für diese Masterarbeit relevant ist.

Die **Parallelprojektion** zeichnet sich durch einen parallelen Verlauf der Projektionsstrahlen aus. Folglich schneiden sie sich nicht, sodass das Projektionszentrum im Unendlichen liegt. Daher wird in diesem Fall auch nicht von Projektionszentrum gesprochen, sondern von Projektionsrichtung (engl. direction of projection DOP). Sie zeichnen sich unter anderem dadurch aus, dass parallel verlaufende Linien eines Bildes auch nach der Projektion noch parallel dargestellt werden. Parallelprojektionen lassen sich weiterhin in orthographisch, axonometrisch und oblique unterteilen. [2]

In Abbildung 2.8(a) ist eine **orthographische Projektion** zu sehen, bei der die Projektionsstrahlen senkrecht zur Bildebene, sowie parallel zueinander abgebildet sind. In Abbildung 2.8(b) sind verschiedene orthographische Ansichten gezeigt.

Die **axonometrische Projektion** zeichnet sich durch senkrecht zur Bildebene verlaufende Projektionsstrahlen aus, wobei die Bildebene gegenüber dem Objekt jede Orientierung besitzen

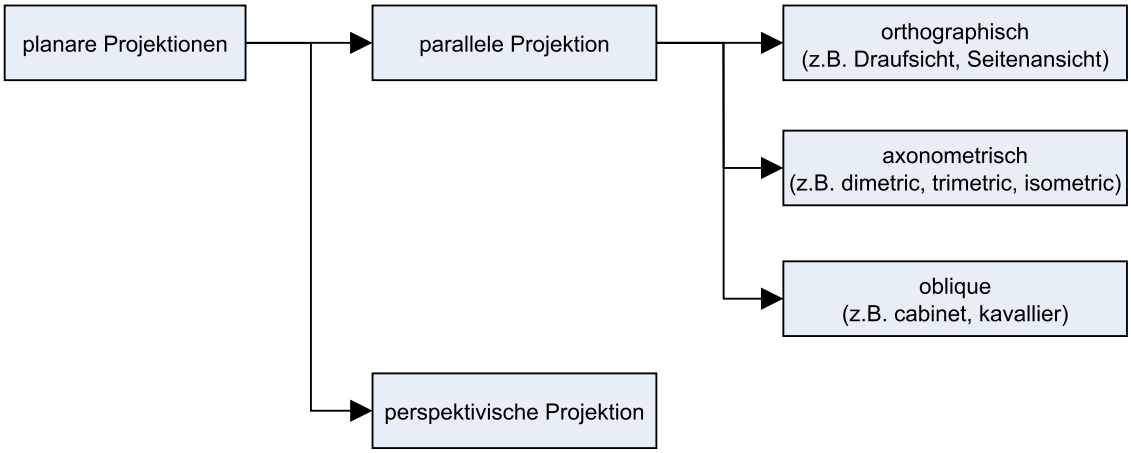


Abbildung 2.7.: Übersicht der Darstellungstypen von planaren Projektionen, nach [21]

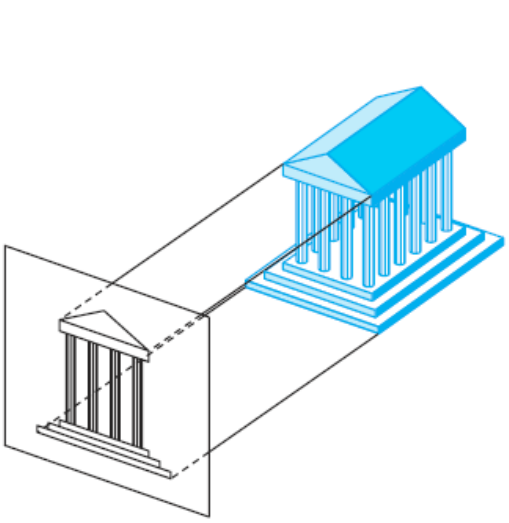
kann. In Abbildung 2.8(c) ist zu sehen, dass die trimetrische[4] Ansicht (linke Grafik) parallel verlaufende Projektionsstrahlen besitzt. Diese Art der Projektion ist in den beiden Ansichten daneben zum einen von oben (mittlere Grafik) und zum anderen von der Seite (rechte Grafik) dargestellt. Darin ist zu erkennen, dass die Projektionsstrahlen weiterhin senkrecht zur Bildebene stehen, das Objekt aber in Bezug auf die Bildebene eine abweichende Orientierung besitzt.

In Abbildung 2.8(d) ist eine **oblique Projektion** zu sehen. Es zeigt sich, dass die Projektionsstrahlen nicht im rechten Winkel auf der Bildebene stehen, was die oblique Projektion auszeichnet. Sie ist damit die allgemeinste Form der parallelen Projektionen, da neben den parallel verlaufenden Projektionsstrahlen keine weiteren Einschränkungen getroffen werden. [2]

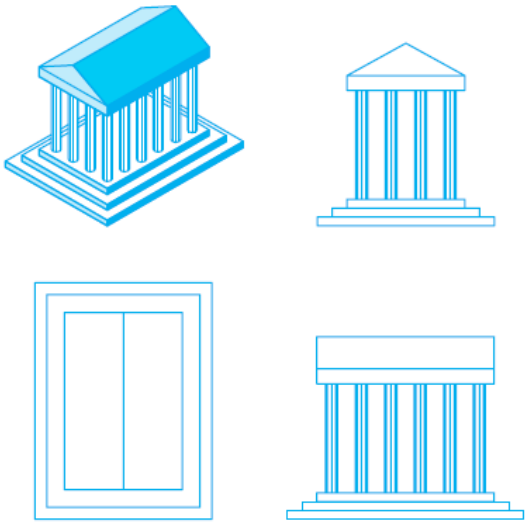
Die **perspektivische Projektion** oder auch Zentralprojektion genannt, stellt 3D Objekte realistisch dar. Objekte erhalten ihre natürliche Erscheinung wie Menschen sie wahrnehmen, da das menschliche Auge Objekte in der Umgebung ebenfalls durch eine perspektivische Projektion wahrnimmt. Im Gegensatz zur Parallelprojektion sind die Projektionsstrahlen dabei nicht parallel zueinander, sondern laufen im Projektionszentrum zusammen. Im Bild parallel verlaufende Linien, die nicht parallel zur Projektionsebene sind, konvergieren in einem Punkt. Diese Punkte werden **Fluchtpunkte** genannt. Je nach Lage des Objektes können bei einer perspektivischen Projektion ein bis drei Fluchtpunkte (ein Fluchtpunkt pro Achse des Koordinatensystems) existieren. Aufgrund dieser Darstellung bleiben die Längen und Winkel von Objekten bei dieser Projektionsart nicht erhalten. Die Größe von Objekten verändert sich in Bezug auf die Distanz zur Kameraposition. Je weiter das Objekt von der Kamera entfernt ist, desto kleiner wird dieses dargestellt. In Abbildung 2.9 ist eine perspektivische Projektion wiedergegeben. [2]

In den nächsten Abschnitten soll aus mathematischer Sicht betrachtet werden, wie Punkte durch eine perspektivische Projektion auf eine Bildebene abgebildet werden können. Dabei wird wie

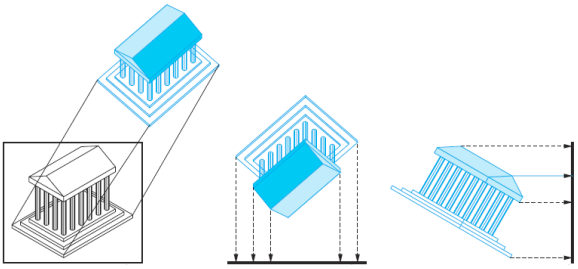
[4] trimetrisch: Winkel zwischen allen drei Koordinatenachsen ist unterschiedlich



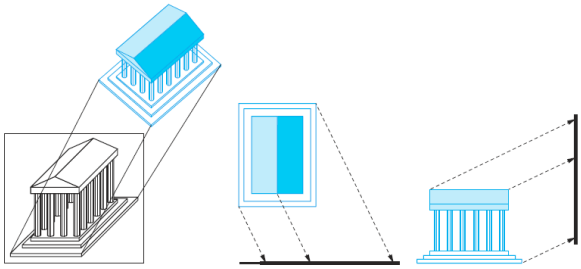
(a) orthographische Projektion [2, S.198]



(b) verschiedene orthographische Ansichten [2, S.199]



(c) axonometrische Projektion [2, S.199]



(d) oblique Projektion [2, S.200]

Abbildung 2.8.: Darstellung verschiedener Projektionen

in Abbildung 2.10 davon ausgegangen, dass sich der abzubildende Punkt $P(x, y, z)$ in einem dreidimensionalen Raum befindet und auf einen Punkt $\tilde{P}(\tilde{x}, \tilde{y}, \tilde{z})$ in die Bildebene projiziert wird. Die Kamera stellt das Projektionszentrum dar und zeigt in Richtung der z-Achse. Die Bildebene steht dabei senkrecht auf der z-Achse und liegt auf der Höhe $z = f$. f ist dabei die Brennweite der Kamera, die angibt, wie weit die Kamera ($z = 0$) von der Bildebene entfernt ist. In der rechten Grafik der Abbildung 2.10 ist gut zu erkennen, dass mit Hilfe des Strahlensatzes die Koordinate $\tilde{x}$ zu ermitteln ist. Nach Anwendung des Strahlensatzes

$$\frac{\tilde{x}}{f} = \frac{x}{z} \quad und \quad \frac{\tilde{y}}{f} = \frac{y}{z}$$

wird für $\tilde{x}$ und $\tilde{y}$ folgendes Ergebnis erhalten:

$$\tilde{x} = \frac{x \cdot f}{z} \quad und \quad \tilde{y} = \frac{y \cdot f}{z}$$

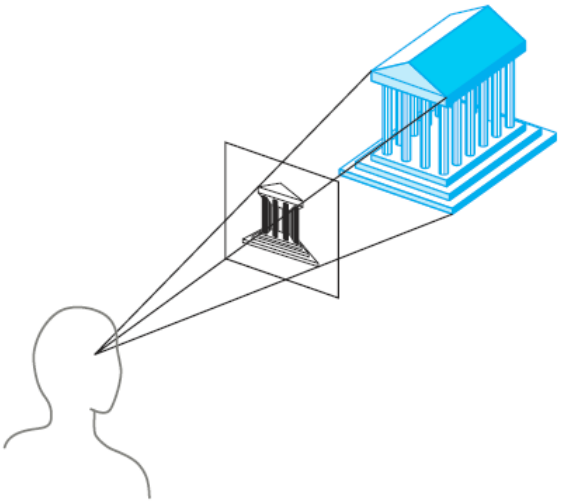


Abbildung 2.9.: Darstellung einer perspektivischen Projektion [2, S.201]

Dadurch ergeben sich für Punkt $\tilde{P}$ folgende Koordinaten:

$$\tilde{P} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ f \end{bmatrix} = \begin{bmatrix} \frac{x \cdot f}{z} \\ \frac{y \cdot f}{z} \\ f \end{bmatrix} \tag{2.10}$$

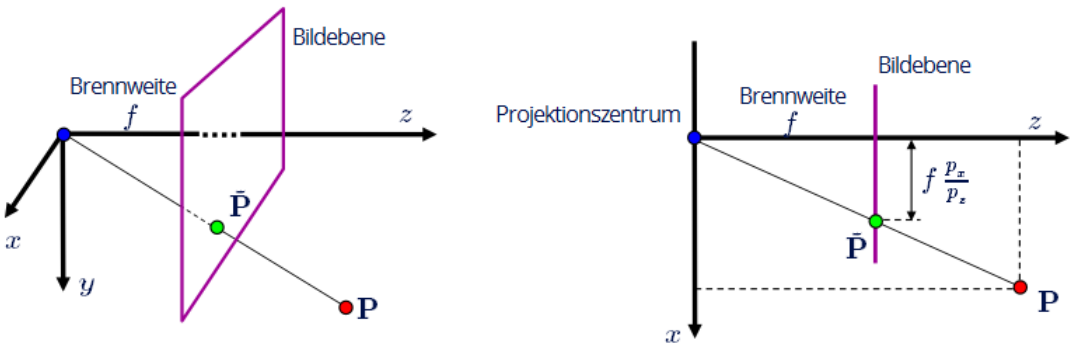


Abbildung 2.10.: Skizze einer perspektivischen Projektion in seitlicher Ansicht [18, S.11]

Durch eine Division durch z ergibt sich die Verkleinerung der Objekte, je weiter die abzubildenden Bildpunkte vom Projektionszentrum (Kamera) entfernt sind. Allerdings ist dieser Transformationsprozess nicht umkehrbar. Der Bildpunkt P kann zwar auf $\tilde{P}$, aus $\tilde{P}$ allerdings nicht eindeutig auf P abgebildet werden, da alle Punkte, die auf demselben Projektionsstrahl liegen, in den gleichen Punkt $\tilde{P}$ der Bildebene projiziert werden. Dadurch fehlt die Distanz zur Kamera, die es ermöglicht entscheiden zu können welcher Punkt der Kamera am nächsten liegt.

Diese Projektion kann ebenfalls mit homogenen Koordinaten dargestellt werden. Homogene

Koordinaten stellen dreidimensionale Punkte (x, y, z) als vierdimensionale Punkte $(x, y, z, 1)$ dar. Aus der Umformung der Gleichung 2.10 in homogene Koordinaten resultiert folgende Gleichung:

$$\tilde{P} = \begin{bmatrix} \frac{x \cdot f}{z} \\ \frac{y \cdot f}{z} \\ f \\ 1 \end{bmatrix} \tag{2.11}$$

Diese Transformation lässt sich ebenfalls als Matrix M darstellen, die den Punkt P auf den Punkt $\tilde{P}$ in die Bildebene projiziert.

$$\tilde{P} = M \cdot P = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix}}_{M} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{f} \end{bmatrix} = \begin{bmatrix} \frac{x \cdot f}{z} \\ \frac{y \cdot f}{z} \\ f \\ 1 \end{bmatrix} \tag{2.12}$$

Gleichung 2.12 zeigt die Abbildungsvorschrift der perspektivischen Projektion durch die ein Punkt P auf $\tilde{P}$ in die Bildebene abgebildet wird. Wie zuvor beschrieben, ist es allerdings nicht möglich aus Punkt $\tilde{P}$ und Matrix M auf Punkt P zu schließen, da hierbei die Informationen der Tiefe (z Koordinate) fehlen. Dies zeigt sich auch darin, dass die Matrix M nicht invertiert werden kann, da die Determinante von M null ergibt. Daher soll im Anschluss eine weitere Möglichkeit vorgestellt werden, wie vom projizierten Punkt $\tilde{P}$ der Ursprungspunkt P hergeleitet werden kann. [18] [2]

Dies ist durch Bildung einer **Homographie** Matrix möglich. Die Homographie Matrix H beschreibt die Zuordnung von Punkten zwischen zwei Projektionsebenen, die das gleiche Projektionszentrum besitzen. Dadurch ist eine Zuordnung $P' = H \cdot P$, durch Matrixmultiplikation von der Homographie Matrix H mit dem Punkt P zu P' möglich, der sich auf einer anderen Projektionsebene befindet, aber auf dem gleichen Projektionsstrahl wie Punkt P liegt. Dies ist in Abbildung 2.11 zu sehen, in der die Projektionsstrahlen durch zwei Projektionsebenen hindurchgehen und somit das gleiche Projektionszentrum besitzen.

Die Homographie Matrix H ist eine $3x3$ Matrix, die im zweidimensionalen Raum die Zuordnung von Bildpunkten verschiedener Bilder zulässt. Dies wird durch folgende Gleichung beschrieben:

$$P' = H \cdot P = \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.13}$$

In Abhängigkeit der Beziehungen der beiden Projektionsebenen zueinander ist die Homographie Matrix H unterschiedlich aufgebaut. Sie kann Objekte durch Rotation, Translation und Skalierung verändern. Bei perspektivischen Projektionen kommt zudem eine Verzerrung hinzu, sodass die Matrix mit 9 Elementen insgesamt 8 Freiheitsgrade besitzt.

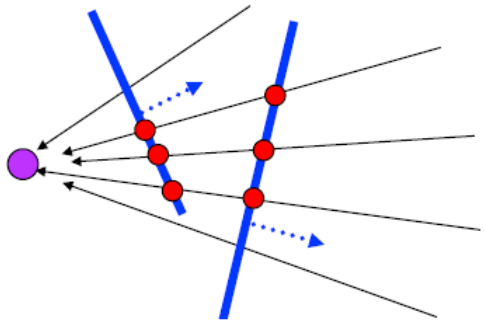


Abbildung 2.11.: Darstellung von zwei Projektionsebenen mit gleichem Projektionszentrum, nach [22]

Um die Homographie Matrix H zu erhalten, muss die Formel $P' = H \cdot P$ nach H aufgelöst werden.

$$\underbrace{\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{h} = \underbrace{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}_{Nullvektor} \tag{2.14}$$

Bei Verwendung von vier Zuordnungen von Punkten zwischen den Projektionsebenen ergibt sich für Matrix A eine $8x : 9$ Matrix. Die Gleichung 2.15 stellt eine direkt lineare Transformationsgleichung dar, die durch diese vier Zuordnungen durch den **Least Square Fit** Algorithmus gelöst werden kann. Durch Einsetzen der vier Punktzuordnungen von P und P' wird nachfolgende

Gleichung erhalten:

$$A \cdot h = 0 \Leftrightarrow \underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 & -y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 & -x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 & -y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 & -y'_4 \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{h} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{Nullvektor} \tag{2.15}$$

Der Least Square Fit Algorithmus ist ein Verfahren das Gleichungssysteme, mit mehr Gleichungen als Unbekannten, näherungsweise bestimmen kann. Dazu werden als Lösung die Werte angenommen, deren Summe aus den einzelnen quadrierten Fehlern jeder einzelnen Gleichung das Minimum ergibt. Er minimiert folglich die oben genannte Gleichung und definiert daraus ein Least Squares Problem: $\left\|Ah - 0\right\|^2$. [22, S.19]

In Fernsehaufnahmen wird ein perspektivisch verzerrtes Tennisfeld dargestellt. Durch Zuordnung der offiziellen Spielfeldmaße zum perspektivischen Spielfeld kann eine Homographie Matrix berechnet werden, mit der das perspektivische Bild in ein Bild in Draufsicht[5] umgewandelt werden kann.

Die Anwendung des Least Square Fit Algorithmus resultiert in einer Homographie Matrix, mit der es möglich ist die Punkte aus dem Ausgangsbild P in das Zielbild P' anhand der Formel $P' = H \cdot P$ aus Gleichung 2.13 zu transformieren. Dieser Vorgang wird **Image Rectification** genannt. [24] [22]

2.3. Kalman Filter

Gerade in der Bildverarbeitung geht es häufig darum in Echtzeit Objekte nachzuverfolgen. Der Aufwand das gesamte Bild zu analysieren ist jedoch sehr hoch. Durch eine Methode, die es erlaubt die zukünftige Position des verfolgten Objektes vorauszuahnen, kann die nächste Objekterkennung in einem kleineren Suchbereich stattfinden. Der diskrete Kalman Filter ist ein rekursiver Algorithmus, der gut für das Vorhersagen von Bewegungen geeignet ist. Die Vorhersage des Kalman Filters ist bei hoher Framerate und bei Vermeidung von plötzlichen Richtungswechseln sehr gut. Er bezieht alle bekannten Teile des Systems mit ein und versucht daraus eine Prognose für den Bewegungsablauf zu treffen. Er gilt dabei als optimaler Schätzer, der aus ungenauen

[5] Draufsicht: ist eine orthographische Ansicht, bei der aus einer Vogelperspektive senkrecht auf das betrachtende Objekt (Bildebene) geblickt wird.

oder fehlerbehafteten Daten die Vorhersage trifft. Dafür minimiert er den durchschnittlichen quadratischen Fehler der geschätzten und fehlerbehafteten Parameter. [5] [9]

Allgemein lässt sich der Kalman Filter wie folgt beschreiben: Es wird ein Wert gemessen und eine Schätzung aufgrund von vergangenen Messwerten angestellt. Je nach Güte der Schätzung wird dieser ein "Gain", also ein Verstärkungsfaktor, beigemessen. Ist die Schätzung so genau, dass ein hoher "Kalman-Gain" verwendet wird, fällt die Schätzung somit stärker ins Gewicht. Durch Mittelung des Messwerts und der Schätzung (multipliziert mit dem Gain-Wert) wird ein gefiltertes Signal berechnet, wodurch Störeinflüsse minimiert wurden.

Im Folgenden soll der diskrete Kalman Filter genauer vorgestellt werden. Dieser hat grundsätzlich als Ziel den Systemzustand $x_k \in \mathbb{R}^n$ zu schätzen. x_k wird auch als Zustandsvektor bezeichnet. Dafür wird die Gleichung

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.16}$$

mit der Messung $z \in \mathbb{R}^m$ verwendet.

$$z_k = Hx_k + v_k \tag{2.17}$$

Die $n\,x\,n$ Matrix A aus Gleichung 2.16 beschreibt die Bewegung des Objektes und wird Transitionmatrix genannt. Die $n\,x\,1$ Matrix B stellt die Verbindung von Zustand x zu den Kontrolleingaben u her. Diese ist allerdings häufig optional und entfällt daher oftmals. Die $m\,x\,n$ Matrix H verbindet die Messung mit dem Zustandsvektor. Die beiden Zufallsvariablen w_{k-1} und v_k stellen die Messstörungen (v_k) und das Prozessrauschen (w_{k-1}) dar. Sie sind unabhängig voneinander und werden als normale Wahrscheinlichkeitsverteilung beschrieben.

$$p(w) \sim N(0, Q) \tag{2.18}$$

$$p(v) \sim N(0, R) \tag{2.19}$$

Die Kovarianzmatrix Q beschreibt die Prozessstörungen, die Kovarianzmatrix R die Messstörungen. Sie können sich normalerweise in jedem Zeitschritt ändern, werden aber häufig zur Vereinfachung als konstant angenommen. Weiterhin verwendet der Kalman Filter eine Kovarianzmatrix P, die den Fehler zwischen dem aktuellen Zustand x_k und der Schätzung des Zustandes $\hat{x}_k$ des Kalman Filters selbst beschreibt.

$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \tag{2.20}$$

Mit Hilfe dieser Gleichungen kann der Algorithmus des Kalman Filters nun genauer beschrieben werden. Wie bereits erwähnt, ist es das Ziel den Zustandsvektor x_k abzuschätzen. Der geschätzte Zustandsvektor wird im Weiteren durch folgende Gleichung dargestellt:

$$\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - H\hat{x}_{k-1}) \tag{2.21}$$

Die Indizes k stellen hierbei die Zustände dar. Der Zustand k repräsentiert den aktuellen Messwert und $k-1$ den Vergangenheitswert. Dabei setzt sich die Schätzung aus einem gewichteten Messwert und einer gewichteten Schätzung des vorherigen Zustands zusammen. Der gemessene Wert z_k wird mit einem "Kalman-Gain" K_k gewichtet. Dieser kann zu Beginn festgelegt werden, zweckdienlicher ist aber die kontinuierliche Neuberechnung in jedem Messschritt, da sich die Schätzungstreue über die Messung hinweg verändern kann. Der Term $K_k * z_k$ stellt dabei die Gewichtung multipliziert mit dem gemessenen Wert des aktuellen Zustandes dar, der Term $(1-K_k) * \hat{x}_{k-1}$ enthält den gewichteten Vergangenheitswert der Schätzung. Der Kalman Gain beschreibt mit seiner Gleichung

$$K_k = P_k H^T (H P_k H^T + R)^{-1} = \frac{P_k H^T}{H P_k H^T + R} \tag{2.22}$$

wie sicher der aktuellen Messung beziehungsweise der aktuellen Schätzung vertraut werden kann. Wenn die Kovarianzmatrix R des Messfehlers sehr klein wird, steigt das Vertrauen in die aktuelle Messung z_k. Für den Fall, dass die Kovarianzmatrix P des Schätzungsfehlers kleiner wird, wird der Schätzung stärker vertraut.

Der Kalman Filter bezieht durch seine rekursive Eigenschaft die vergangenen Mess- und Schätzwerte mit ein. Dafür schätzt er den Systemzustand ab und benutzt die Messung als Feedback. Der Kalman Filter kann daher in zwei Schritte eingeteilt werden. a) die **Vorhersage** und b) die **Korrektur**. Zuerst wird ein Systemzustand vorhergesehen. Die dafür verwendeten Gleichungen werden auch als "time update" Gleichungen bezeichnet. Anhand der Messungen wird die Schätzung korrigiert, was durch die "measurement update" Gleichungen beschrieben wird. Anschließend kann der nächste Zustand mit den korrigierten Messwerten ermittelt werden.

Dieser Kreislauf wird in Tabelle 2.1 allgemein beschrieben:

Bei der Vorhersage wird die aktuelle Schätzung x_k anhand der vorherigen Schätzung x_{k-1} ermittelt. Weiterhin wird die Fehlerkovarianz P_k bestimmt. Im Korrekturschritt wird der Kalman Gain K_k anhand der Fehlerkovarianz P_k, der Matrix H und der Kovarianzmatrix R für Messungenauigkeiten berechnet. Mit Hilfe des Kalman Gains kann die Schätzung x_k korrigiert werden, genauso wie die Fehlerkovarianz P_k. Diese Schritte werden rekursiv ausgeführt.

Im Falle der Vorhersage von Objekten mit Hilfe von Bilderkennung können einige Matrizen als konstant angesehen werden und vereinfachen dadurch die Gleichungen. Die Matrix A beschreibt die Bewegung des Objektes in Form von Position- und Geschwindigkeitsangaben. In der Bildverarbeitung handelt es sich um 2D Koordinaten x und y, sowie deren Geschwindigkeit v_x und v_y, die als zeitliche Änderung des zurückgelegten Weges $v_x = \frac{x}{\Delta t}$ dargestellt werden kann. Daraus ergibt sich für die Matrix A folgende Darstellung:

$$\hat{x} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \tag{2.23}$$

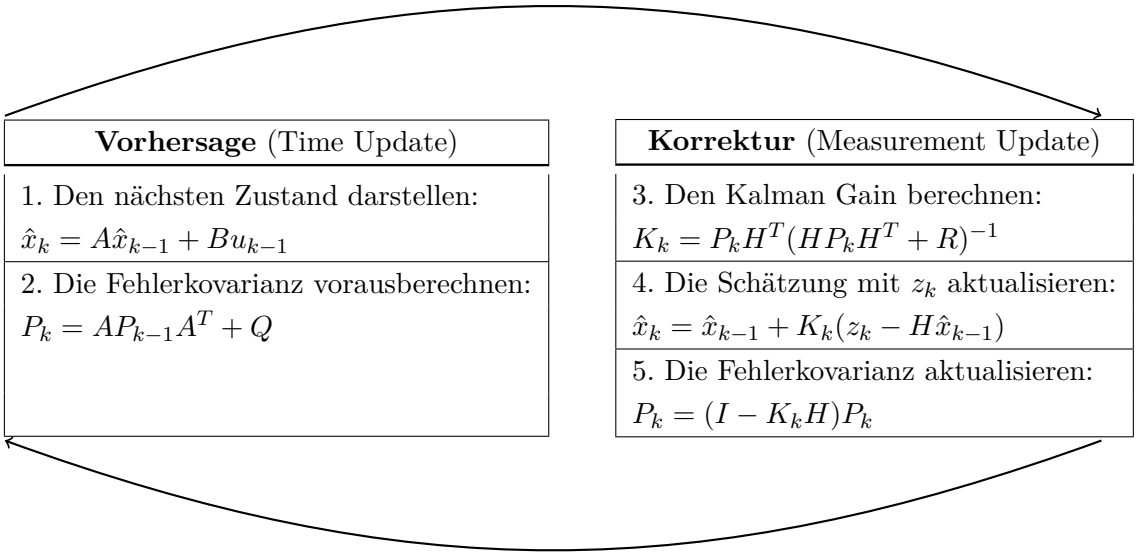


Tabelle 2.1.: Ablauf des Kalman Filtervorgangs, nach [5]

Die Matrix B, sowie der Vektor u entfällt, da von außen nicht in den Prozess der Bilderkennung eingegriffen wird. Die Kovarianzmatrizen Q und R können als konstant angesehen werden und müssen daher nur initial bestimmt werden. Die Fehlerkovarianz der Messung R spiegelt bei der Bilderkennung die quadratische Standardabweichung des Messfehlers wieder. Dieser Fehler wird durch die Bildqualität und die Verhältnisse auf dem Bild beeinflusst. Das gleiche Objekt kann daher, je nach Bildqualität oder beispielsweise bei unterschiedlichen Lichtverhältnissen, verschieden groß dargestellt werden. Die Fehlerkovarianz des Prozesses Q ist dagegen schwerer zu bestimmen. Hierbei handelt es sich um die Abweichung in deren Rahmen das Objekt seine Richtung ändern kann. Diese Richtungswechsel sind im Tennis sehr plötzlich, da der Ball beim Schlagtreffpunkt ohne Vorwarnung die Richtung wechselt und auch über $200km/h$ fliegen kann. Kohler beschreibt in seinem Paper [9], dass menschliche Bewegungen mit einer Beschleunigung von $a < 11\frac{m}{s^2}$ stattfinden. Die Kovarianzmatrizen Q und R können laut [9] somit folgendermaßen dargestellt werden:

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{2.24}$$

$$Q = \frac{a^2\Delta t}{6} \begin{bmatrix} 2I(\Delta t)^2 & 3I\Delta t \\ 3I\Delta t & 3I \end{bmatrix} \tag{2.25}$$

Die Matrix H stellt die Beziehung zwischen dem Zustandsvektor und der Messung dar. Gemessen wird lediglich die Position. Die Geschwindigkeit wird vom Kalman Filter selbst berechnet. Daher

ergibt sich für die Matrix H folgende Gleichung:

$$z = \begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{H} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \tag{2.26}$$

Die Fehlerkovarianzmatrix der Schätzung P muss nicht zwingend initialisiert werden, da sie vom Kalman Filter aktualisiert wird. Alles in allem ist die Festlegung der Werte für die Kovarianzen nicht immer einfach, daher ist es oftmals nötig, den Kalman Filter durch Testen der entsprechenden Parameter zu optimieren. [5] [9]

2.4. Clusterverfahren

Das Ziel von Clusterverfahren ist es Objekte sinnvoll zu gruppieren, sodass Objekte einer Gruppe möglichst ähnliche Eigenschaften besitzen. Weiterhin sollen sich die daraus gebildeten Cluster möglichst stark unterscheiden. Die Ähnlichkeit der Objekte wird dabei häufig anhand Distanzfunktionen festgemacht. Objekte mit einer kleineren Distanz zueinander sind sich ähnlicher, als Objekte die eine große Distanz zueinander aufweisen. In Abbildung 2.12 sind Cluster dargestellt die alle unterschiedliche Formen aufweisen. Ziel der Clusterverfahren ist es diese Cluster und auch dessen Formen zu erkennen. Dafür gibt es verschiedene Gruppen von Clusterverfahren von denen die Bekanntesten in den nächsten Abschnitten kurz vorgestellt werden. [6, S.384ff]

Abbildung 2.12.: Darstellung verschiedener Cluster mit unterschiedlichen Formen, Größen und Dichten [11, S.3]

2.4.1. Partitionierende Clusterverfahren

Partitionierende Clusterverfahren versuchen Objekte in k Cluster einzuteilen, sodass die Kosten des Clusters minimiert werden. Ein Maß für die Kosten ist beispielsweise die Kompaktheit E eines Clusters C, also der quadrierte Abstand zwischen jedem einzelnem Objekt p des Clusters zum Centroid[6] m_i des Clusters i. Die Kosten eines kompletten Clustering, also von k Clustern,

[6] Centroid: Mittelwert aller Punkte im Cluster

ist die Summe der Kosten der k einzelnen Clustern. Insgesamt ergibt sich daher folgende Formel:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2 \tag{2.27}$$

Der **k-means Algorithmus** ist dabei einer der bekanntesten Ansätze der partitionierenden Clusterverfahren. Dieser teilt den zuvor initial festgelegten Centroiden Objekte zu, sodass alle Objekte den nächstliegende Centroid zugeordnet werden. Anschließend werden die Centroide von den neu entstandenen Clustern bestimmt. Diese beiden Schritte werden solange wiederholt, bis sich die Centroide und deren Cluster nicht mehr verändern. Vorteil dieses Algorithmus ist, dass es sich hierbei um einfache Implementierung handelt und lediglich einen Aufwand von $O(n)$ für eine Iteration besitzt. Weiterhin wird bereits nach $5 - 10$ Iterationen ein Clusterergebnis gefunden. Allerdings muss beim k-means Algorithmus die Anzahl der Cluster k zuvor angegeben werden, was oft schwer zu entscheiden ist. Ebenfalls werden alle Punkte in ein Cluster mit einbezogen, sodass Ausreißer das Clustering stark beeinflussen. Weiterhin müssen die Cluster für diesen Algorithmus eine konvexe Form haben, da nur der Abstand zum Centorid betrachtet wird. Auch spielt die initiale Festlegung der Centroide eine starke Rolle, da das Clustering dadurch ein anderes Ergebnis und auch schneller oder langsamer das endgültige Clusterergebnis erhalten kann. Deswegen gibt es neben dem k-means vielen weiteren partitionierenden Clusterverfahren, wie k-medoid, aber auch andere Arten von Clusterverfahren. [6, S.401ff]

2.4.2. Hierarchische Clusterverfahren

Bei dem hierarchischen Clusterverfahren werden Objekte zu einer Baumstruktur gruppiert. Dies wird auch Dendrogramm genannt. Bei einem Dendrogramm spiegelt jeder Knoten einen Cluster wieder. Die Wurzel des Baumes umfasst die gesamte Datenbasis. Die Blätter repräsentieren Objekte. Dadurch entsteht eine Hierarchie der Cluster. Es gibt zwei Arten dieses Clustering zu bilden. Zum einen den **agglomerativen Ansatz**, der auch Bottom-Up Ansatz genannt wird. Dieser teilt zu Beginn jedes Objekt einem Cluster zu. Die Objekte die nahe beieinanderliegen werden Schritt für Schritt zusammengefasst. Zum anderen der **divisive Ansatz**, der auch Top-Down Ansatz genannt wird. Er teilt anfänglich alle Objekte genau einem Cluster zu. Diese Cluster werden nun schrittweise anhand von Kriterien aufgeteilt, bis jedes Objekt genau einem Cluster entspricht oder ein Endekriterium erfüllt.

Der **Algorithmus des agglomerativen Clusterverfahrens** soll hier kurz dargestellt werden:

1. Bilde aus jeweils einem Objekt ein Cluster und bestimme die Distanzen zwischen allen Paaren der Cluster
2. Bilde ein neues Cluster aus den Clustern mit der geringsten Distanz (anhand Distanzfunktion) zueinander
3. Bestimme die Distanz zwischen dem neuen Cluster und allen anderen Clustern
4. Wiederhole Schritt 2 und 3 solange, bis sich alle Objekte in einem einzigen Cluster befinden

Für die Distanzfunktionen zwischen Clustern gibt es drei typische Ansätze. Der **Single Link** Ansatz nimmt als Clusterdistanz die kleinste vorhandene Distanz zwischen den jeweiligen Objekten der verschiedenen Cluster an. Beim **Complete Link** wird im Gegensatz dazu die größtmögliche Distanz als Clusterdistanz gewählt. Die **Avarage Link** Funktion mittelt dagegen die Distanzen zwischen den einzelnen Objekten und gibt dadurch die durchschnittliche Distanz an. Nachfolgend werden die Formeln dazu wiedergegeben:

- Single Link: $d_{min}(C_x, C_y) = min_{x \in C_x, y \in C_y} |x - y|$
- Complete Link:
$$d_{max}(C_x, C_y) = max_{x \in C_x, y \in C_y} |x - y|$$
- Average Link: $d_{avg}(C_x, C_y) = \frac{1}{|C_x| \cdot |C_y|} \cdot \sum_{x \in C_x} \sum_{y \in C_y} |x - y|$

Die Vorteile von hierarchischen Clusterverfahren sind, dass nicht nur ein Clusterergebnis erhalten wird, sondern ganze Clusterhierarchien. Es können folglich auch Cluster in Clustern festgestellt werden. Weiterhin kann die Anzahl der Cluster automatisch ermittelt werden, ohne diese zuvor angeben zu müssen. Allerdings werden auch beim hierarchischen Clusterverfahren Ausreißer miteinbezogen. Dieses Verfahren besitzt eine Laufzeit von $O(n^2)$ für n Objekte. [6, S.409ff]

2.4.3. Dichtebasierte Clusterverfahren

Bei dichtebasierten Clusterverfahren werden einem Cluster Objekte zugeordnet, die in einer vorgegebenen Suchdichte liegen. Die einzelnen Cluster werden durch weniger dichte Gebiete voneinander getrennt. Ein dichtebasiertes Clusterverfahren ist der **DBSCAN Algorithmus** was für *Density-Based Spatial Clustering of Applications with Noise* steht. Bevor dieser Algorithmus genauer erläutert wird, sollen zuvor grundlegende Definitionen von dichtebasierten Clusterverfahren zusammengefasst werden.

O ist eine endliche Menge von Objekten mit $|O| = n$. Weiterhin sind eine Distanzfunktion d sowie Parameter für den DBSCAN Algorithmus $minPts \in \mathbb{N}_{\geq 0}$ und $\epsilon \in \mathbb{R}_{\geq 0}$ gegeben.

Definition 1. *Ein Objekt $o \in O$ heißt Kernobjekt, wenn gilt: $|N_\epsilon(o)| \geq minPts$, mit $N_\epsilon(o) = \{o' \in O | d(o, o') \leq \epsilon\}$*

Kernobjekte sind alle Objekte bei denen in der ϵ-Umgebung[7] mindestens $minPts$ der Objekte liegen. Alle anderen Objekte, werden Nicht-Kernobjekte genannt.

Definition 2. *Ein Objekt $o \in O$ heißt direkt dichte-erreichbar von $p \in O$, wenn p ein Kernobjekt und $o \in N_\epsilon(p)$ ist.*

[7] ϵ-Umgebung: Umgebung um das Objekt mit der Distanz ϵ

Nach Definition 2 sind Objekte die in der ϵ-Umgebung eines Kernobjektes liegen **direkt dichte-erreichbar**.

Definition 3. *Ein Objekt $o \in O$ heißt dichte-erreichbar von $p \in O$, wenn es eine Folge von Objekten $o_1, \ldots, o_m$ gibt, sodass $o_1 = p, o_m = o$ und o_{i+1} direkt dichte-erreichbar von o_i ist, für $1 \leq i < m$.*

Objekte o sind folglich dann **dichte-erreichbar** von p, wenn es eine Kette von direkt dichte-erreichbaren Objekten zwischen p und o gibt. Daraus lässt sich folgende Definition für einen Cluster C bilden.

Definition 4. *Die Menge $C = \{o \in O | o$ ist dichte-erreichbar von $p\}$ ist ein Cluster von O, wenn p ein Kernobjekt in O ist.*

Daraus folgt, dass ausgehend von einem beliebigen Kernobjekt $o \in C$, alle dichte-erreichbaren Objekte o zu einem **Cluster** C gehören. Alle Objekte die keinem Cluster zugeordnet werden, werden als Ausreißer angesehen.

In Listing 2.1 und 2.2 ist der DBSCAN-Algorithmus dargestellt.

Listing 2.1: DBSCAN Algorithmus 1: dbscan(O, ϵ, minPts)

```
1  Setze clusterID auf 1
2  für alle o ∈ O
3      wenn o noch in keinem Cluster
4          wenn erweitereCluster(O, o, clusterID, ϵ, minPts)
5              inkrementiere clusterID
```

Listing 2.2: DBSCAN Algorithmus 2: erweitereCluster(O, o, clusterID, ϵ, minPts)

```
1  wenn |N_ϵ(o)| < minPts
2      kennzeichne Cluster als Ausreißer
3      return false
4
5  addCluster(N_ϵ(o), clusterID)
6  erweitereKandidaten(N_ϵ(o), o, clusterID, kandidaten)
7  solange es weitere Kandidaten gibt
8      wähle ein Objekt o' aus den Kandidaten aus
9      entferne o' aus Kandidaten
10     wenn |N_ϵ(o')| ≥ minPts
11         addCluster(N_ϵ(o'), clusterID)
12         erweitereKandidaten(N_ϵ(o'), o, clusterID, kandidaten)
13 return true
```

In Listing 2.1 wird der Algorithmus gestartet, indem zu Beginn die *clusterID* auf eins gesetzt wird. Nun werden alle Objekte o durchlaufen, die noch keinem Cluster zugeordnet sind. In Zeile 4 wird durch Methode *erweitereCluster* untersucht, ob dem Objekt o ein Cluster zugeteilt werden

kann. Wenn das Objekt o einem Cluster angehört und es sich nicht um einen Ausreißer handelt, wird die *clusterID* erhöht.

In Listing 2.2 wird überprüft, ob das Objekt o zu einem Cluster gehört. Dies wird in Zeile 1 durchgeführt, indem die Anzahl der Objekte, die in der ϵ-Umgebung liegen, mit dem Parameter *minPts* verglichen wird. Wenn o weniger als *minPts* Objekte hat, handelt es sich um einen Ausreißer. In diesem Fall wird die Methode erweitereCluster verlassen und mit dem nächsten Objekt fortgefahren. Für den Fall das o ein Kernobjekt ist, werden alle direkt dichte-erreichbare Objekte dem Cluster *clusterID* hinzugefügt. Weiterhin wird in Zeile 6 die Kandidatenliste erweitert, sodass für alle Objekte, die direkt dichte-erreichbar von o sind, zudem überprüft wird, ob es sich bei ihnen ebenfalls um ein Kernobjekt handelt. Dies wird von Zeile 8 bis 12 iterativ durchgeführt, bis alle Kandidaten untersucht wurden und somit das Cluster, ausgehend von Objekt o vollständig erkannt wurde.

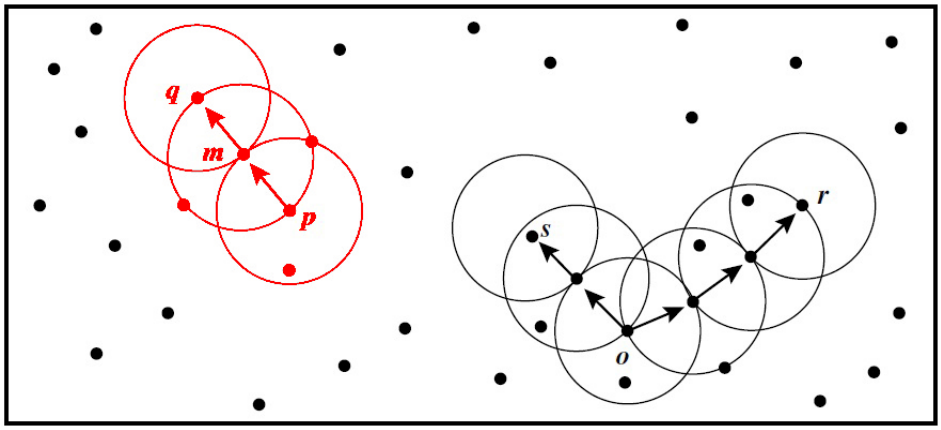


Abbildung 2.13.: Dichte-Erreichbarkeit bei dichtebasiertem Clustering, nach [6, S.419]

Anhand von Abbildung 2.13 wird der DBSCAN Algorithmus beispielhaft erläutert und die Dichte-Erreichbarkeit von Objekten dargestellt. Es wird davon ausgegangen, dass der DBSCAN Algorithmus mit $minPts = 4$ bei Objekt p startet. Bei Objekt p handelt es sich um ein Kernobjekt, da sich in dessen Radius ϵ vier weitere Objekte befinden. Alle dichte-erreichbaren Objekte gehören damit zu diesem Cluster. Daraufhin werden alle Objekte die von p direkt dichte-erreichbar sind überprüft, ob auch sie ein Kernobjekt darstellen. Dies trifft auf Objekt m zu. Dies wird für alle dichte-erreichbaren Objekte in diesem Cluster durchgeführt, sodass am Ende alle rot dargestellten dichte-erreichbaren Objekte zu einem Cluster gehören.

Dichtebasierte Clusterverfahren zeichnen sich dadurch aus, dass die Clusteranzahl nicht angegeben werden muss, sondern alle Cluster anhand der Dichte automatisch erkannt werden. Daher werden auch Cluster beliebiger Form erkannt. Ebenso können Ausreißer identifiziert werden. Sie beeinflussen das Clusterergebnis folglich nicht. Der DBSCAN Algorithmus hat allerdings eine hohe Laufzeit von $O(n^2)$, was als Nachteil dieses Clusterverfahrens genannt werden muss. [10]

2.5. Entwicklungsumgebung OpenCV

Die **Open Computer Vision Library (OpenCV)** ist eine Open Source Bildverarbeitungsbibliothek, die eine Plattform für computerunterstütztes Sehen und maschinelles Lernen bietet. OpenCV bietet mehr als 2500 Algorithmen die beispielsweise für die Gesichtserkennung, Objektidentifikation, Objektnachverfolgung oder aber auch für das Zusammenfügen von Bildern zu einer hochauflösenden Bildszene verwendet werden können. Einige hochrangige Firmen wie Google, Microsoft, Intel, IBM, Sony und Honda entwickelten diese Bibliothek. Hauptsächlich wird sie von Firmen, Forschungseinrichtungen und Regierungsbehörden genutzt. Sie basiert auf C++, bietet aber auch C, Python, Java und Matlab Schnittstellen und ist für Windows, Linux, Android und Mac OS verfügbar.

Zuerst sollen grundlegende API[8]-Konzepte von OpenCV vorgestellt werden. Anschließend wird auf vorhandene Datentypen und Bibliotheken eingegangen, die Algorithmen bereitstellen, um Entwickler in gewissen Anwendungsgebieten zu unterstützen. [1]

2.5.1. API Konzepte

OpenCV stellt alle Klassen und Funktionen in seinem **Namespace cv** zur Verfügung. In Listing 2.3 ist dieser dargestellt. Variante 1 beschreibt, dass die Matrix *Matm* im Namespace cv zu finden ist, ebenso wie die Funktion *findHomography(point1, point2, CV_RANSAC, 5)*. Variante 2 nutzt die using Direktive, die es erlaubt in dem Dokument alle Funktionalitäten des Namespaces cv zu verwenden ohne diesen explizit angeben zu müssen.

Listing 2.3: API Konzept: Namespace

```
1 // Variante 1:
2 cv:: Mat m = cv::findHomography(point1, point2, CV_RANSAC, 5);
3
4 // Variante 2:
5 using namepsace cv;
6
7 Mat m = findHomography(point1, point2, CV_RANSAC, 5);
```

Ein weiteres Konzept von OpenCV ist, dass die Speicherverwaltung von OpenCV selbst realisiert wird. Es handelt sich daher um eine **automatische Speicherverwaltung**. Es werden für Datenstrukturen, wie Vektoren oder Matrizen, die Speicherbereiche (Speicherbuffer) durch die jeweiligen Destruktoren automatisch freigegeben, wenn sie nicht mehr benötigt werden. Der Destruktor gibt den Speicherbereich allerdings nicht sofort frei, sondern verkleinert erst einen Referenzzähler der genutzten Datenstruktur, der angibt wie viele andere Datenstrukturen auf diesen Speicherbereich zugreifen. Erst wenn dieser null erreicht, wird der Buffer freigegeben.

[8] API: Application Programming Interface

In OpenCV werden Pixel in 8 - oder 16 - bit Bereichen pro Kanal dargestellt. Diese repräsentieren die Sättigung beziehungsweise Farbintensität. Durch Konvertierung in verschiedene Farbräume oder Anpassungen der Helligkeit und des Kontrastes können diese Wertebereiche leicht überschritten werden. Beispielsweise kann bei einem 8 - bit Bild der Wertebereich von 0 bis 255 überschritten werden, was zu Artefakten bei weiteren Analysen führen kann, woraus wiederum Probleme entstehen können. Daher bietet OpenCV die sogenannte **Sättigungsarithmetik**, die automatisch kontrolliert, ob die Wertebereiche eingehalten werden. Im Falle der Über- oder Unterschreitung, wird das Ergebnis zum nächst gelegenen Wert im gültigen Wertebereich konvertiert.

Das letzte hier vorgestellte Konzept von OpenCV ist die "Proxy" Klasse **InputArray und OutputArray**. Für einige Funktionalitäten werden in OpenCV mehrdimensionale Arrays verwendet. Diese können neben Matrizen aber auch Vektoren sein. Damit innerhalb der API Funktionalitäten nicht doppelt implementiert werden muss, werden die "Proxy" Klassen *InputArray* und *OutputArray* eingeführt. Diese können Matrizen, Vektoren oder auch skalare Datentypen sein. OpenCV erkennt dadurch automatisch, welche Datentypen benutzt werden. Dies ermöglicht es flexibel die zu verwendenden Datentypen auszuwählen. [1]

2.5.2. Bibliotheken

2.5.2.1. core Bibliothek

Die *core* Bibliothek bietet grundlegende Datenstrukturen, wie zum Beispiel das Array *Mat*, und Funktionen, die von allen anderen Modulen verwendet werden. Im nächsten Schritt sollen die wichtigsten Datentypen kurz vorgestellt werden:

- **Point:** Es gibt Template Klassen für 2D und 3D Punkte. Die Koordinaten des 2D Punktes werden beispielsweise durch x und y Attribute beschrieben. Durch einen cast-Operator kann der Datentyp festgelegt werden. Die Koordinaten des Punktes können Integer, Float oder Double Werte darstellen.
- **Size:** stellt einen Datentyp dar, der die Größe eines Bildes oder Rechtecks beschreibt. Er besitzt zwei Attribute, *width* und *height*, die die Größe beschreiben.
- **Rect:** Rechtecke werden in OpenCV durch die Datenstruktur *Rect* beschrieben. Er besitzt als Attribute die Koordinaten der linken oberen Ecke, sowie die Breite und Höhe des Rechtecks.
- **Scalar:** ist ein vier elementiger Vektor. Dabei kann der Scalar-Vektor durch eine dynamische Anpassung der Größe flexibel genutzt werden, d.h. er kann nur zwei Elemente nutzen, oder aber auch drei oder vier. Typischerweise wird er verwendet, um Pixel- oder Farbwerte zu übergeben. Er kann zum Beispiel die RGB Farben, also die drei Farbanteile, enthalten und so einen RGB Pixel beschreiben. In einigen Funktionen wird ein Parameter *color* verlangt, der einen Scalar-Vektor darstellt. Dabei gibt der erste Kanal den blauen Farbwert an, gefolgt von grün und rot.

- **Mat:** Der Datentyp *Mat* stellt Matrizen in OpenCV dar und ist einer der zentralen Datentypen. Er repräsentiert ein n-dimensionales Array, das entweder einen oder mehrere Kanäle besitzen kann. Dies kann dafür genutzt werden Graubilder (engl. "grayscale image") oder Farbbilder zu repräsentieren und wird durch den Farbraum der Matrix festgelegt. *Mat* kann n Dimensionen enthalten, daher können damit auch 3D Bilder verwaltet werden. Am Häufigsten werden Bilder aber in einem 2-dimensionalen *Mat* Array gespeichert. Matrizen können durch Angabe der Zeilen und Spalten, sowie des Matrizentyps erstellt werden. Die Größe der Matrix wird durch ihre Zeilen- und Spaltenanzahl repräsentiert. Die Zeilen spiegeln die y Koordinaten (Höhe) und die Spalten die x Koordinaten (Breite) eines Bildes wieder. Der Typ beschreibt die Art, wie ein Element des *Mat* Arrays interpretiert werden soll. Er beschreibt die Bitbreite, den Datentyp und die Anzahl der genutzten Kanäle für die Darstellung eines Pixels. Beispielsweise stellt der Typ CV_8UC1 eine Matrix mit 8 - bit unsigned int und einem Kanal dar. Der Typ CV_32FC2 dagegen beschreibt eine Matrix mit 32 - bit mit zwei Gleitkomma (engl. floating-point) Kanälen. Dadurch ist es möglich die Matrizen individuell, in Abhängigkeit des jeweiligen Anwendungsgebietes anzupassen.

Neben diesen Standarddatentypen bietet die Bibliothek *core* auch Funktionen, die auf Arrays ausgeführt oder für das Zeichnen von Bildern verwendet werden können. Im Weiteren sollen die wichtigsten vorgestellt werden:

- **absdiff:** berechnet elementweise die Differenz zwischen zwei Arrays. Dadurch kann ein Differenzbild von zwei aufeinanderfolgenden Bildern erzeugt werden, sodass lediglich die Bewegungen des Bildes dargestellt werden.
- **perspectiveTransform:** führt eine perspektivische Transformation von 2D oder 3D Vektoren durch. Damit kann die Perspektive eines Vektors durch Angabe einer Homographie Matrix (siehe Kapitel 2.2) angepasst werden.
- **circle:** zeichnet einen Kreis in ein angegebenes Bild (Matrix *Mat*). Dabei müssen die Koordinaten des Kreises, der Radius, die Farbe, die Dicke und der Linientyp angegeben werden.
- **line:** zeichnet eine Linie zwischen zwei vorgegebenen Punkten. Dabei können Farbe, Dicke und Linientyp beeinflusst werden.

Weiterhin wird auch ein partitionierendes *k-means* Clusterverfahren durch die Methode *kmeans(InputArray data, int k, InputOutputArray bestLabels, TermCriteria criteria, int attempts, int falgs)* bereitgestellt. Der Parameter *data* enthält die zu clusternden Daten als Array von n-dimensionalen Punkten als Gleitkommazahlen. k gibt die Anzahl der Cluster an, in die die Objekte eingeteilt werden sollen. *bestLabels* speichert für jeden Punkt die Clusterzugehörigkeit ab. *criteria* gibt das Kriterium für die Beendigung des *k-means* Algorithmus an. Wenn ein bestimmter Wert unterschritten wird, wird der Algorithmus beendet. *attempts* gibt an, wie oft der Algorithmus mit einer unterschiedlichen Initialisierung gestartet werden soll, um dadurch das beste Clusterergebnis zu erhalten. Die Festlegung der initial festgelegten Zentren der k Cluster wird durch den Parameter *flags* bestimmt. [1]

2.5.2.2. Bildverarbeitung: imgproc Bibliothek

Das Modul *imgproc* stellt Funktionalitäten zur Verfügung um Bilder zu verarbeiten. Dies können Filter sein, um beispielsweise ein Bild mit einem Gaußfilter zu glätten, oder auch Transformationen, wie resize, um das Bild zu vergrößern.

Die wichtigsten Funktionen der *imgproc* Bibliothek sind im Folgenden als Überblick dargestellt:

- **dilate:** führt eine morphologische Dilatation auf ein Bild aus.
- **erode:** führt eine morphologische Erosion auf ein Bild aus.
- **GaussianBlur:** glättet ein Bild in dem ein Gaußfilter genutzt wird.
- **resize:** verändert die Größe des Bildes.
- **getPerspectiveTransform:** liefert die Homographie Matrix zurück, die sich aus den beiden angegebenen Matrizen ergibt.
- **warpPerspective:** führt eine perspektivische Transformation einer Matrix durch. Damit kann die Perspektive einer Matrix durch Angabe einer Homographie Matrix verändert werden. In dieser Arbeit wird damit beispielsweise ein perspektivisches Bild in Vogelperspektive in ein Bild in Draufsicht umgewandelt.
- **cvtColor:** konvertiert ein Bild in einen anderen Farbraum.
- **findContours:** findet alle zusammenhängende Konturen eines Binärbildes in Form einer Aneinanderreihung von Punkten. Die Eingabematrix muss aus einem einkanaligen 8 - bit Bild bestehen. Sämtliche Pixel, die ungleich null sind, werden dabei als eine 1 interpretiert, sodass das Bild nur aus 0 und 1 besteht. Es stellt also ein Binärbild dar. Dabei geht *findContours()* nach dem Algorithmus von Suzuki, der in seinem Paper [16] beschrieben ist, vor. Die Methode liefert als Ergebnis einen *vector* zurück, der alle Konturen enthält. Jede Kontur ist wiederum als *vector* dargestellt, der die Punkte, die die Kontur aufspannen, enthält. Weiterhin wird die Hierarchie der Konturen in einer eigenen Matrix abgespeichert, die jede Kontur einer Hierarchieebene zuordnet. Durch den Parameter *method* können die Konturen komprimiert werden. Der Wert *CV_CHAIN_APPROX_SIMPLE* gibt an, dass horizontale, vertikale und diagonale Kontursegmente als Linie mit Anfangs- und Endpunkt dargestellt werden können. Die Punkte dazwischen werden dabei weggelassen.
- **approxPolyDP:** glättet eine Kurve anhand einer vorgegebenen Genauigkeit. Die Punkte einer Kurve werden entsprechend der angegebenen Genauigkeit mit Hilfe des *Ramer-Dobulas-Peucker Algorithmus* entfernt.
- **contourArea:** berechnet die Fläche einer Kontur.
- **canny:** findet, anhand des in Kapitel 2.1.3 beschriebenen Algorithmus, alle Kanten in einem Bild. Das Eingangsbild wird in das Ausgangsbild umgewandelt. Das Ausgangsbild ist ein binäres Bild, das die detektierten Kanten, die innerhalb der beiden *threshold* Parameter liegen, weiß darstellt. Alle anderen Pixel, die keine Kanten repräsentieren, sind

schwarz. Es gibt also nur zwei Werte. Dabei sind zwei Parameter *thresholdMin* und *thresholdMax* entscheidend für die Detektion von Kanten. Der Parameter *thresholdMax* gibt den niedrigsten Schwellwert des Gradienten für starke Kanten an. Alle Punkte im Bild, deren Gradientenwert höher als dieser Schwellwert ist, werden als starke Kante angesehen. Ausgehend von diesen, wird eine Kantennachverfolgung gestartet. Diese wird mit dem Parameter *thresholdMin* gesteuert. Dieser kleinste Schwellwert gibt an, wie hoch der Wert des Gradienten für die Verbindung der starken Kanten mindestens sein muss. Liegt der Gradient oberhalb des Schwellwertes, so wird die Kante, ausgehend von einer starken Kante, weiterverfolgt. Wenn nicht, dann endet der Kantenverlauf in diesem Punkt und es wird bei einer anderen starken Kante neu begonnen. OpenCV empfiehlt ein Verhältnis von minimalen zu maximalen Thresholdwerten von 1 : 2 bis 1 : 3.

- **HoughLines:** findet Linien eines Binärbildes anhand einer Hough Transformation[9]. [1]

2.5.2.3. High Level GUI und Media I/O: highgui Bibliothek

Diese Bibliothek bietet Funktionalitäten für eine GUI (Graphical User Interface), sowie für Interaktionen mit dem User, die sowohl mit der Maus als auch mit der Tastatur erfolgen können. Weiterhin ermöglicht sie es, Bilder oder Videos auf die Festplatte zu schreiben oder von dieser zu lesen. Sie stellt außerdem einige Funktionalitäten für die Verwendung von Codecs für Bilder und Videos zur Verfügung.

Anschließend sollen einige Funktionen dieser Bibliothek erwähnt werden:

- **imshow:** zeigt ein Bild in einem Fenster an.
- **waitKey:** wartet auf eine Tastatureingabe und gibt den Code der gedrückten Taste zurück.
- **imread:** liest ein Bild aus einer Datei, dessen Dateipfad angegeben ist.
- **imwrite:** schreibt ein Bild in eine Datei.
- **VideoCapture:** ermöglicht es Videos einzulesen und abzuspielen. VideoCapture bietet mit *read* eine Funktion, die das nächste Bildframe des Videos ausliest. Dadurch ist es möglich das Video bildweise zu verarbeiten.
- **VideoWriter:** schreibt Videos auf die Festplatte unter Angabe des Video Codecs. [1]

2.5.2.4. Kamera Kalibrierung und 3D Rekonstruktion: calib3d Bibliothek

Die calib3d Bibliothek stellt unter anderem Algorithmen für unterschiedliche geometrische Ansichten, sowie 3D Rekonstruktionen zur Verfügung. Sie benutzt dafür das Lochkamera Modell und stellt 3D Punkte durch eine perspektivische Transformation auf der Bildebene dar. [1]

[9] Hough Transformation: Verfahren zu Erkennung von Geraden oder Kreisen in einem Bild. [15, S.212]

2.5.2.5. Video Analyse: video Bibliothek

Das Modul *video* stellt Algorithmen für die Videoanalyse, beispielsweise zur Vorhersehung von Bewegungen, Hintergrundsubtraktion und Objektnachverfolgung, bereit. Ein Teil dieser Bibliothek ist auch eine Implementierung des Kalman Filters, der für die Bewegungsvorhersage genutzt werden kann. [1]

3. Forschungsstand und Hypothesen

Bevor mit der eigenen Arbeit begonnen wird, soll der aktuelle Forschungsstand bei der Ballerkennung von Tennisspielen betrachtet werden. Darunter sind einige Artikel, die sich ebenfalls mit Videoanalysen von Fernsehaufnahmen beschäftigen und dabei versuchen mit der vorhandenen Videoqualität eine möglichst gute Ballerkennung zu ermöglichen. Andere Artikel wiederum nutzen eigene Videoaufnahmen, um die Ballerkennung zu optimieren.

Eine dieser wissenschaftlichen Arbeiten ist von Qazi et al. [12], der für die Bildgewinnung einen Quadrocopter einsetzt. Die dadurch gewonnenen Bilder werden zuerst stabilisiert, sodass die unregelmäßigen Bilder des Quadrocopters verbunden werden und anschließend weiter untersucht werden können. Hierbei wird die Farbe des Balles verwendet, um ihn gegenüber dem Hintergrund herauszufiltern. Laut Qazi et al. [12] konnte er 94% der Bälle erkennen. Da der Einsatz von Quadrocoptern für eine grundlegende Taktikanalyse von Tennisspielen nicht praktikabel ist, soll auf diese Funktionsweise nicht genauer eingegangen werden.

In dem von Pingali et al. verfassten Paper [4] wird die Ballerkennung ebenfalls mit eigenen Videoaufnahmen realisiert. Er beschreibt darin ein echtzeitfähiges Verfahren, das Spieler und Ball nachverfolgen kann, dafür werden allerdings Videokameras benötigt, die eine hohe Bildfrequenz erlauben. Für eine verbesserte Erkennung des Spielers wird darin ein System verwendet, das vier Kameras nutzt. Für die Erkennung des Spielers und des Balles werden die Bewegungen durch Differenzbilder erkannt, die anschließend durch Thresholding hervorgehoben und mit morphologischen Operatoren optimiert werden. Weiterhin werden zur Erkennung des Spielers durch Feature Detection[1] eindeutige Merkmale der registrierten Objekte identifiziert, die durch Clusterfahren zusammengefasst werden können. Dadurch können Spieler erkannt und durch Nachverfolgung der Feature-Punkte auch die Bewegung nachverfolgt werden. Für die Ballerkennung wird nicht nur das durch Thresholding gewonnene Binärbild untersucht, sondern auch die Farben der darin erkannten Bereiche. Diese werden mit der Farbe und Größe des Balles verglichen. Die Bereiche mit bester Übereinstimmung in Farbe und Größe werden als Ball angenommen. Allerdings konnte dieses Verfahren nicht bei einem Wettkampfspiel getestet werden, da die Bildfrequenzen der aufzeichnenden Kameras zu niedrig waren. Daher wurde dieses Verfahren nur in einem Selbstversuch getestet, bei dem der Ball mit durchschnittlich $60km/h$ durch den Bildbereich der Kamera flog. In diesem Fall konnte der Ball gut erkannt werden. [4]

Yu et al. [23] dagegen beschreibt ein Verfahren, das den Ball auch bei Fernsehaufnahmen entdeckt und nachverfolgen kann. Er betrachtet hier allerdings nur Bilder auf denen das Spielfeld komplett zu sehen ist und geht davon aus, dass die Videos so geschnitten sind, dass Werbung

[1] Feature Detection: befasst sich mit der Erkennung von Merkmalen eines Bildes. Die Merkmale sollen dabei das Bild möglichst aussagekräftig beschreiben. [19]

und Nahaufnahmen aus der Videoaufnahme entfernt wurden. Er erkennt dabei den Ball, indem er soweit wie möglich alles ausschließt, was nicht der Ball sein kann. Es werden beispielsweise Objekte, die deutlich größer als der Ball sind, entfernt, oder aber auch Objekte die nicht die gleiche Farbe als der Ball besitzen. Auch die Form wird mit einbezogen. Die übriggebliebenen Objekte werden als Ballkandidaten angenommen und werden mit der zuvor erkannten Ballflugkurve verglichen. Durch die Berechnung der Flugkurve des Balles können auch Positionen des Balles ermittelt werden, bei der der Ball mit Hilfe von Bildverarbeitungsverfahren nicht erkannt werden kann. Er beschreibt hier den Fall, dass der Ball beim Spieler auf der hinteren Spielfeldseite nur noch sehr klein ist und daher nicht erkannt werden kann. Weiterhin wurde durch Analyse der Audiosignale der Moment des Schlages erkannt, wodurch die Balldetektion zusätzlich verbessert werden kann. Mit dieser Vorgehensweise entdeckt der Verfasser der Arbeit bis zu 96% der Bälle in Fernsehaufnahmen, wobei hier nicht die Position bei Bodenkontakt erfasst wurde, sondern nur die grobe Position beim Balltreffpunkt des Spielers.

Ein weiteres Verfahren das sich besonders auf Fernsehaufnahmen mit geringer Auflösung, die im Internet verfügbar sind, konzentrierte, erzielte bei der Ballerkennung einer geschnittenen Fernsehaufnahme eine durchschnittliche Precision von 83.7% und Recall von 82%. Dieses Verfahren wird in [17] beschrieben. Es erkennt zum einen die Spieler anhand eines Mean Shift Algorithmus[2], das das Spielercluster aufgrund eines Farbvergleiches dieses Clusters mit dem Spielfeldhintergrund erkennt. Zum anderen wird der Ball durch Analyse von Differenzbildern entdeckt. Diese werden anschließend durch morphologische Operationen optimiert, sodass die Objekte anhand der Ballgröße und der entdeckten Position gegebenenfalls als Ball klassifiziert werden. Des Weiteren können die Balltreffpunkte identifiziert werden, da sowohl die Bewegung der Spieler als auch die des Balles mitverfolgt werden. Dies ermöglicht die Ermittlung des Ballltreffpunktes durch Schnittpunkte der Bewegungskurven. [17]

All diese Verfahren beschäftigen sich mit der Erkennung von Spieler und Ball im Tennis. Einige erzielten bereits sehr gute Ergebnisse, nutzten dafür allerdings eigene Kameras wie in [12] und [4], die eine sehr gute Videoqualität boten und somit den Ball sehr gut erkannten. Die Verfahren dagegen, die auf Fernsehaufnahmen basieren, wie beispielsweise von [23] und [17], vereinfachen die Detektion des Balles durch vorheriges Schneiden der Videos, sodass lediglich die Situationen analysiert werden müssen, in denen der Ball identifiziert werden kann. Dadurch erreichen auch sie gute Ergebnisse bei der Ballerkennung.

Auf Grundlage dieser Informationen soll in dieser Arbeit untersucht werden, wie zuverlässig der Ball bei ungeschnittenen Fernsehaufnahmen entdeckt werden kann. Der Ball soll in den Videos automatisch erkannt werden. Werbepausen und Nahaufnahmen von Spielern oder Zuschauern sollen die Bilderkennung nicht stören. Sie sollen lediglich ausgeblendet werden, sodass der Ball wiedererkannt werden kann, wenn die Kamera auf das ganze Spielfeld gerichtet ist. Insbesondere soll analysiert werden wie stark die Ballerkennung durch verschiedene Spielfeldbeläge beeinflusst wird. Erhebliche Differenzen in der Erfolgsquote sind bei Untergründen, die sich farblich nicht stark von der Farbe des Balles abheben, wie ein trockener, also sehr heller, Sandplatz, zu er-

[2]Mean Shift Algorithmus: ist ein Verfahren dass das Maximum einer Dichte Funktion bestimmen kann[15, S. 258ff]

warten. Weiterhin ist von großen Unterschieden bei statischem oder dynamischem Kamerafokus auszugehen. Es ist anzunehmen, dass bei erhöhter Kamerabewegung die Ballerkennung erschwert wird. Ebenso wird die Videoqualität wesentlich dazu beitragen, ob der Ball gut erkannt wird, da er nur $6.5cm$ klein ist und sich auf einem $23.77m$ langem Spielfeld bewegt. Dadurch wird er vor allem im weiter von der Kamera entfernten Bereich des Spielfeldes nur sehr klein dargestellt, was die Erkennung des Balles erschwert. Diese Szenarien sollen in Kapitel 5 genauer betrachtet werden. In dessen Untersuchung liegt das Hauptaugenmerk dieser Arbeit.

4. Implementierung

In diesem Kapitel werden Überlegungen und Konzepte für die Spielfeld- und Ballerkennung, sowie dessen Umsetzung beschrieben. Es wird zuerst auf theoretische Überlegungen eingegangen und übersichtlich dargestellt. Im Anschluss werden Auffälligkeiten und Probleme bei der Umsetzung aufgezählt und erläutert. Als Übersicht für dieses Kapitel wird zuerst der Ablauf der Ballerkennung in Diagramm 4.1 dargelegt.

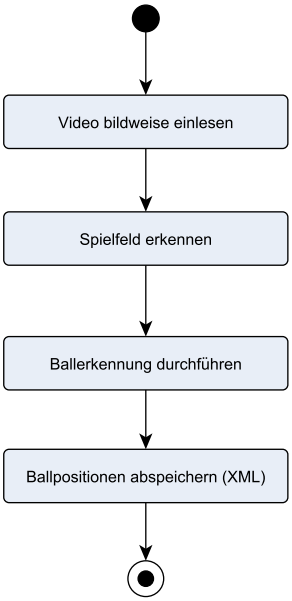


Abbildung 4.1.: Übersicht des Programmablaufes

Darin ist zu erkennen, dass das Video zuerst bildweise eingelesen werden muss. Anschließend erfolgt die Spielfelderkennung, durch die festgestellt wird, wann der Ball zu erkennen ist und wann nicht. In dieser Arbeit soll der Ball in Fernsehaufnahmen von Tennisspielen erkannt werden können, die auch längere Pausen zwischen den einzelnen Ballwechseln enthalten. Dort ist meist kein Spielfeld zu finden. Erst wenn ein Spielfeld identifiziert wurde, kann der Ball erfolgreich erkannt werden. Bei erfolgreicher Ballerkennung muss die festgestellte Ballposition abgespeichert werden. Am Ende sollen die Ballwechsel in einer XML-Datei abgebildet sein.

Für all diese Schritte sind Verfahren aus der Bildverarbeitung notwendig. Daher wurde die OpenCV Bibliothek verwendet, die in Kapitel 2.5 genauer beschrieben ist. In dieser Arbeit

wurde die OpenCV Version 2.4.11 vc10 verwendet, die den Einsatz von OpenCV mit Visual Studio 2010 ermöglicht. In den nächsten Abschnitten sollen die eben vorgestellten Schritte dieser Arbeit genau beschrieben werden.

4.1. Einlesen des Videos

Wie in Grundlagenkapitel 2.1.2 beschrieben, bestehen Videos aus einzelnen Bildern. Diese Bilder können durch Verfahren der Bilderkennung untersucht werden. Um das Spielfeld und den Ball aus den einzelnen Bildern zu gewinnen, muss das Video eingelesen und Bild für Bild ausgewertet werden. Hierfür bietet die OpenCV Bibliothek *highgui* eine *VideoCapture* Klasse, die Videos einliest. Diese Klasse stellt, wie im Kapitel 2.5 beschrieben, Funktionalität zur Verfügung mit der ein Video bildweise eingelesen werden kann. Jedes Bild wird dabei in einer Matrix abgespeichert.

Für die Erkennung des Spielfeldes ist ein einzelnes Bild ausreichend, da dort lediglich die Konturen eines Bildes benötigt werden. Bei der Ballerkennung geht es um die Erkennung von Bewegungen. Bewegungen in Bildern können durch die Differenzen von zwei aufeinanderfolgenden Bildern sichtbar gemacht werden. Dafür stellt die Bibliothek *core* eine Methode bereit, die die absolute Differenz zwischen jedem Pixel eines Bildes berechnet und in einer neuen Matrix abspeichert. Dort sind ausschließlich die Bewegungen, wie in Bild 4.2 zu sehen, enthalten. In dem dargestellten Bild bewegt sich lediglich der aufschlagende Spieler im Vordergrund, der Ball und der annehmende Spieler im hinteren Bildbereich. Die dadurch gewonnenen Matrizen mit den dort enthaltenen Bildinformationen können anschließend für die Spielfeld- und Ballerkennung genutzt werden.

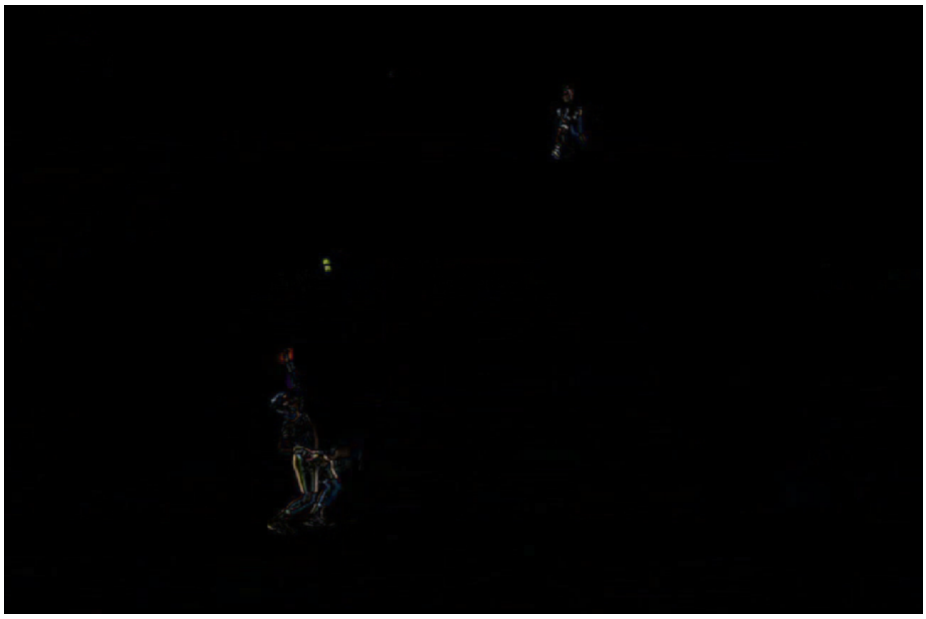


Abbildung 4.2.: Differenzbild von zwei aufeinanderfolgenden Bildern eines Videos

4.2. Erkennung des Spielfeldes

Fernsehaufnahmen von Tennisspielen werden sehr abwechslungsreich aufgenommen. Es gibt meistens mehrere Kameras, die das Spiel aus verschiedenen Perspektiven aufnehmen. Neben der typischen Großaufnahme des Spielfeldes, bei der das ganze Spielfeld durch eine Art Vogelperspektive[1] auf dem Bildschirm dargestellt wird, gibt es zahlreiche Nahaufnahmen der Spieler sowie weitere Szenen der Zuschauer oder von sonstigen Geschehnissen. Diese müssen erkannt werden oder zumindest die Ballerkennung nicht beeinflussen. Daher ist es entscheidend die Fernsehbilder herauszufinden bei dem das Spielfeld komplett erkannt werden kann. Erst nach Erkennen des Spielfeldes kann die Nachverfolgung des Balles gestartet werden. Außerdem kann durch Feststellung des Spielfeldes die Position des Balles in Relation zum Spielfeld gebracht werden.

Dies ist ein Grund wieso die Spielfelderkennung ein wichtiger erster Schritt für die Ballerkennung ist. Hierfür gibt es verschiedene Möglichkeiten das Spielfeld zu erkennen. Diese werden in den nachfolgenden Abschnitten erläutert.

Bei der Bilderkennung wird grundsätzlich versucht alle Strukturen und Objekte des Bildes zu erkennen. Die detektierten Strukturen können anschließend untersucht werden, sodass ihnen eine Bedeutung zugeordnet werden kann. Eine Spielfelderkennung kann durch verschiedene Ansätze erreicht werden, die in den nächsten Abschnitten vorgestellt werden sollen. Wurde das Spielfeld einmal erkannt, kann die Suche im nächsten Frame eingeschränkt werden, sodass die Erfolgswahrscheinlichkeit einer Erkennung des Spielfeldes steigt. Dies muss für jedes neue Bildframe durchgeführt werden. Dieses Vorgehen ist in Abbildung 4.3 dargestellt.

Weiterhin ist es bei der Erkennung des Spielfeldes ausreichend die vier Eckpunkte des Spielfeldes herauszufinden. Alle weiteren Linien und Berechnungen bezüglich der Ballposition kann aus diesen Punkten ermittelt werden. Hierfür muss das perspektivisch dargestellte Spielfeld, allerdings in eine Draufsicht konvertiert werden, da nur dadurch die offiziellen Spielfeldmaße mit dem erkannten Spielfeld kompatibel sind. Dies ist durch Verwendung einer Homographie Matrix möglich, die die perspektivischen Feldmaße in Bezug zu den offiziellen Feldmaßen setzt und dadurch das Bild in die Draufsicht transformiert.

4.2.1. Erkennung von Konturen

Zur Erkennung des Spielfeldes muss als erster Schritt die Struktur des Bildes analysiert werden. Besonders die Kanten im Bild sind hierbei von Bedeutung, da bereits wenige Kanten Objekte im Bild eindeutig repräsentieren [14, S.13]. Hierfür stehen Möglichkeiten wie die Hough Transformation[2], die Canny Edge Detection[3], die Harris Corner Detection[4] und weitere Verfahren zur

[1] Vogelperspektive: Ansicht hinter und überhalb des Spielfeldes, sodass eine Sicht von oben auf das Spielfeld möglich ist. Bei Tennisspielen ist die Kamera dabei normalerweise hinter der Grundlinie, sowie in der Mitte des Spielfeldes positioniert.

[2] Hough Transformation: Verfahren zu Erkennung von Geraden oder Kreisen in einem Bild. [15, S.212]

[3] Canny Edge Detection: Verfahren zur Kantenerkennung, siehe Kapitel 2.1.3.

[4] Harris Corner Detection: Verfahren um Ecken und interessante Punkte eines Bildes zu erkennen. [15, S.158]

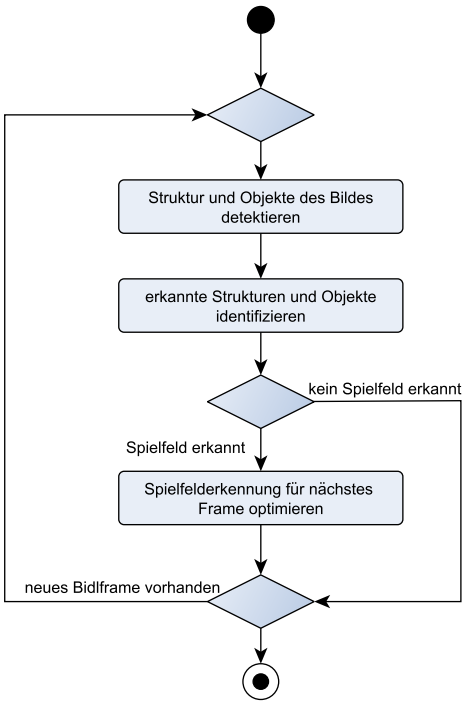


Abbildung 4.3.: Übersicht des Ablaufes bei der Felderkennung

Verfügung. Alle Ansätze haben die gleichen Ziele, nämlich Linien, Kanten oder besonders eindeutige Eckpunkte im Bild zu erkennen. Alle Verfahren haben Vor- und Nachteile die anschließend erläutert werden sollen.

Die **Hough Transformation** ist ein Verfahren zu Erkennung von Geraden oder Kreisen in einem Bild [15, S.212]. In Abbildung 4.4 ist die Anwendung einer probabilistischen Hough Lines Transforamtion[5] dargestellt, bei der alle erkannten Linien rot dargestellt wurden. Hier ist zu sehen, dass die meisten, jedoch nicht alle, Spielfeldlinien erkannt wurden, sodass ein paar Lücken bei der Spielfelderkennung übrigbleiben. Weiterhin wurden einige Linien im Zuschauerbereich erkannt. Oft werden auch Linien mehrfach dicht nebeneinander erkannt und zurückgegeben. Daher muss bei Verwendung dieses Verfahrens anschließend herausgefiltert werden welche Linien relevant sind und zum Spielfeld gehören, sowie eine Möglichkeit gefunden werden die Lücken im Spielfeld zu erkennen und zu schließen. Weiterhin sind die Linien des Spielfeldes nicht durchgehend erkannt, sondern sind teilweise überlappend oder verlaufen dicht nebeneinander, sodass solch ein Linienverbund zu einer durchgehenden Linie konvertiert werden müsste, um das Spielfeld als Rechteck zu erhalten. Ein Vorteil von der Hough Transformation dagegen ist, dass Spieler die über den Spielfeldlinien stehen dieses Verfahren nicht stört und somit lediglich die Linien des Spielfeldes erkannt werden.

[5] probabilistische Hough Line Transformation: Eine effizientere Implementierung gegenüber der Standard Hough Transformation die auf Wahrscheinlichkeiten beruht. [15, S.222]

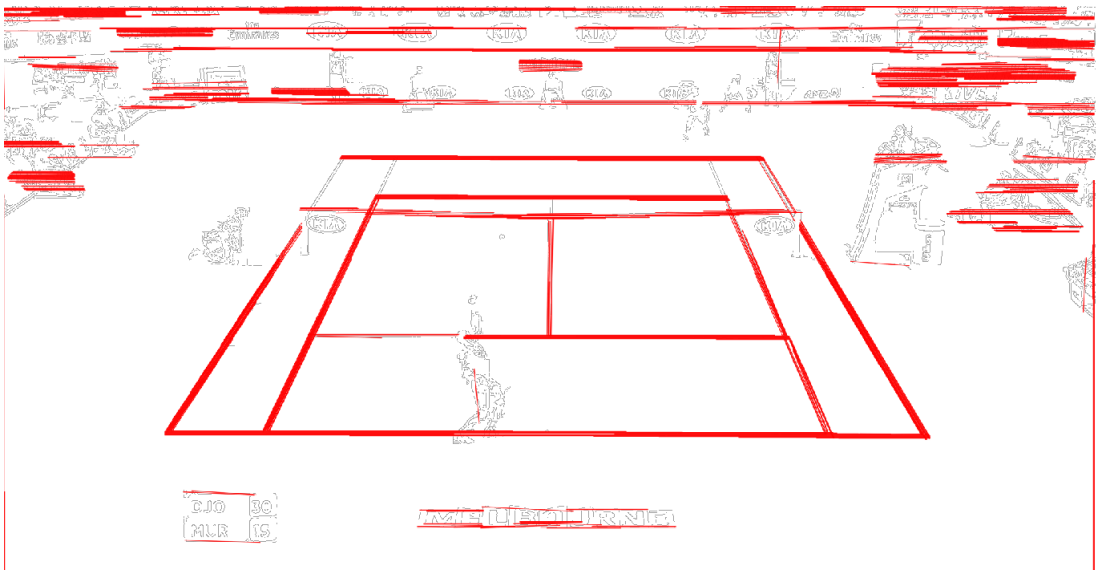


Abbildung 4.4.: Darstellung von Linien, die durch eine probabilistische Hough Line Transformation erkannt wurden

Der **Canny Edge Algorithmus** ist, wie in Kapitel 2.1.3 beschrieben, ein Verfahren, um Kanten in Bildern zu erkennen. Die *imgproc* Bibliothek bietet mit der Methode *canny(Mat src, Mat dst, double thresholdMin, double thresholdMax)* eine einfach Anwendung des Canny Edge Algorithmus. Die Funktionsweise und die Anpassung durch die Parameter ist in Kapitel 2.5.2.2 dargestellt. Weiterhin bietet OpenCV in der Bibliothek *imgproc* mit der Methode *findContours(Mat src, vector<vector<Point» contours, vector<Vec4i> hierarchy, int mode, int method)* eine Möglichkeit Kantenverläufe zu erkennen und zurückzuliefern. Auch diese ist in Kapitel kap-theorie-opencv-findcontours genauer beschrieben. In Abbildung 4.5 sind alle erkannten Konturen rot dargestellt. Diese Vorgehensweise zeichnet sich durch eine gute und vorteilhafte Erkennung des Spielfeldes dar. In den meisten Fällen ist das Spielfeld die größte Kontur im Bild, sodass das Spielfeld bereits als Ganzes erkannt werden kann. Allerdings wird durch die Methode *findContours()* immer der äußere Kantenverlauf betrachtet, sodass Störungen bei der Erkennung der Spielfeldlinien auftreten können, wenn beispielsweise der Spieler auf der Linie steht. Dies ist in Abbildung 4.5 bei dem unteren Spieler zu erkennen, der die geradlinige Kontur des Spielfeldes durch seine Füße unterbricht. Für diese Situationen müssen Verfahren gefunden werden, die den eigentlichen Linienverlauf trotz Störungen feststellen können.

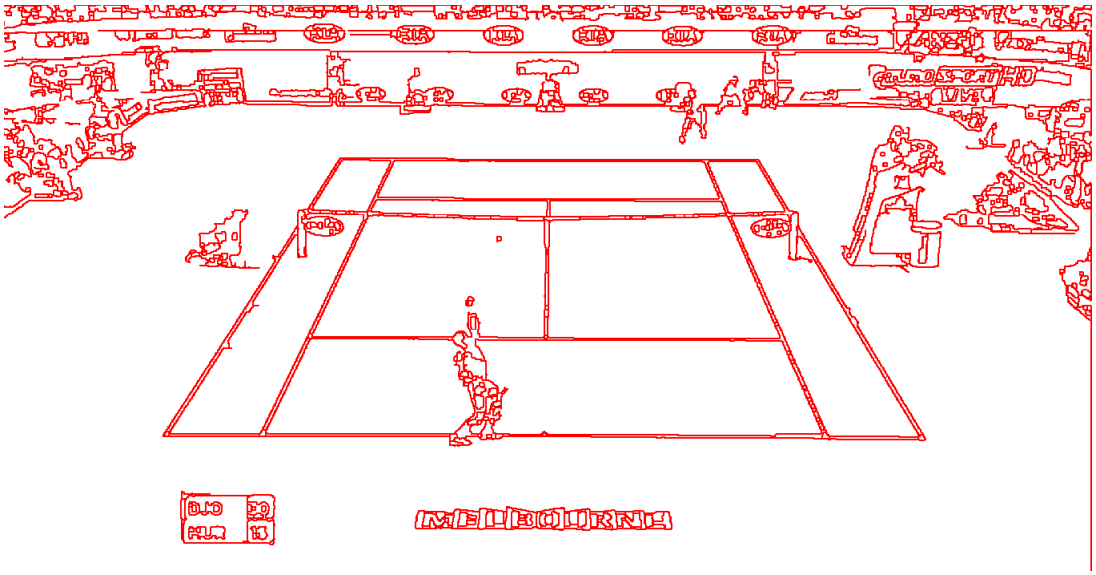


Abbildung 4.5.: Darstellung aller erkannten Konturen des Bildes

Die **Harris Corner Detection** ist ein Verfahren das Ecken und interessante Punkte eines Bildes erkennen kann [15, S.158]. In OpenCV ist sie mit der Methode *cornerHarris()* implementiert. Sie berechnet für jeden Pixel der Ursprungsmatrix einen Wert, der aussagt, wie wahrscheinlich es sich bei diesem Pixel um eine Ecke oder einen interessanten Punkt handelt. Dabei war zu erkennen, dass besonders die Schriftzüge in Bilder als interessante Punkte erkannt werden. Die Eckpunkte des Spielfeldes sind allerdings nicht so aussagekräftig. Sie werden erst bei feinerer Suche mit dem Harris Corner Algorithmus entdeckt. Weiterhin muss aus einer Vielzahl von interessanten Punkten, diejenigen Punkte, die für das Spielfeld wichtig sind, herausgefunden werden. Die Punkte geben allerdings keine Anhaltspunkte darauf, ob sie zum Spielfeld gehören oder laut der Harris Corner Detection andere interessante Punkte darstellen. Daher wird dieser Ansatz nicht weiter betrachtet.

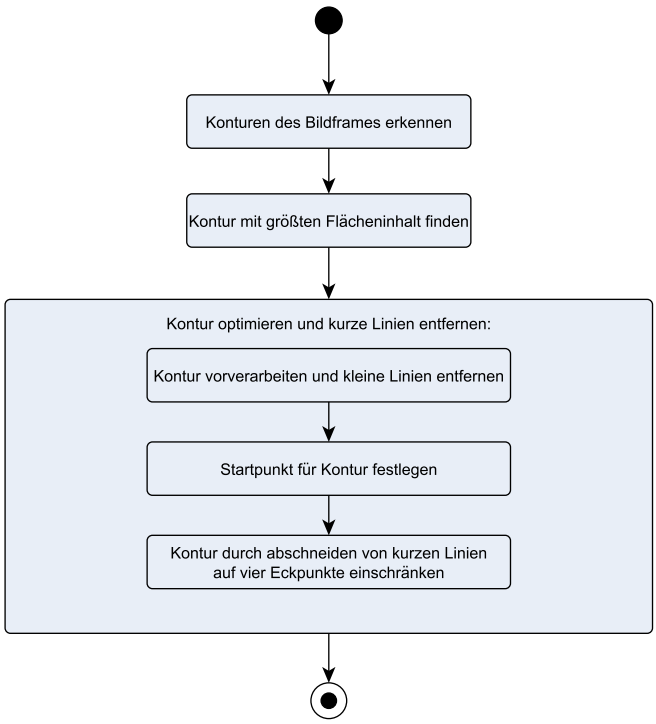


Abbildung 4.6.: Ablauf der Konturendetektion und dessen Optimierung

Besonders vielversprechend zeigte sich der Canny Edge Algorithmus mit Kantennachverfolgung, sodass dieser für die weitere Felderkennung verwendet wurde. Nachdem die Konturen des Bildes erkannt wurden geht es, wie in Abbildung 4.6 zu sehen ist, darum diese Konturen so zu bearbeiten, sodass es möglich ist, ihnen eine Bedeutung zuweisen zu können. Wie in Abbildung 4.5 zu sehen ist, werden in einem Bild viele Konturen erkannt. Das Spielfeld stellt hier die Kontur dar, die den größten Flächeninhalt besitzt. Daher werden alle Konturen durchlaufen und die Kontur mit dem größten Flächeninhalt wird genauer untersucht. Diese Annahme kann allerdings nicht immer garantiert werden, da je nach Bild, auch Konturen aus dem Zuschauerraum

wahrgenommen werden und daher unter Umständen eine größere Kontur gefunden werden kann.

Weiterhin ist eine Kontur immer die äußere Kante eines Objektes, sodass die Kontur größer wird, wenn ein Spieler auf der Linie steht. Der Spieler wird in diesem Fall miteingeschlossen, sodass nicht ausschließlich das Spielfeld erkannt wird. Hierfür muss die gefundene Kontur weiter optimiert werden, sodass am Ende das Spielfeld lediglich mit vier Eckpunkten gefunden wurde. Um dies zu ermöglich war die Idee kleine Linien in der Kontur zu entfernen, da das Spielfeld aus Linien bestehen, die mindestens eine gewisse Länge besitzen. Dafür werden zuerst sehr kleine Linien aus der Kontur entfernt. Diese kürzeren Linien entsprechen meistens den Spielern die auf der Linie stehen oder das Netz und Schiedsrichterstuhl die sich an der Seitenlinie befinden und in die Konturen des Spielfeldes miteingeschlossen werden. Dies ist in Abbildung 4.7 veranschaulicht, in der die rot dargestellte Kontur am Netz als auch beim unteren Spieler über die Spielfeldlinien hinaus erkannt werden. Diese werden durch mehrere kurze Linien repräsentiert und müssen entfernt werden, um das Spielfeld mit vier Eckpunkten zu erhalten. Ein großes Problem dabei ist, dass die Linien immer vom Ausgangspunkt zum Zielpunkt gemessen werden. Es stellt daher einen Unterschied dar von welchem Ausgangspunkt gestartet wird. Daher wird vom unteren rechten Punkt des Spielfeldes begonnen, da hier die Erfahrung zeigte, mit diesem Ausgangspunkt gute Ergebnisse zu erhalten. Ausgehend von diesem Startpunkt können nun Schritt für Schritt alle kurzen Linien entfernt werden bis die Kontur noch vier Eckpunkte besitzt. Durch dieses Vorgehen kann die Kontur des Spielfeldes gewonnen werden. Im nächsten Schritt muss diese Kontur validiert werden, sodass sie sicher einem Spielfeld zugeordnet werden kann. Dies wird im nächsten Abschnitt beschrieben.

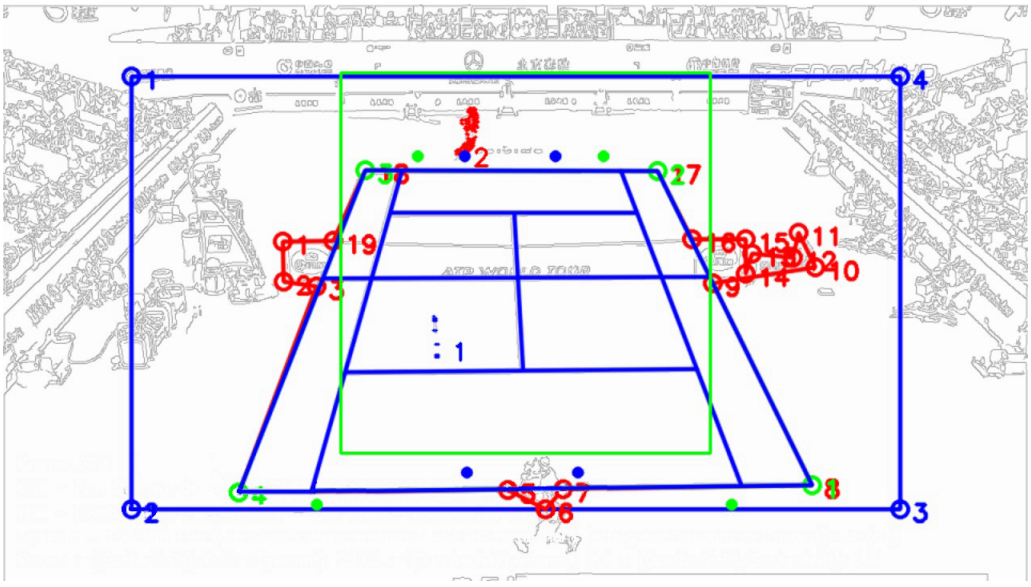


Abbildung 4.7.: Darstellung der erkannten Kontur des Spielfeldes: kurze Linien (rot) aufgrund des Spielers und des Netzes

4.2.2. Identifikation der Konturen

Im nächsten Schritt muss die erkannte Kontur identifiziert werden, sodass dieser Kontur eine Bedeutung zugewiesen werden kann. Wir wollen herausfinden, ob es sich bei der erkannten Kontur um ein Tennisfeld handelt. Dafür gibt es wie in Abbildung 4.8 zu sehen ist, verschiedene Ansätze die Kontur zu identifizieren.

1. **Flächeninhalt im Vergleich zum Bild:**
   Bei Großaufnahmen des Tennisfeldes, aus einer Vogelperspektive heraus, ist das Spielfeld meist in der Mitte des Bildes platziert. Weiterhin ist häufig das Ziel bei Fernsehaufnahmen den ganzen Ballwechsel ohne Schwenkbewegungen der Kamera aufnehmen zu können, sodass neben den Spielfeldlinien ausreichend Platz bis zum Bildrand vorhanden ist. Der Flächeninhalt des Spielfeldes im Vergleich zum Flächeninhaltes des gesamten Bildes beträgt dabei meist zwischen 20% und 35%. Sofern der Flächeninhalt der Kontur in diesem Bereich liegt, kann davon ausgegangen werden, dass es sich um ein Spielfeld handelt. Allerdings können einige Fälle eintreten bei denen auch andere Konturen fälschlicherweise als Spielfeld erkannt werden, wenn die Fläche dieser Konturen innerhalb des akzeptierenden Bereiches liegt. Dies kann dadurch entstehen, da der Bereich des Flächeninhaltes nicht zu klein gewählt werden darf, da durch kleine Unterschiede in der Kameraposition und Kamerafokussierung bei unterschiedlichen Fernsehaufnahmen dieser variieren kann. Weiterhin können auch andere Formen von Konturen, wie beispielsweise ein Kreis, zufällig solch einen Flächeninhalt aufweisen. Aus den eben genannten Gründen wird diese Vorgehensweise im weiteren Verlauf der Arbeit nicht genauer untersucht, da alleine anhand diesen Überlegungen eine richtige und zuverlässige Identifikation des Spielfeldes als unwahrscheinlich erscheint.

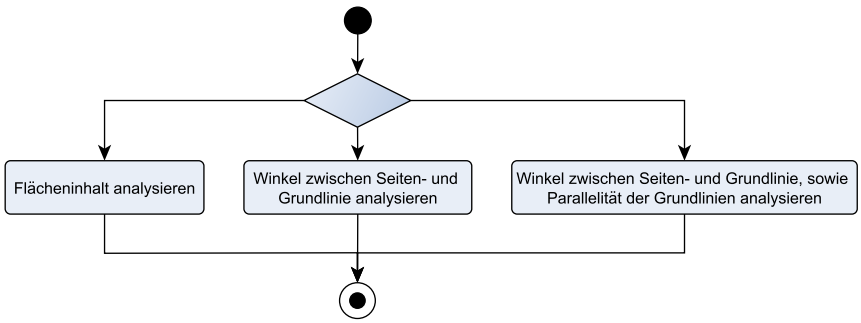


Abbildung 4.8.: Ablauf der Identifikation des Spielfeldes

2. **Winkel zwischen Seiten- und Grundlinie sind gleichmäßig:** Eine sicherere Möglichkeit das Spielfeld zu erkennen, ist die Überprüfung der Winkel zwischen Seiten- und Grundlinie. Durch die Aufnahme in Vogelperspektive und einer zentralen Position der Kamera hinter dem Spielfeld, ist davon auszugehen, dass die Winkel zwischen linker Seitenlinie und unterer Grundlinie, sowie zwischen unterer Grundlinie und rechter Seitenlinie annähernd gleich sind. Liegen diese beiden Winkel in einem Toleranzbereich von 6 Grad,

so wird die untersuchte Kontur als Spielfeld angesehen. Allerdings kann dadurch nicht sichergestellt werden, ob wirklich alle Spielfeldecken richtig erkannt wurden. Eine Spielfeldecke kann zum Beispiel nicht erkannt werden, wenn ein Spieler diese verdeckt, aber ein anderer Konturpunkt des Spielers dessen Platz einnimmt. Dadurch könnten die Winkel zwischen Seiten- und Grundlinie nicht beeinflusst, aber die Länge der Seitenlinie verkürzt oder verlängert werden.

3. **Winkel zwischen Seiten- und Grundlinie sind gleichmäßig, sowie Grundlinien sind parallel zueinander:** Eine Erweiterung des eben beschriebenen Verfahrens, das Spielfeld anhand der Winkel zwischen Seiten- und Grundlinie zu identifizieren, ist, wenn zusätzlich zu den gleichmäßigen Winkelmaßen ebenfalls überprüft wird, ob die Grundlinien parallel zueinander liegen. Dadurch ist es möglich, die typische Trapezform des Spielfeldes, die aus der Vogelperspektive der Kamera gewonnen wird, zu erkennen. Auch hier ist es zusätzlich möglich den Flächeninhalt des erkannten Spielfeldes, wie in Variante eins, zu überprüfen. Diese Variante wird in der weiteren Arbeit verwendet.

4.2.3. Optimierung der Spielfelderkennung

Nach Erkennung und Identifikation des Spielfeldes muss nicht mehr das gesamte Bild nach einem neuen Spielfeld abgesucht werden. Die Aufnahme bleibt während des Ballwechsels größtenteils gleich. Auch wird während des Ballwechsels nur selten eine andere Kamera und somit ein anderer Winkel auf das Spielfeld benutzt, sodass sich das Spielfeld im laufenden Ballwechsel nicht groß verändert. Daher kann die Suche des nächsten Spielfeldes optimiert werden.

Zum einen kann das neue Spielfeld auf einen Bereich um das zuletzt erkannte Spielfeld herum angenommen werden. Infolgedessen kann der Suchbereich für die neuen Konturpunkte auf diesen Bereich eingeschränkt werden, sodass größerer Ausreißer vermieden werden. Dieses Vorgehen löst aber nicht die Problematik, wenn die Spieler über eine Linie laufen und dadurch die Kontur des Spielfeldes vergrößert wird. Aus diesem Grund besteht die Möglichkeit den Suchbereich, in dem die neuen Konturen gesucht werden sollen, auf einen Bereich, um die bereits erkannten Eckpunkte des Spielfeldes herum zu legen. Infolgedessen kann der Bereich noch weiter eingeschränkt werden, um so fehlerhafte Detektionen sowie Identifikationen von diesen Konturen zu vermeiden. Ebenso werden die Spieler in der Mitte der Grundlinien nicht mehr erkannt, da nur der Bereich an den Eckpunkten des Spielfeldes betrachtet wird.

Diese zwei Verfahren, um das Spielfeld besser erkennen zu können, schränkt den Suchbereich ein. Daraufhin können alle Konturpunkte, die sich nicht in diesem Suchbereich befinden, entfernt werden. Deshalb gibt es zum einen wesentlich weniger Konturen die auf ein mögliches Spielfeld untersucht werden müssen und zum anderen werden nur die entscheidenden Eckpunkte des Spielfeldes betrachtet, wodurch viele störende Konturpunkte entfernt werden können. Dieses Vorgehen wird in Abbildung 4.9 dargestellt.

Eine weitere Optimierung bei der Spielfelderkennung ist, dass sich ein Spielfeld nur langsam verändert. Dies könnte beispielsweise durch heranzoomen oder seitliche Schwenkbewegungen der

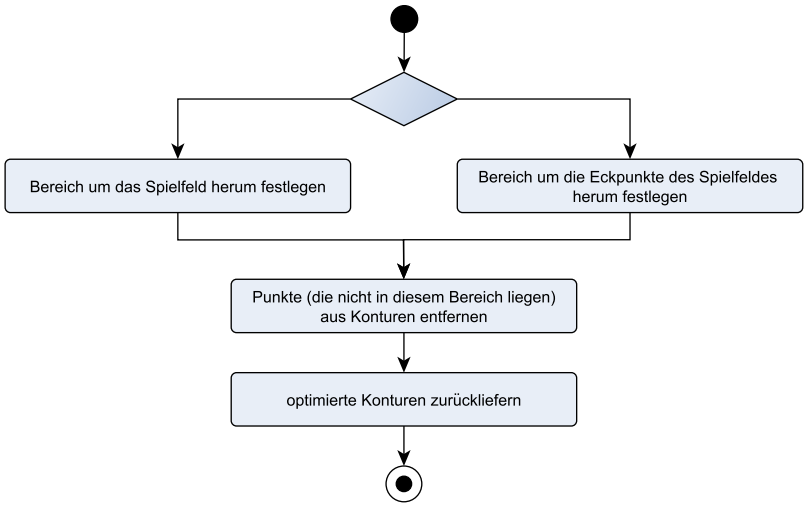


Abbildung 4.9.: Ablauf der verbesserten Erkennung des Spielfeldes

Kamera eintreten. Aus diesem Grund kann bei fehlerhafter Spielfelderkennung davon ausgegangen werden, dass das zuvor erkannte Spielfeld weiterhin gültig ist. Erst nachdem das Spielfeld 10 mal nicht identifiziert werden konnte, zählt das Spielfeld als nicht erkannt. Dadurch können kurzfristige Falschidentifikationen überbrückt werden, sodass die Ballerkennung nicht gestört wird.

4.2.4. Ermittlung der restlichen Spielfeldlinien

Nach Erkennung der vier Eckpunkte des Spielfeldes können die restlichen Spielfeldlinien, wie die Einzelseitenlinie und Aufschlaglinien, anhand der echten Spielfeldmaße ermittelt werden. Dabei wird eine Homographiematrix, ausgehend von den erkannten Spielfeldeckpunkten zu den Eckpunkten die mit Hilfe der offiziellen Spielfeldmaßen gebildet wurden, berechnet. Anschließend kann mit dieser Homographimatrix eine Draufsicht des Spielfeldes erzeugt werden, in der ohne perspektivische Verzerrung die restlichen Spielfeldlinien mit Hilfe des Strahlensatzes ermittelt werden können. Durch Benutzung der inversen Homographiematrix kann die Draufsicht wieder in die perspektivische Darstellung des Spielfeldes umgewandelt werden, die alle Spielfeldlinien enthält.

4.3. Erkennung des Balles

Das Hauptaugenmerk in dieser Arbeit liegt auf der Erkennung des Tennisballes. Dabei sollen lediglich Aufnahmen aus der Vogelperspektive, also diejenigen die hinter dem Spielfeld aufgenommen wurden und auf denen das Spielfeld komplett zu sehen ist, analysiert werden. Ebenso

sollen Fernsehaufnahmen bei denen die Kamera geschwenkt, sowie heran- oder weggezoomt wird keine Beachtung geschenkt werden. Der Schwerpunkt bei der Ballerkennung liegt bei statischen Aufnahmen deren Kameraperspektive unverändert bleibt. Die bei der Ballerkennung erkannten Positionen sollen erfasst und in einer XML Struktur abgespeichert werden, sodass sie für weitere Programme einlesbar sind. Im Speziellen soll die Prologdatenbank von Herrn Prof. Dr. Seipel diese Balldaten einlesen können.

In den nachfolgenden Abschnitten wird der Ablauf bei der Ballerkennung dargestellt und auf die Ideen und Überlegungen dahinter eingegangen. Es werden mögliche Vorgehensweisen zur Ballerkennung vorgestellt und die Vor- und Nachteile dazu erläutert, sowie auf dabei auftretende Probleme bei der Umsetzung eingegangen. Zunächst soll der Ablauf für die Ballerkennung in Abbildung 4.10 in Form eines Aktivitätendiagrammes beschrieben werden.

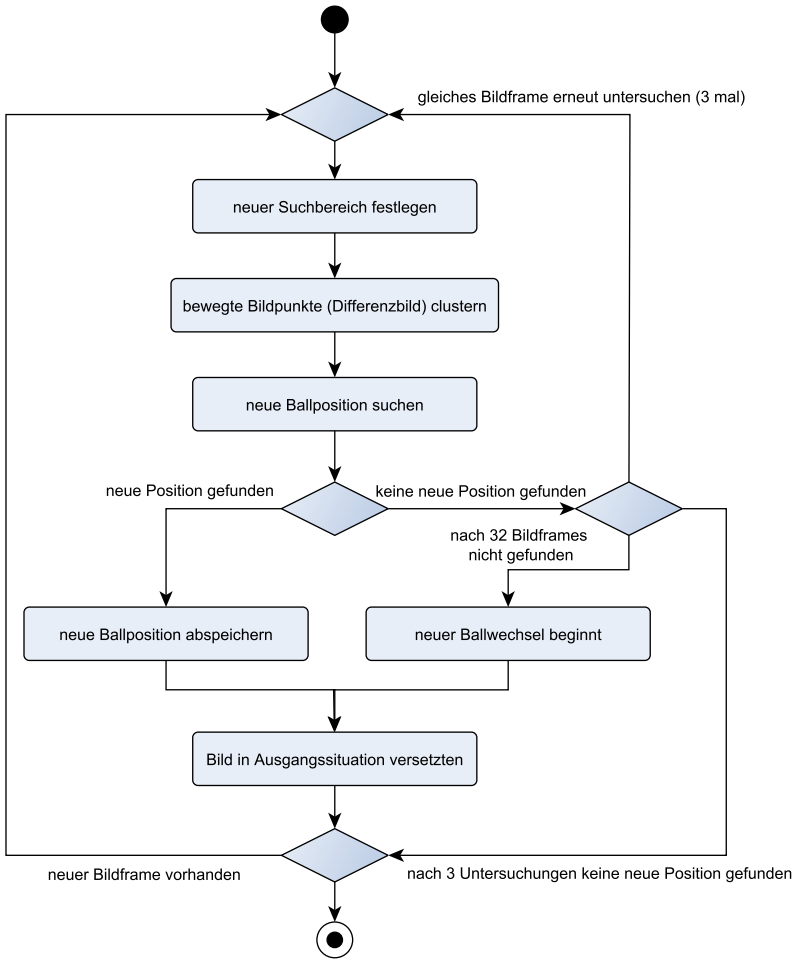


Abbildung 4.10.: Übersicht des Programmablaufes für die Ballerkennung

Darin ist zu erkennen, dass zu Beginn der Suchbereich (engl. Region of Interest ROI) im aktu-

ellen Bildframe festgelegt wird, indem der Ball vermutet wird. Dieser Ablauf wird in Abschnitt 4.3.1 genauer beschrieben und die Idee dahinter erläutert. Daraufhin werden in diesem Bereich die bewegten Bildpunkte, also alle Differenzen vom aktuellen zum vorherigen Bildframe, mit einem Clusterverfahren gruppiert (siehe Abschnitt 4.3.2). Dadurch ist es möglich Bewegungen von Spielern und Ball zu erkennen. Diese Cluster werden auf eine neue Ballposition untersucht (siehe Abschnitt 4.3.3). Sofern diese gefunden wurde, wird das nächste Bildframe eingelesen. Bei Nichterkennen einer neuen Ballposition wird das Bild erneut in einem größeren Bereich als zuvor untersucht. Die Ballposition wird bis zu dreimal in einem Bildframe gesucht, bevor das nächste Frame betrachtet wird. Dies ist notwendig da gerade bei plötzlichen Richtungswechseln des Balles, beim Schlag eines Spielers oder aber auch beim Aufspringen des Balles auf dem Boden, es vorkommen kann, dass der Ball nicht mehr im Suchbereich zu finden ist. Auch dies wird im nachfolgenden Abschnitt 4.3.1 genauer dargestellt. Im Falle, dass in 32 aufeinanderfolgenden Bildframes keine neue Ballposition gefunden wurde, wird davon ausgegangen, dass der Ballwechsel zu Ende ist. 32 Bildframes spiegeln die Zeitspanne zwischen zwei Schlägen wieder, sodass der Ball in diesem Zeitraum einmal das Feld überquert hat. Dies entspricht, ja nach Framerate des Videos, der Zeitspanne von 1.07 bis 1.28 Sekunden und spiegelt die durchschnittliche Dauer zwischen zwei Schlägen im Tennis dar. In diesem Zeitraum müsste der Ball zumindest am Netz erkannt werden, wenn er zuvor nicht mehr im Suchbereich gefunden wurde. Dieser Ablauf wird solange wiederholt bis keine weiteren Bilder des Videos vorhanden sind. Der hier dargestellte Ablauf soll in den nächsten Abschnitten genauer erläutert werden.

4.3.1. Festlegen des relevanten Suchbereiches

Zu Beginn muss der Ball in einen zuvor festgelegten statischen Suchbereich, indem er bei einer Aufschlagsituation vermutet wird, gefunden werden. Dabei wird ausgenutzt, dass sich die Positionen der Spieler in den Aufschlagsituationen meistens nur leicht unterscheiden. Der Aufschlagende steht, wie in Bild 4.11, in der Nähe eines der dunkelblauen Punkten. Der Annehmende steht diagonal gegenüber in der Nähe eines der grünen Punkte (Annahmepositionen). Sofern also beide Spieler in diesen für den Aufschlag typischen Positionen erkannt werden, kann daraus gefolgert werden, dass sich der hier rot dargestellte Ball (mit der Clusternummer 2) vermutlich in der Nähe des roten Punktes befindet. Es wird auch ausgenutzt, dass bei Aufschlagsituation klar ist, dass der Ball diagonal in einen Aufschlagbereich geschlagen werden muss. Dies ist vor allem dann interessant, wenn sich der Aufschlagende am oberen Bildrand aufhält, da sich der Ball in dieser Situation häufig bei Anwurf visuell vor den Zuschauer zu finden ist und auch vor dem Spieler fliegt, was viele Störungen durch Bewegungen der Zuschauer oder des Spielers mit sich bringt. Daraus resultiert oft eine falsche Erkennung des Balles. Der Ball soll folge dessen erst bei Überfliegen des Netzes beziehungsweise beim Erreichen des Aufschlagbereiches erkannt werden.

Ist der Ball zum ersten Mal erkannt, ist es das Ziel die Grenzen des Suchbereichs so weit wie möglich einzuschränken. Dadurch kann im Suchbereich genauer nach dem Ball gesucht werden, da dort weniger bewegte Bildpunkte, zum Beispiel weniger Bewegungen durch Spieler oder von Zuschauern, zu finden sind. In Abbildung 4.12 ist dies dargestellt, indem der

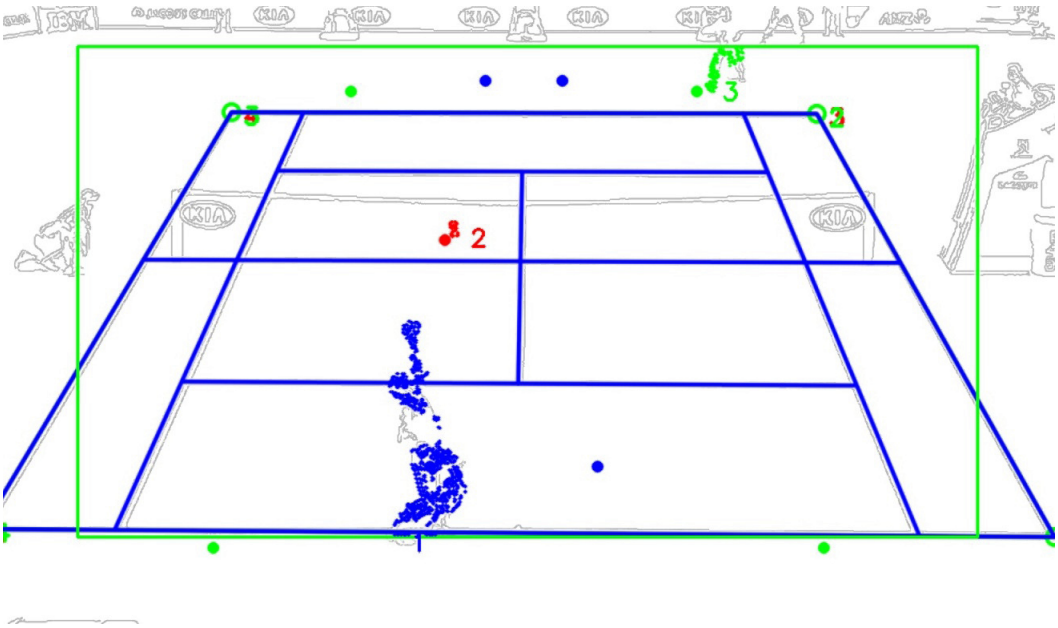


Abbildung 4.11.: Suchbereich bei Aufschlagsituation

Suchbereich (als grüner Rahmen dargestellt) eingeschränkt ist, sodass lediglich der Ball darin entdeckt werden kann. Störungen durch Spieler und Zuschauer können daher vermieden werden. Weiterhin kann durch die verminderte Anzahl an Bildpunkten in diesem Bereich diese Untersuchung performanter durchgeführt und somit Zeit eingespart werden. Die große Herausforderung liegt folglich darin den Suchbereich möglichst stark einzuschränken, sodass der Ball allerdings weiterhin in diesem Bereich liegt. Der Suchbereich wird durch die Formel $neuer\text{-}ROI = ROI + (ROI \cdot Wiederholungsfaktor \cdot AnzahlNichtErkannteBallpositionen)$ wieder vergrößert, sofern der Ball nicht mehr im Suchbereich liegt und er daher nicht gefunden werden kann (siehe Abbildung 4.20). Dies wird, wie in Diagramm 4.10 zu erkennen ist, bis zu dreimal durchgeführt. Wenn der Ball bis dahin nicht entdeckt wurde, wird das nächste Frame, wieder mit möglichst eingeschränkten Suchbereich, untersucht. Das Vergrößern des Suchbereiches wird für maximal drei aufeinanderfolgende Frames praktiziert. Falls der Ball dann immer noch nicht entdeckt wurde, wird der Suchbereich auf einen großflächigeren Bereich in die Spielfeldmitte (siehe Abbildung 4.15) erweitert, da der Ball das Netz in diesem Bereich mit sehr großer Wahrscheinlichkeit überfliegt.

In Abbildung 4.13 ist der **Ablauf für die Einschränkung des Suchbereiches** zu finden. Zu Beginn wird das gesamte Bildframe verkleinert, da aufgrund dessen weniger Bildpunkte untersucht werden müssen. Dadurch kann die Performanz bei der Ballsuche verbessert werden. Es zeigte sich, dass die Suchdauer halbiert werden kann, wenn das Bild ebenfalls um die Hälfte verkleinert wird. Allerdings hat solch eine Verkleinerung des Bildframes großen Einfluss auf die Ballerkennung, da dadurch einige Parameter neu kalibriert werden müssten. In dieser Arbeit wurde das Bild mit einem Skalierungsfaktor von 0.75 verkleinert.

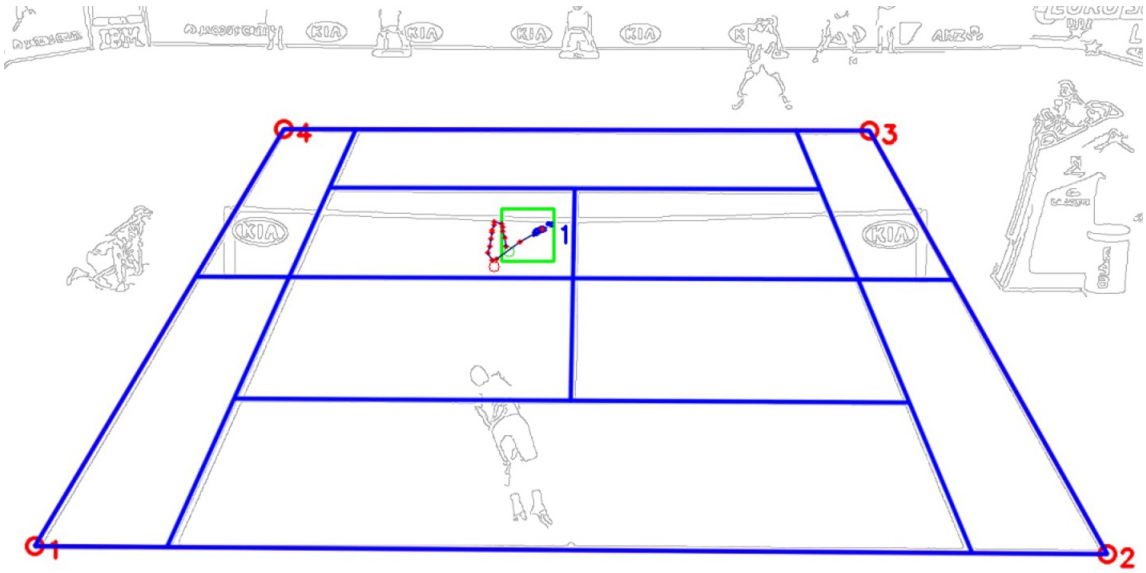


Abbildung 4.12.: Beispiel für den eingeschränkten Suchbereich (grün) nach erkannter Ballposition

Damit der Suchbereich möglichst klein bleiben kann ohne dass der Ball verloren geht, wird der **Kalman Filter** (siehe Kapitel 2.3 'Kalman Filter') verwendet. Dieser ermöglicht es die Flugkurve des Balles vorherzusehen, wodurch der Bildbereich, je nach Güte der Vorhersage, vergrößert oder verkleinert und zusätzlich in die Richtung der Vorhersage verschoben werden kann. Dafür wurde der Kalman Filter, der von OpenCV bereitgestellt wird, verwendet. Die dabei verwendete Transitionmatrix A, die die Bewegung des Objektes darstellt, wurde folgendermaßen initialisiert:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Normalerweise müsste die Transitionsmatrix A, wie in Kapitel 2.3 beschrieben aussehen, wodurch die Geschwindigkeit in Abhängigkeit von Δt in die Positionsbestimmung der Schätzung mit eingeht. Aufgrund unterschiedlicher Frameraten der Videos und der Feststellung, dass der Ball nicht in jedem Frame zu erkennen ist, wäre diese Annahme allerdings nicht richtig, da Δt hier stark variieren kann. Dies wäre nur für den Fall gültig, wenn die Framerate des Videos bekannt ist und der Ball in jedem Frame erkannt werden kann. Aus diesem Grund wird die Geschwindigkeit pro Sekunde mit einbezogen, wodurch die Anpassung des Kalman Filters träger wird. Diese Trägheit des Systems wurde dafür genutzt, den Suchbereich in die entsprechende Richtung, in der der Ball vermutet wird, zu verschieben. Auch die weiteren Parameter für die Kovarianzen Q und R wurden so angepasst, dass sich die Vorhersage des Kalman Filters nicht zu schnell an die Flugkurve und damit an den korrigierten Messwert anpasst. Die Kovarianz Q wurde nach einigen Tests dieser Parameter und durch Erfahrungswerte während der Umsetzung auf den Wert 0.01

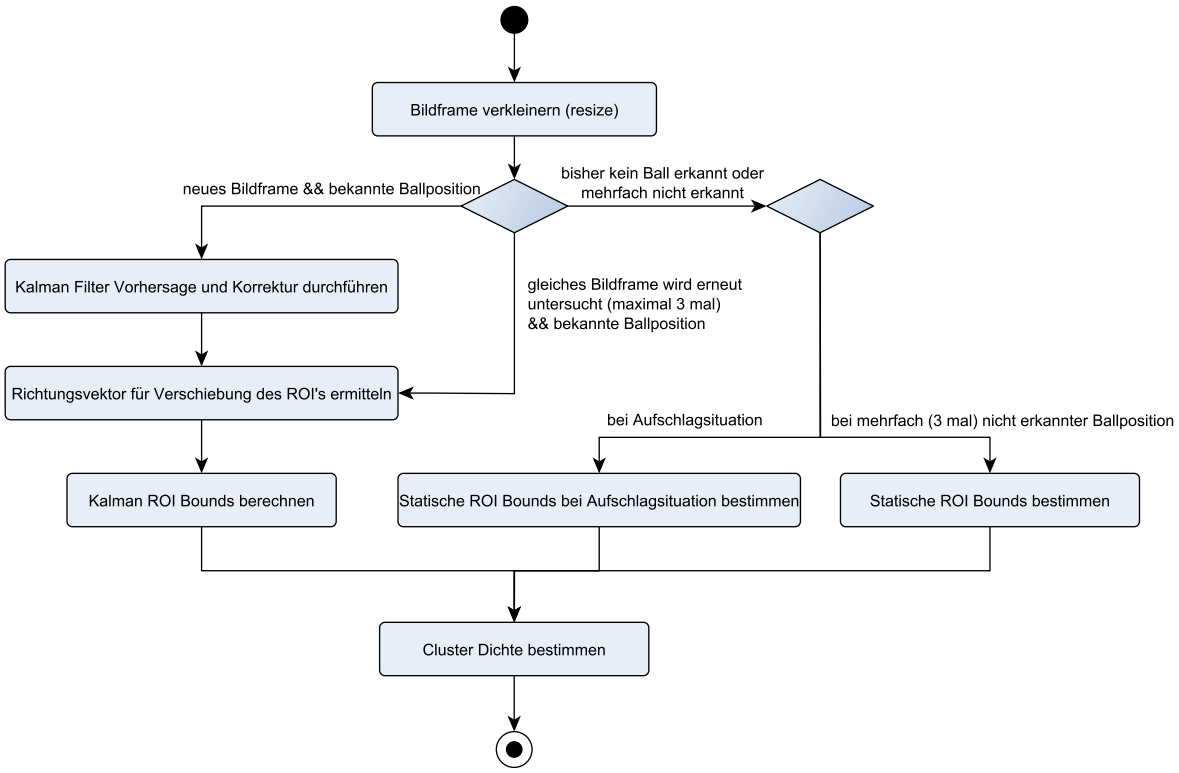


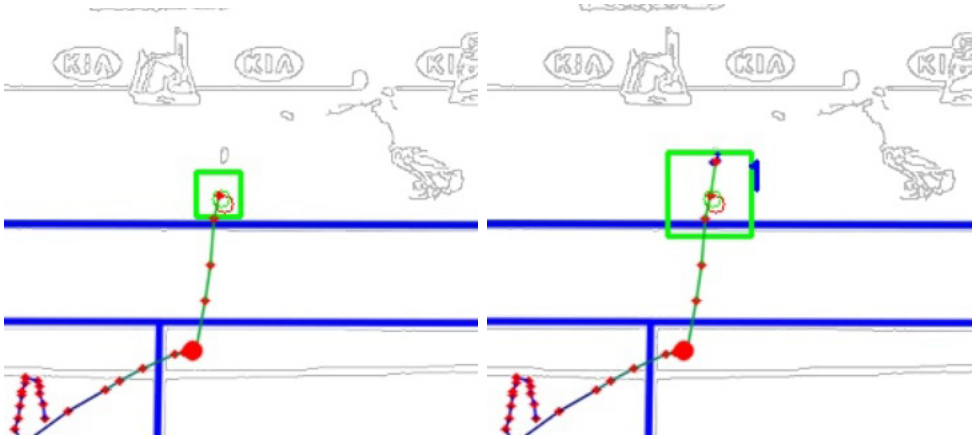
Abbildung 4.13.: Übersicht über das Vorgehen den Suchbereich festzulegen

und die Kovarianz R auf den Wert 0.1 festgelegt. Diese Werte stellten sich als sinnvoll heraus, sodass der Abstand zwischen korrigierten Messwert und der Vorhersage für die Anpassung der Größe, sowie die Verschiebung des Suchbereiches verwendet werden konnte. Die Verschiebung des Suchbereiches in die vermeintliche Richtung in der der Ball als nächstes sein wird, kann anhand des Kalman Filters, aber auch mit Hilfe der zuletzt erkannten Ballpositionen ermittelt werden. Es kann durch einen Parameter festgelegt werden, anhand welchen Kriterien die Verschiebung stattfinden soll. Bei Verschiebung mit Hilfe des Kalman Filters, wird der Suchbereich in Richtung des Vektors von Vorhersage zur korrigierten Messung verschoben. Im Falle, dass die Verschiebung durch die letzten Ballpositionen bestimmt wird, wird der Suchbereich in Richtung der zuletzt erkannten Ballposition ausgehend von der vorletzten erkannten Ballposition verschoben. Die Größe des Suchbereiches hängt wiederrum von dem Abstand zwischen korrigierten Messwert und der Vorhersage des Kalman Filters ab.

Hier gibt es je nach Parameterauswahl zwei Methoden die **Breite des rechteckigen Suchbereiches** auszuwählen.

- lineare Berechnung der Breite:

$$ROIBreite = mind.BreiteROI + linearScale \cdot KalmanBetrag \tag{4.2}$$



(a) Versuch 1: Ball liegt nicht im Suchbereich (b) Versuch 2: Ball liegt jetzt im Suchbereich

Abbildung 4.14.: Beispiel für die Vergrößerung des Suchbereiches, wenn der Ball nicht im Suchbereich liegt

- exponentielle Berechnung der Breite:

$$ROIBreite = mind.BreiteROI \cdot KalmanBetrag^{exponentiellerScale} \tag{4.3}$$

Diese Möglichkeiten die ROI Breite festzulegen, werden im Kapitel 5.1.3 Evaluierung genauer untersucht.

Je nach Größe des Suchbereiches wird die **Suchdichte** festgelegt, sodass bei kleinem Suchbereich dieser genauer untersucht werden kann als ein großer Suchbereich. Zu einem kann dies zu einer besseren Ballerkennung führen, wenn der Suchbereich sehr klein ist und der Ball sich in der Nähe des Spielers befindet, da das Cluster des Balles auch bei näherkommen mit dem Spielercluster nicht verschmilzt. Daher wird der Ball auch sehr nahe am Spieler noch als eigenständiges Cluster erkannt. Zum anderen treten weniger fehlerhafte Erkennungen des Balles auf, wenn in einem großen Suchbereich mit einer größeren Suchdichte gesucht wird als mit einer sehr kleinen, da dadurch der Spieler als Ganzes erkannt wird und nicht als mehrere Cluster, auch wenn der Ball in diesem Fall nicht mehr erkannt werden kann. Darauf wird in den nächsten Abschnitten 4.3.2 genauer eingegangen.

Bei der Umsetzung zeigte sich, dass gerade kleinere Suchbereiche Probleme verursachen können, wenn sie den Ball in der Nähe eines Spielers suchen. Dort wird neben dem Ball ebenfalls der Spieler als relativ kleines Cluster erkannt, da der Suchbereich nur einen Teil des Spielers enthält, was dazu führen kann, dass der Spieler mit dem Ball verwechselt werden kann und so der Ball verloren geht. Daher wird hierfür in Kapitel 5.1.3 eine Auswertung erstellt, welche Suchbereiche die beste Ballerkennung ermöglichen.

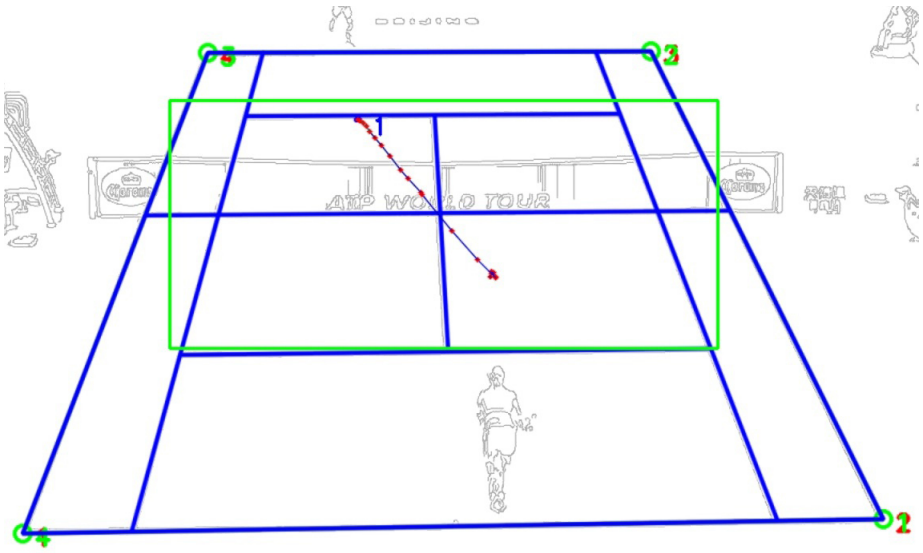


Abbildung 4.15.: Suchbereich bei verlorener Ballposition in der Spielfeldmitte (Netz)

4.3.2. Gruppieren von bewegten Bildpunkten

Wie in Kapitel 2.1 beschrieben, werden Bilder pixelweise dargestellt. Bei der Bilderkennung können daher nur Punkte (Pixel) erkannt werden, anhand dessen der Ball identifiziert werden soll. Um dies zu realisieren, war die Herangehensweise in dieser Arbeit, die bewegten Bildpunkte durch Clusterverfahren zu gruppieren und anschließend zu untersuchen.

Bewegungen von Videos können durch Differenzbilder von zwei aufeinanderfolgenden Bildframes erkannt werden. Dazu kann jeder Pixelwert des Ausgangsbildes vom nachfolgendem Bild abgezogen werden. Dadurch eliminieren sich die Pixel des Hintergrunds, die sich in diesen beiden Bildern nicht verändern. Lediglich die Änderungen der Bilder, also die bewegten Bildpunkte, werden dargestellt. Diese Bildpunkte können daraufhin gruppiert werden, sodass daraus Cluster entstehen. Diesen Clustern soll anschließend eine Bedeutung zugewiesen werden. Hierbei stellen sich vor allem die Konturen der Objekte als wichtig heraus, da dadurch bereits gute Aussagen über ein Objekt getroffen werden können. Daher werden auch hier die Konturen der bewegten Bildpunkte betrachtet, die durch die OpenCV Funktionalität *findContours*() (siehe 2.5.2.2) gewonnen werden können. Diese Konturen bestehen aus einzelnen Bildpunkten, die die Kontur aufspannen. Diese Bildpunkte können anschließend durch ein Clusterverfahren gruppiert werden. Weiterhin bestehen die Konturen aus weniger Bildpunkten, als dass das Bild insgesamt an bewegten Bildpunkten registriert hat. Dadurch wird die Anzahl der erkannten Bildpunkte reduziert, was vorteilhaft für die Laufzeit des Clusterverfahren ist.

Ein Überblick über diesen Ablauf ist in Aktivitätsdiagramm 4.16 dargestellt. Die aus den Konturen gewonnenen bewegten Bildpunkte werden gruppiert. Die daraus entstandenen Cluster können anschließend analysiert werden, ob anhand ihnen auf den Ball geschlossen werden kann. Ein Problem bei dieser Vorgehensweise ist jedoch, dass die Konturen in Bildern in einigen Situa-

tionen nicht gut zu erkennen sind. Gerade bei Bildern mit niedriger Bildqualität oder geringen Kontrastwerten, das zum Beispiel durch Sonnenlicht auf dem Spielfeld auftreten könnte, ist es schwer gute Thresholdwerte für die Canny Edge Detection und damit für die Konturenfindung festzulegen. Daher ist es nicht immer möglich Konturen zuverlässig zu finden. Infolgedessen können Spieler und Ball nicht immer gefunden werden, da bereits die Konturen unvollständig oder gar nicht erkannt werden. Ein Vorteil, die Bilder anhand der Konturen zu untersuchen, ist es, dass die Bälle und Spieler bei ausreichender Bildqualität schnell und bereits komplett erkannt werden können.

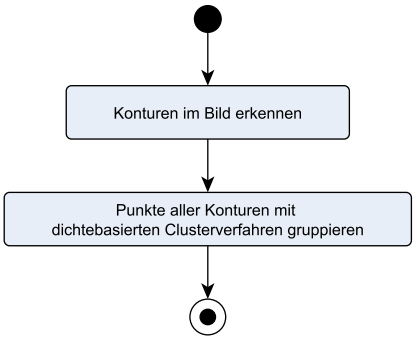


Abbildung 4.16.: Übersicht über den Ablauf, um die bewegten Bildpunkte zu gruppieren

Zum Gruppieren der Bildpunkte wurde der, in Kapitel 2.4 beschriebene, **DBSCAN Algorithmus** verwendet, der Objekte anhand der Dichte zueinander gruppiert. Bei der Umsetzung zeigte sich, dass der Parameter *minPts* mit einem Wert von vier gute Ergebnisse liefert. Die Dichte wird dynamisch je nach Größe des Suchbereiches und Güte der Kalman Filter Vorhersage durch die Distanz ϵ festgelegt. Bei kleineren Suchbereichen wird eine höhere Dichte (kleinere Distanz ϵ) genutzt, sodass die kleinen Suchbereiche feiner untersucht werden können. In größeren Suchbereichen ist die Wahrscheinlichkeit größer, dass der Spieler oder andere Objekte ebenfalls entdeckt werden, sodass eine niedrigere Dichte gewählt wird. Damit war es möglich bei guter Güte der Kalman Filter Vorhersage und damit mit einem kleinen Suchbereich den Ball auch in der Nähe des Spielers gut zu erkennen, ohne dass der Ball mit dem Spieler verschmilzt. Problematisch ist die Ballerkennung aber grundsätzlich, wenn der Ball vor dem Spieler oder sogar vom Spieler verdeckt ist. Dort kann der Ball nicht erkannt werden. Weiterhin können je nach Clusterdichte Teile des Spielers, wie Schläger oder Kopf als eigenes Cluster erkannt werden, sodass diese fälschlicherweise als Ball identifiziert werden. Daher soll in diesen Fällen die Clusterdichte niedriger gewählt werden, sodass die Spieler als Ganzes erkannt und somit nicht als Ball identifiziert werden.

Neben dem DBSCAN Algorithmus wäre der *k-means* Algorithmus eine Alternative gewesen die erkannten Bildpunkte zu gruppieren. Allerdings müsste hierfür bereits zuvor die Anzahl der Cluster k angegeben werden, was in einem solch dynamischen Anwendungsgebiet nicht ohne weiteres zu realisieren ist. Weiterhin bevorzugt *k-means* ähnliche Clustergrößen, wodurch er für diesen Anwendungszweck nicht gut geeignet ist, da die Clustergröße des Balles und der Spieler

sich deutlich unterscheiden.

Nachteil des DBSCAN Algorithmus ist zwar die Laufzeit von $O(n^2)$ für n zu clusternde Bildpunkte, dem allerdings durch ausschließliche Verwendung der Bildpunkte der Konturen entgegengewirkt werden kann. Weiterhin ist die Performanz in dieser Arbeit nicht das entscheidende Kriterium, sodass hier keine Bedeutung auf die Optimierung der Laufzeit des Clusterverfahrens gelegt wurde.

4.3.3. Suche nach der neuen Ballposition

In diesem Abschnitt werden die Ideen und Überlegungen zu der Ballsuche und dessen Umsetzung erläutert. Hierfür soll zunächst der Ablauf in Abbildung 4.17 ein Überblick schaffen, sodass auf die einzelnen Schritte nach und nach eingegangen werden kann.

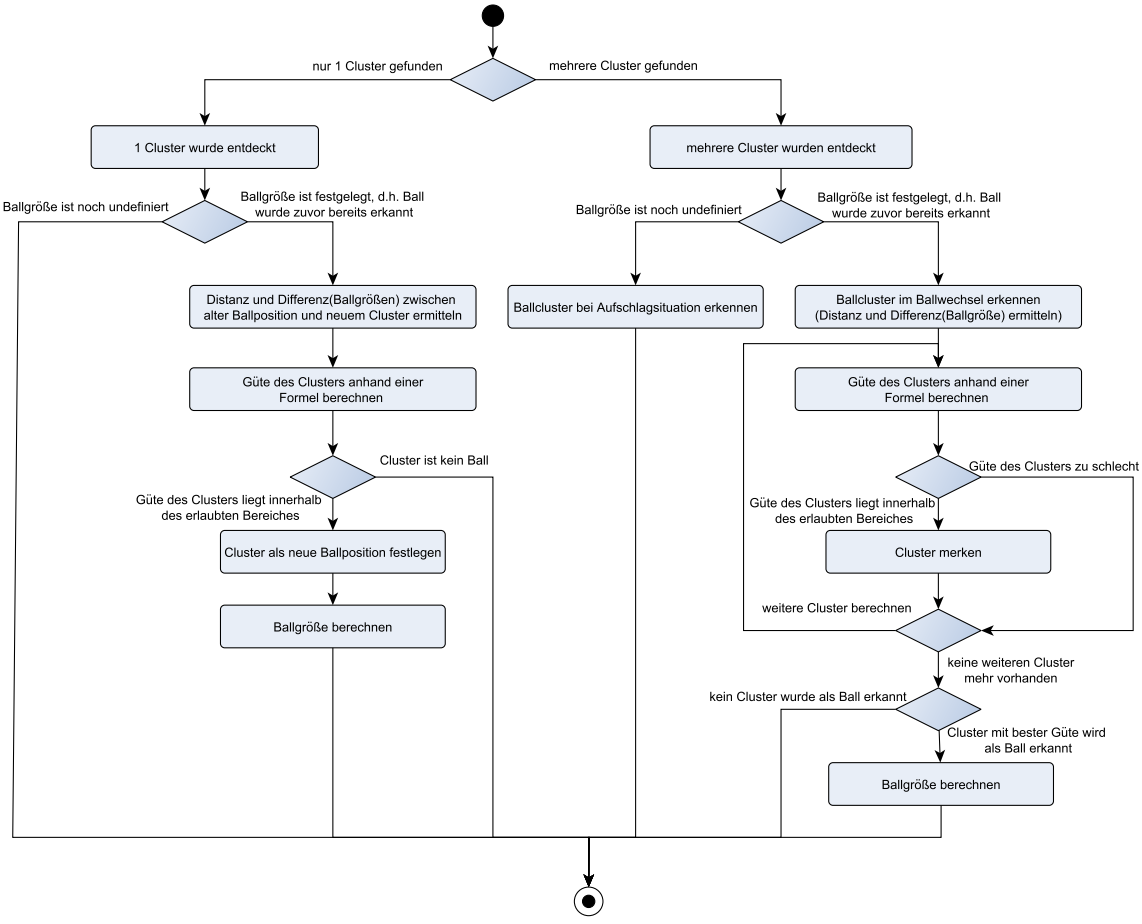


Abbildung 4.17.: Übersicht über das Vorgehen bei der Suche der Ballposition

Bei der Suche nach der neuen Ballposition werden die zuvor geclusterten Bildpunkte untersucht,

sodass ein Cluster als Ball identifiziert werden kann. Grundsätzlich werden die Clustergrößen (Anzahl an Punkten pro Cluster) zur Ballerkennung genutzt. Je nach Größe des Clusters soll auf den Ball oder die Spieler geschlossen werden. Bei bereits erkannten Ball, kann neben der Clustergröße, in Form der Größendifferenz des bereits erkannten Balles und des zu untersuchenden Clusters, auch die Distanz zwischen der bereits erkannten Ballposition und des zu untersuchenden Clusters herangezogen werden. Die Größen der Cluster unterscheiden sich je nach Bildqualität und den Objekten die betrachtet werden. Auch durch morphologische Funktionen, die vor der Kanten- und Konturerkennung auf das Bild angewandt werden, kann die Qualität der Konturen und damit die Anzahl der Bildpunkte beeinflusst werden. Unabhängig davon wird ein vollständig erkannter Spieler immer mehr Clusterpunkte besitzen als der Ball. Dadurch kann durch einfache Bereichsüberprüfungen erkannt werden, ob sich die Clustergröße in einem akzeptablen Bereich befindet, sodass das überprüfte Cluster als Ball identifiziert werden kann. Dieser Bereich wird anhand der Bildgröße und eines bestimmten prozentualen Anteiles festgelegt. Es kann allerdings leider nicht immer garantiert werden, dass es sich bei allen Cluster, die in einem akzeptablen Größenbereich liegen, um einen Ball handelt. Bei schlechter Bildqualität oder bei Einschränkung des Bildbereiches kann es vorkommen, dass der Spieler nicht als Ganzes erkannt wird. Hier kann der Spieler beziehungsweise Teile von ihm auch als kleineres Cluster dargestellt werden. Um auch in diesen Fällen den Ball zuverlässig erkennen zu können, werden im Allgemeinen zwei Situationen unterschieden:

- Aufschlag- und Annahmesituation
- im laufenden Ballwechsel

Bei der **Aufschlag- und Annahmesituation** liegt die Herausforderung darin, den Ball zu erkennen, ohne dass zuvor eine Position des Balles festgestellt wurde. Hier gibt es verschiedene Überlegungen, um in solchen Situationen den Ball möglichst zuverlässig zu erkennen.

- **Erkennung von Spielern und Ball:**
  Bei Betrachtung eines Bereiches, um das Spielfeld herum, wird davon ausgegangen zwei größere Cluster entdecken zu können. Diese größeren Cluster stellen die Spieler dar, da sich ausschließlich die Spieler und der Ball auf dem Spielfeld befinden sollten. Wenn ein kleineres drittes Cluster entdeckt wird, müsste es sich dabei um den Ball handeln. Hier können allerdings viele Störungen auftreten, wenn beispielsweise ein Linienrichter den Ball holt oder der Suchbereich zu groß ist und ein Linienrichter sich bewegt. Durch entsprechende Einschränkung des Suchbereiches können diese Störungen verringert werden. Weiterhin kann eine sicherere Ballidentifikation erreicht werden, indem davon ausgegangen wird, dass sich der Ball, also das kleinere Cluster, in der Y-Richtung des Bildes zwischen den beiden Spielern befinden muss.
- **Erkennung des aufschlagenden und annehmenden Spielers:**
  Bei diesem Ansatz wird sich die Tatsache zu nutzen gemacht, dass der Aufschläger fast ausschließlich aus der gleichen Position des Feldes aufschlägt. Er steht immer möglichst nahe an der Grundlinie, sowie in der Mitte des Feldes. Zwar muss, je nach Punktestand, von links oder rechts aufgeschlagen werden, was aber durch Erkennung des Spielfeldes fest-

gestellt werden kann. Der Annehmende dagegen steht zwar ebenfalls ungefähr auf Höhe der Grundlinie (je nach Aufschlag des Gegners entweder kurz hinter oder etwas vor der Grundlinie), dafür ehr an der Seitenauslinie, da der Aufschlag immer diagonal in den gegenüberliegenden Aufschlagbereich gespielt werden muss. Nachdem diese Aufschlag- und Annahmesituationen immer sehr ähnlich sind, kann nur anhand der Position der Spieler erkannt werden, wer aufschlägt und wer annimmt. Unter Umständen können sich die Spieler auch während des Ballwechsels auf diesen Positionen befinden, allerdings dann meistens nur sehr kurz. In der Aufschlag- und Annahmesituation verweilen die Spieler, je nach Spieler, mehrere Sekunden in diesen Bereichen des Spielfeldes. Ist solch eine Aufschlagsituation erkannt, kann der Suchbereich des Balles eingeschränkt werden, da klar ist, dass der Ball sich beim Aufschläger befindet.

Ist der Ball erstmal erkannt, gibt es Möglichkeiten den **Ball während des Ballwechsels** zu verfolgen. Hier ist die Größe des bereits erkannten Balles ausschlaggebend, aber auch die Distanz von dem zu untersuchenden Cluster zur alten Ballposition kann zur Identifizierung des Balles verwendet werden. Aus diesen beiden Werten kann eine Güte des Clusters ermittelt werden, die beschreibt, wie wahrscheinlich es sich bei dem zu untersuchten Cluster um einen Ball handelt. Grundsätzlich zeigt es sich, je weiter das Cluster von der alten Ballposition entfernt und je größer die Differenz der Clustergrößen ist, desto unwahrscheinlicher ist es, dass es sich um einen Ball handelt.

Anschließend soll etwas genauer auf die Umsetzung und dabei auftretende Probleme der Ballidentifikation eingegangen werden, da sie ein sehr wichtiger Bestandteil der Ballerkennung darstellt. Zuerst soll auf den in Abbildung 4.17 beschriebenen **Ablaufpfad für ein erkanntes Ballcluster** eingegangen werden. Anhand einer Variable, die die bisher erkannte Ballgröße speichert, wird erkannt, ob der Ball zum ersten Mal erkannt werden muss oder ob der Ball zuvor schon erkannt wurde. Ist diese Größenvariable undefiniert, wurde zuvor noch kein Ball erkannt. Bei einem der ersten Versuche den Ball zu erkennen, wurde bei einem erkannten Cluster davon ausgegangen, dass es sich auf jeden Fall um den Ball handeln muss. Allerdings wurden dadurch fälschlicherweise auch die Spieler oder andere Störungen häufiger als erste Ballposition angenommen. Durch Überprüfung der Clustergröße konnten zumindest größere Cluster, wie der komplett erkannte Spieler, ausgeschlossen werden. Je nach Spielsituation wirkten sich allerdings andere Störungen, wie Bewegungen eines Linienrichters, auf die Ballerkennung aus. Ein grundsätzliches Problem ist, dass aufgrund der Differenzbilder, der Spieler nicht immer komplett erkannt wird, da die Differenzbilder nur Bewegungen zeigen. Wenn sich also ein Spieler nicht oder nur seinen Arm bewegt, wird er nicht oder nur sein Arm erkannt. Dadurch können auch nur Teile des Spielers als Cluster erkannt werden, was die Identifizierung anhand der Clustergröße erschwert. Dies ist in Abbildung 4.18 dargestellt. Darin ist lediglich der Arm und Schläger (rotes Cluster) und der Fuß (blaues Cluser) des Aufschlägers zu erkennen, da vor dem Aufschlag der Körper noch nicht in Bewegung ist. Der annehmende Spieler dagegen erwartet in Ruhe den Aufschlag und wird daher gar nicht erkannt.

Weiterhin ist das Ziel dieser Arbeit ganze Ballwechsel zu erkennen, sodass der Ball beginnend beim Aufschlag erkannt werden soll. Daher wurden klar definierte Aufschlagsituationen einge-

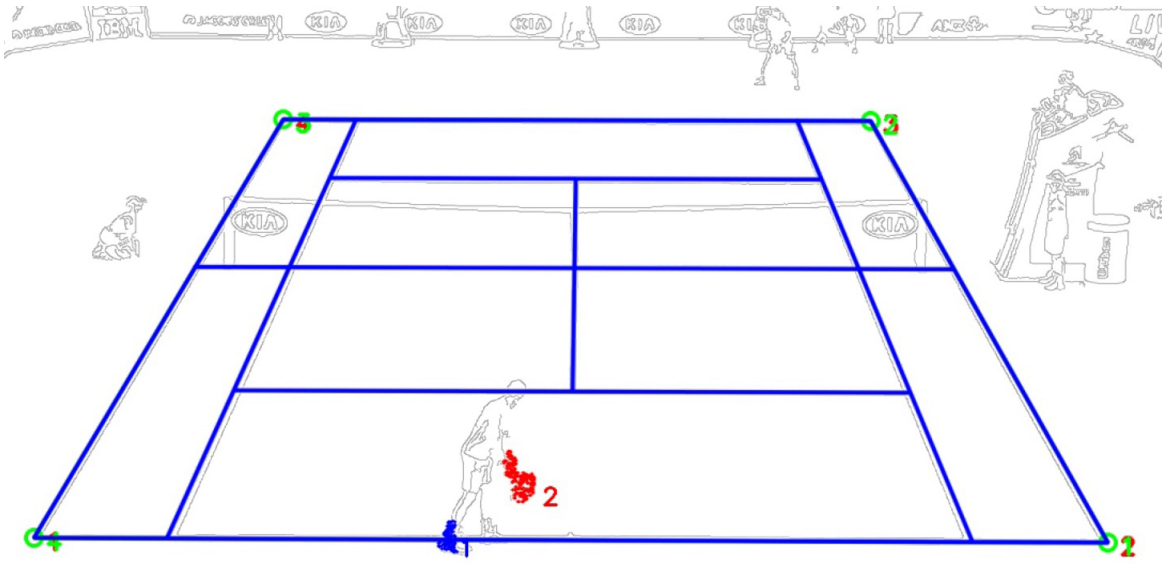


Abbildung 4.18.: Darstellung der bewegten Bildpunkte als Cluster: lediglich der Arm und Fuß des unteren Spielers bewegt sich

führt, sodass der Ball zuverlässig zu Beginn eines Ballwechsels erkannt werden kann. Um dies zu ermöglichen werden drei Cluster benötigt. Auf dieses Vorgehen wird in einer der nächsten Abschnitte genauer eingegangen.

Wenn der **Ball bereits erkannt** wurde, wird zuerst die Ballgröße und Distanz zwischen der bereits erkannten Ballposition und des zu untersuchenden Clusters ermittelt. Die **Ballgröße** spiegelt die Anzahl der Punkte, aus dem das Cluster besteht, dar. Aus dieser wird eine Größendifferenz mit der bisher erkannten Ballgröße ermittelt. Dabei wird nur der Betrag der Differenz betrachtet, d.h. es kann sich nicht um eine negative Differenz handeln.

Für die Distanzbestimmung wird zuerst der Mittelpunkt des erkannten Clusters bestimmt. Dafür wird das *k-means* Clusterverfahren verwendet, da hier der Parameter k feststeht ($k = 1$) und dadurch einfach und schnell das Zentrum des Clusters bestimmt werden kann. Die **Distanz** wird durch Verwendung des Betrages des Vektors von der bisher erkannten Ballposition a zum Mittelpunkt des erkannten Clusters b berechnet. Die Formel dazu lautet: $\left|\vec{ab}\right| = \sqrt{(a_x - b_x)^2 + (a_y - b_y^2)}$. Probleme können auftreten, wenn der Ball nicht in jedem Bildframe erkannt wird und daher große Distanzen auftreten. Hier kann es leichter passieren, dass andere Cluster, die sich näher zur alten Ballposition befinden, als neuer Ball erkannt werden. Daher wird, zur besseren Erkennung des Balles bei fehlgeschlagener Ballerkennung, ein Korrekturbetrag von der Distanz abgezogen, da davon auszugehen ist, dass der Ball sich von der zuletzt erkannten Ballposition entfernt hat. Er beschreibt also quasi die durchschnittlich vermutete Distanz des Balles zwischen zwei Bildframes. Dieser Korrekturbetrag wird mit der Anzahl der Frames, in dem der Ball nicht erkannt werden konnte, multipliziert. Dadurch kann der Ball leichter

wiedergefunden werden, nachdem er in einzelnen Bildframes nicht erkannt werden konnte.

Grundsätzlich wird anhand der Differenz und der Distanz eine Güte des zu untersuchenden Clusters ermittelt. Die **Güte des Clusters** kann durch linearer oder exponentieller Berechnungsformel bestimmt werden. Dabei kann die Distanz und Differenz gewichtet mit eingehen. Bei der linearen Formel *Clustergüte*= (*linearerDistanzParameter* · *Distanz*) · (*linearerDifferenzParameter* · *Differenz)* gehen sowohl die Distanz als auch die Differenz durch Multiplikation mit einem Parameter in die Clustergüte ein. Anhand dieser Parameter kann die Distanz und Differenz gewichtet werden. Die exponentielle Berechnungsformel *Clustergüte*= (*linearerDistanzParameter* · *Distanz*) · (*exponentiellerDifferenzParameter* · $Differenz^2$) lässt die Differenz quadratisch eingehen und sorgt so für einen steileren Anstieg der Güte, wenn die Differenz größer wird. Grundsätzlich ist die Güte des Clusters besser, wenn das Ergebnis der Berechnungsformel klein ist. Liegt die Güte des zu untersuchenden Clusters in einem akzeptierenden Bereich, so ist eine neue Ballposition gefunden. Dieser Bereich hängt stark von der Berechnungsformel, sowie dessen Parametern ab. Aufgrund der Vielzahl der Parameter, ist es sehr aufwendig geeignete Parameter und einen akzeptierenden Bereich zu finden. Daher sollen in Kapitel 5.1.5 diese Parameter untersucht werden, um eine gute Konfiguration der Parameter zu erhalten.

Zum Schluss wird die Ballgröße bestimmt, sodass sie als bisherige Ballgröße für die nächste Ballidentifikation genutzt werden kann. Zur **Bestimmung der Ballgröße** gibt es verschiedene Vorgehensweisen. Zum einen kann der Ball durch einen einfachen Vergleich der bisherigen Clustergröße des Balles mit der zu untersuchenden Clustergröße gut identifiziert werden, sofern die Clustergröße sich nicht groß verändert. Allerdings verändert sich die Clustergröße des Balles in einem gewissen Rahmen, da der Ball durch unterschiedliche Geschwindigkeiten und Positionen auf dem Spielfeld unterschiedlich gut erkannt wird. Beispielsweise wird der Ball aufgrund der perspektivischen Videoaufnahmen im unteren Bereich des Spielfeldes größer dargestellt und besteht daher aus mehr Bildpunkten. Dem kann durch Ermittlung einer durchschnittlichen Ballgröße über mehrere zuletzt erkannte Ballpositionen hinweg entgegengewirkt werden. In dieser Arbeit wurden die letzten vier Ballpositionen gewählt, aus denen der Durchschnitt für die neue Ballgröße bestimmt werden soll. Dadurch passt sich bei kleineren Ausreißern, eventuell auch durch falsche Erkennung des Balles in der Nähe des Spielers, die Ballgröße nicht sofort an. Ein weiterer Ansatz die Ballgröße noch zuverlässiger zu bestimmen ist die letzten vier Ballgrößen gewichtet eingehen zu lassen. Die zuletzt erkannte Ballgröße s_0 wird am höchsten Gewichtet und wirkt sich somit stärker auf die neue Ballgröße aus. Die am frühsten erkannte Ballgröße s_3 wirkt sich am geringsten aus. Dies zeigt folgende Formel: $\text{neueBallgröße} = \frac{1.0 \cdot s_0 + 0.5 \cdot s_1 + 0.25 \cdot s_2 + 0.125 \cdot s_3}{1.875}$. Die Auswirkungen der Berechnungsformel und der Wahl der entsprechenden Parameter soll in Kapitel 5.1.4 dargestellt werden.

Bei **Erkennung von mehreren Clustern** läuft die Erkennung des Balles auch wie eben beschrieben ab, sofern zuvor ein Ball identifiziert wurde. Es wird lediglich die Güte aller erkannten Cluster berechnet und das Cluster gewählt, das die beste Güte besitzt und im akzeptierenden Bereich liegt.

Dies war das Vorgehen für die Ballidentifikation in einem Ballwechsel. Nun soll der Ball initial erkannt werden. Dafür müssen drei Cluster erkannt werden, da dadurch eine **Aufschlagsitua-**

tion klar identifiziert werden kann. Wie zuvor beschrieben, gibt es zwei Möglichkeiten dies zu realisieren.

Die erste Möglichkeit ist die **Spieler zu erkennen und den Ball zwischen den beiden Spielern zu suchen**. Während der Umsetzung zeigte sich, dass die Ballgröße von 20 bis teilweise sogar 80 Clusterpunkten variieren kann. Daher kann angenommen werden, dass die Spielercluster deutlich größer als die des Balles sein müssen. Anhand von Erfahrungswerten wurde eine Grenze von 150 Punkten festgelegt, die Cluster besitzen müssen, damit es sich um einen Spieler handeln kann. Werden nun drei Cluster festgestellt von denen das obere und untere ein Spieler ist, so ist es sehr wahrscheinlich, dass das Cluster, das dazwischen liegt und die Grenze von 150 Clusterpunkten nicht überschreitet, ein Ball ist. Allerdings wird der Ball dadurch immer erkannt, auch während eines bereits laufenden Ballwechsels, wenn zuvor der Ball verloren wurde. Es kann also nicht sichergestellt werden, den Ball von Beginn eines Ballwechsels zu entdecken.

Aus diesem Grund wurde Möglichkeit zwei in dieser Arbeit verwendet, die die **Positionen der Spieler bei einer Aufschlagsituation** mit einbezieht. Auch hier werden drei Cluster erwartet. Es wird, wie oben beschrieben, ausgenutzt, dass sich bei Aufschlagsituationen der aufschlagende Spieler als auch der annehmende Spieler in stark eingegrenzten Bereichen des Spielfeldes befinden. Solch eine Situation ist in Abbildung 4.11 dargestellt. Der Aufschläger (hier blaues Cluster) steht normalerweise auf Höhe der Grundlinie und ist entweder etwas links oder rechts von der Mittellinie (hier links) positioniert. Der Annehmende (hier grünes Cluster) steht auf Höhe der gegenüberliegenden Grundlinie, allerdings weiter außen an der Seitenlinie. Diese Positionen werden normalerweise eingenommen, sodass sich die Spielercluster, ausgehend von den hier dargestellten blauen und grünen Punkten, mit einer gewissen Toleranz in diesen Bereichen befinden müssen, bevor mit der Ballsuche begonnen wird. Ist eine solche Aufschlagsituation erkannt, kann der Ball gefunden werden, sofern er sich über dem Spieler befindet, da Aufschläge von oben (Überkopf) serviert werden. Dieser Bereich ist in Abbildung 4.11 mit einem roten Punkt gekennzeichnet. Befindet sich ein Cluster mit einer gewissen Toleranz in der Nähe dieses Bereiches, so wird dieses Cluster als Ball angenommen. Bei Aufschlagsituationen bei denen sich der aufschlagende Spieler, wie in Abbildung 4.19 verbildlicht ist, an der oberen Grundlinie befindet, wird der Ball allerdings nicht über dem Spieler gesucht, sondern auf Höhe des Netzes im jeweils diagonal gegenüberliegenden Aufschlagbereiches, also in der Nähe des rot markierten Punktes. In dieser Abbildung ist der Ball (grünes Cluster) zwar noch über dem Spieler (rotes Cluster) zu sehen, allerdings verschmilzt der Ball bereits im darauffolgenden Bildframe des Videos mit dem Spieler, wie in Abbildung 4.20(a) zu sehen ist.

Ebenso könnte je nach Spielfeldumgebung der Ball über dem Spieler nur schwer zu erkennen sein, wenn der Ball visuell nach oben vor die sich bewegenden Zuschauer geworfen wird. In diesem Fall würden auch die Bewegungen der Zuschauer gegebenenfalls als Cluster erkannt werden, wodurch der Ball nicht eindeutig identifiziert werden kann. Weiterhin verläuft die Flugkurve des Balles vor dem Spieler, sodass der Spieler und Ball aus einem Cluster besteht und der Ball dadurch nicht vom Spieler abgegrenzt werden kann. Zusätzlich bewegt sich der Ball bei einem Aufschlag durchaus mit über $200 km/h$, was die Erfassung des Balles erschwert. Dies zeigt sich in Abbildung 4.20(b), in dem der Ball erst nach neun Bildern (ausgehend vom Bild in Abbildung

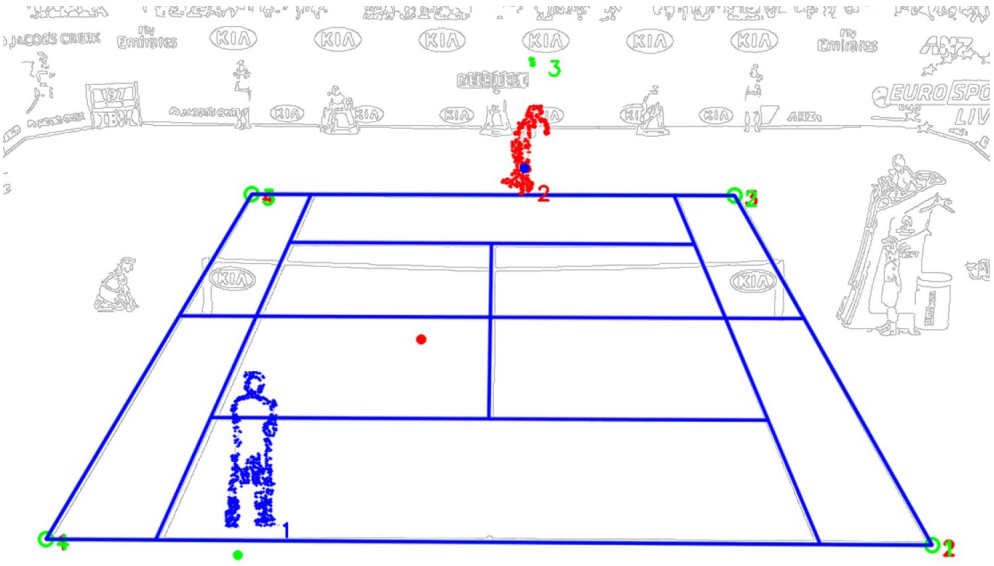


Abbildung 4.19.: Ballerkennung bei Aufschlagsituation - Aufschläger rechts oben: Ball wird vor dem Aufschlag noch erkannt

4.19) wiedererkannt wird und sich dort bereits am Netz befindet. In solchen Aufschlagsituationen kann das Ballcluster erst sicher als Ball identifiziert werden, wenn er sich in der Nähe des gegenüberliegenden Aufschlagbereiches (roter Punkt) befindet.

Zusammenfassend kann gesagt werden, dass durch Erkennen der Aufschlagsituation der Ball von Beginn des Ballwechsels, also ab des Aufschlags, identifiziert werden kann . Anschließend wird anhand der Größendifferenz und Distanz zwischen zu untersuchenden Cluster und der zuletzt erkannten Ballposition überprüft, ob es sich bei dem zu untersuchenden Cluster um einen Ball handelt. In dieser Arbeit wird der Ball durch das eben beschriebene Vorgehen identifiziert und nachverfolgt. Probleme können dadurch allerdings vor allem beim Näherkommen des Balles zum Spieler entstehen, da hier der Ball mit dem Spieler verschmilzt und daher für einige Zeit nicht zu finden ist. Auch können in diesen Situationen fälschlicherweise Teile des Spielers eine ähnliche Clustergröße als die bisherige Ballgröße besitzen, sodass diese Cluster ebenfalls als Ball erkannt werden können. Hier wird durch entsprechende Anpassung der Clusterdichte und der Parameter für die Ballerkennung versucht, falsche Identifikationen zu vermeiden. Nach verlorengegangenem Ball, wird der Suchbereich an das Netz gelegt, da der Ball nach jedem Schlag über das Netz fliegen muss, sodass hier die Wahrscheinlichkeit hoch ist, den Ball wieder zu finden.

4.4. Speicherung der Flugkurve des Balles

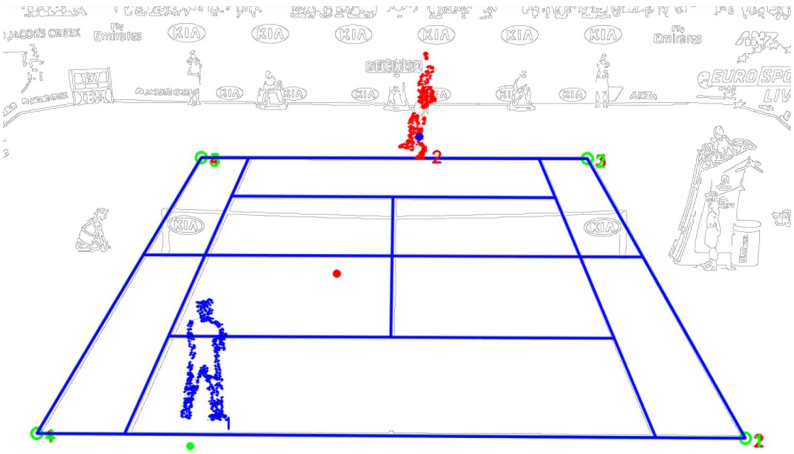
Nach erfolgreicher Ballerkennung bei Fernsehaufnahmen von Tennisspielen sollen die erkannten Ballwechsel für die von Herrn Prof. Dr. Seipel entwickelte Prologdatenbank nutzbar gemacht

werden. In dieser ist es möglich XML-Dateien mit einer bestimmten Hierarchie einzulesen. Daher müssen die Ballwechsel im Anschluss der Ballerkennung in einer XML-Datei abgespeichert werden. Listing 4.1 zeigt diese Struktur und einen kurzen Ausschnitt eines Ballwechsels.

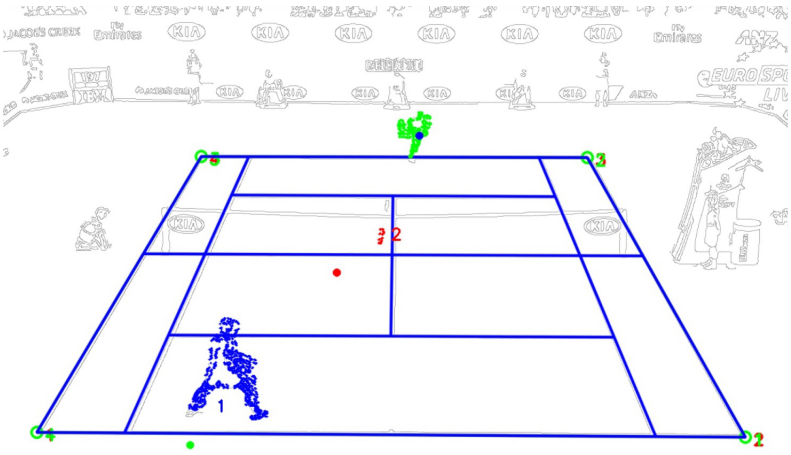
Listing 4.1: Speicherung der Flugkurve des Balles in einer XML-Datei

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<match>
  <player id="A" name="?"/> <!-- Spielername muss manuell eingefuegt werden -->
  <player id="B" name="?"/>
  <result>
    <score set="1" player_A="?" player_B="?"/> <!-- Ergebnis muss manuell eingefuegt werden -->
    <score set="2" player_A="?" player_B="?"/>
    <score set="3" player_A="?" player_B="?"/>
    <score set="4" player_A="?" player_B="?"/>
    <score set="5" player_A="?" player_B="?"/>
  </result>
  <match_facts> <!-- Daten zum Spiel muessen manuell eingetragen werden -->
    <tournament>?</tournament>
    <surface>?</surface>
    <where>?</where>
    <when>?</when>
    <round>?</round>
  </match_facts>
  <court xLT="599" yLT="290" xRT="1330" yRT="292" xLB="288" yLB="810" xRB="1627" yRB="820"
      xTopLT="599" yTopLT="290" xTopRT="1696" yTopRT="290" xTopLB="599" yTopLB="884"
      xTopRB="1696" yTopRB="884">
    <hit id="1" hand="forehand" type="ground" time="0.04" xPixel="876.349" yPixel="435.429"
        xTPixel="1044" yTPixel="534" xTMeter="-0.9" yTMeter="2.1" />
    <hit id="2" hand="forehand" type="ground" time="0.08" xPixel="874.222" yPixel="424.704"
        xTPixel="1040" yTPixel="519" xTMeter="-1" yTMeter="2.7" />
    <hit id="3" hand="forehand" type="ground" time="0.12" xPixel="872.667" yPixel="417.725"
        xTPixel="1037" yTPixel="509" xTMeter="-1" yTMeter="3.1" />
    <hit id="4" hand="forehand" type="ground" time="0.16" xPixel="871.889" yPixel="412.056"
        xTPixel="1035" yTPixel="501" xTMeter="-1" yTMeter="3.4" />
    <hit id="5" hand="forehand" type="ground" time="0.2" xPixel="869.854" yPixel="406.309"
        xTPixel="1031" yTPixel="492" xTMeter="-1.1" yTMeter="3.8" />
    <hit id="6" hand="forehand" type="ground" time="0.24" xPixel="0" yPixel="0" xTPixel="0"
        yTPixel="0" xTMeter="0" yTMeter="0" />
    <hit id="7" hand="forehand" type="ground" time="0.28" xPixel="0" yPixel="0" xTPixel="0"
        yTPixel="0" xTMeter="0" yTMeter="0" />
    <hit id="8" hand="forehand" type="ground" time="0.32" xPixel="860.667" yPixel="406" xTPixel="1019"
        yTPixel="492" xTMeter="-1.2" yTMeter="3.8" />
    <hit id="9" hand="forehand" type="ground" time="0.36" xPixel="861" yPixel="409.111" xTPixel="1020"
        yTPixel="497" xTMeter="-1.2" yTMeter="3.6" />
    <hit id="10" hand="forehand" type="ground" time="0.4" xPixel="859.137" yPixel="415.922"
        xTPixel="1019" yTPixel="507" xTMeter="-1.2" yTMeter="3.2" />
  </court>
</match>
```

Im XML werden zuerst einige Informationen zum Spiel, wie Spielernamen und Ergebnis, angegeben, die allerdings manuell eingetragen werden müssen. Der Ballwechsel wird innerhalb eines *<court>*-Tags erfasst, das angibt bei welchen Pixeln des Bildes sich das Spielfeld befindet. Hier werden Attribute wie *xLT* für die X-Koordinate der linken oberen Spielfeldecke angegeben. Auch werden die Koordinaten zum einen als perspektivische Koordinaten als auch in Koordinaten, die in Draufsicht dargestellt sind, abgespeichert. Dies ist für eine weitere Umrechnung der Koordinaten des Bildes in die Koordinaten, die in der Prologdatenbank für weitere Taktikanalysen benötigten werden, notwendig. Dies wird ebenso bei der Beschreibung der Ballposition gemacht, bei der in jedem Frame (Attribute *id*) die Position durch X und Y-Koordinaten angegeben ist. In dem Tag *<hit>* gibt es Attribute die die Koordinaten des Balles sowohl perspektivisch (Attribute *xPixel* und *yPixel*) als auch in Draufsicht (Attribute *xTPixel* und *yTPixel*) darstellen. Die Attribute *xTMeter* und *yTMeter* geben die Ballkoordinaten im Bezug zum Spielfeld in Metern an. Dabei besitzt der Mittelpunkt des Spielfeldes (am Netz und bei der Linie, die die Aufschlagfelder unterteilt) die Koordinaten (0,0). Die Attribute *hand* und *type* werden später von der Taktikanalyse bestimmt. All diese Angaben der Koordinaten des Balles sollen dazu beitragen, die für die Taktikanalyse notwendigen Ballkoordinaten, bestimmen zu können. Problematisch ist hier vor allem die fehlende Höhenangabe des Balles in 2D Fernsehaufnahmen. Diese muss durch weitere Logik herausgerechnet werden, was durch Logikverknüpfungen in der Prologdatenbank realisiert werden kann.



(a) Ball verschmilzt mit Spieler (1 Bild später)



(b) Ball wird wieder am Netz erkannt (9 Bilder später)

Abbildung 4.20.: Ballerkennung bei Aufschlagsituation - Aufschläger rechts oben

5. Evaluierung

In diesem Kapitel soll die Ballerkennung im Tennis untersucht werden. Zuerst sollen, anhand der nachfolgend vorgestellten Parameter, verschiedene Möglichkeiten den Ball zu entdecken und nachzuverfolgen gegenübergestellt werden. Dabei soll eine effektive Konfiguration dieser Parameter gefunden werden, damit das Spielfeld und der Ball möglichst zuverlässig erkannt wird. Ebenso sollen die unterschiedlichen Parameter interpretiert werden, sodass deren Auswirkungen ersichtlich werden. Weiterhin wird in den Hauptszenarien aufgezeigt, unter welchen Umständen der Ball identifiziert werden kann und in welchen Situationen nicht. Hierbei sollen die Spielfeldbeläge, die Kameraführung und die Bildqualität bei ungeschnittenen Fernsehaufnahmen betrachtet werden.

Diese **Hauptszenarien** sollen genauer betrachtet und gegenübergestellt werden. Begonnen wird mit der Untersuchung der unterschiedlichen **Bodenbeläge** und ihrer Auswirkung auf die Ballerkennung. Anzunehmen ist, dass sich ein guter Kontrast von Ball zu Spielfelduntergrund die Ballerkennung positiv beeinflusst. Aus diesem Grund ist zu erwarten, dass ein gelber Ball bei einem blauen Spielfelduntergrund gut identifiziert werden kann. Dies soll mit der Ballerkennung bei Rasen- und Sandplätzen verglichen werden.

Ein weiterer Schritt soll die Effektivität der Ballerkennung bei unterschiedlicher **Kameraführung** bestimmen. In Bezug auf die Kameraführung sind Videoaufnahmen zu unterscheiden, bei denen sich die Kamera bewegt, sowie näher an das Spielgeschehen heran oder weiter vom Spielgeschehen weg zoomt (dynamisch) und solche bei denen die Aufnahmen aus einer festen Kameraposition und ohne Fokussierung (statisch) aufgenommen werden. Zudem sollen Kamerawechsel, zum Beispiel bei Nahaufnahmen oder Werbepausen, separat betrachtet werden, sodass die Ballanalyse davon nicht gestört wird. Ebenso könnte die Bildqualität großen Einfluss auf eine erfolgreiche Ballerkennung haben. Dies soll ausgewertet werden, indem dasselbe Video mit unterschiedlicher **Videoqualität** untersucht wird. Zuletzt soll eine umfangreiche Analyse stattfinden, die grundsätzlich die Effektivität, der in dieser Arbeit genutzten Ballerkennung, aufzeigt. Dazu sollen 94 Ballwechsel untersucht und die Erfolgsquote, wie viele **Ballwechsel** davon erfolgreich erkannt werden können, dargestellt werden.

Eine automatische Aussage über die Effektivität der Spielfeld- oder Ballerkennung ist in der Bildererkennung grundsätzlich nicht möglich. Die erkannten Ballpositionen können nicht ohne weiteres als richtig angenommen werden, da sie auch falsch klassifiziert sein könnten. Die Spieler oder Linienrichter können beispielsweise in bestimmten Situationen fälschlicherweise als Ball erkannt werden und daher das Ergebnis der Ballerkennung verzerren. Die untersuchten Szenarien können nur manuell auf ihre Richtigkeit überprüft werden. Dazu werden Videos mit den erkannten Ballpositionen erstellt und können anschließend gesichtet werden. Dies ist notwendig,

um zu validieren, dass die als Ball identifizierten Objekte auch wirklich den Ball darstellen. Bei der Evaluierung der Parameter werden sehr viele unterschiedliche Konfigurationen getestet, wodurch eine große Anzahl an Videos erzeugt wird. All diese müssten auf ihre Korrektheit untersucht werden. Es werden allerdings nur die Kandidaten genauer betrachtet und beurteilt, die vielversprechende Ergebnisse bei der automatischen Auswertung zeigten.

Bei der Untersuchung der beschriebenen Hauptszenarien sollen zudem die Ergebnisse anhand zweier Bezugsgrößen, Precision und Recall, dargestellt werden. Diese helfen Systeme zur Bilderkennung oder bei Information Retrival Szenarien zu bewerten. Die Precision gibt dabei an, mit welcher Genauigkeit die Ballerkennung funktioniert, also wie oft der Ball richtig identifiziert wurde. Der Recall dagegen beschreibt, ob die Positionen, der als Ball klassifizierten Objekte, vollständig erkannt wurden. Er gibt folglich das Verhältnis von richtig erkannten Ballpositionen zur Anzahl der Ballpositionen an, die richtig identifiziert hätten werden können. Eine manuelle Analyse, des für jede einzelne Parameterkonfiguration erstellten Videos, ist sehr aufwendig, da jeder Bildframe kritisch beleuchtet werden muss, weswegen dies, aufgrund der hohen Anzahl an verschiedenen Konfigurationen, für die Analyse der Parameter nicht möglich ist.

Die Ergebnisse werden je nach Fall in Prozent oder durch die Anzahl der erkannten Ballpositionen angegeben. Dabei soll als weiteres Gütekriterium die Anzahl der Versuche, die nötig sind, um den Ball zu finden, angegeben werden. Die Maximalanzahl der Versuche den Ball zu finden wurde auf drei festgelegt. Wird der Ball häufiger als dreimal im gleichen Bildframe nicht entdeckt, zählt der Ball in diesem Frame als nicht erkannt und der nächste Frame wird betrachtet.

In Fällen bei denen die Erfolgsquote der Ballerkennung in Prozent angegeben wird, sollen ausschließlich die Frames mit einberechnet werden, in denen es potentiell möglich ist das Spielfeld zu erkennen. Werden zwischen den Ballwechseln Nahaufnahmen oder Werbung eingeblendet, so sollen diese Frames nicht in die Ballerkennungsquote mit einfließen. Das ist darin begründet, dass der Ball bei einer Werbepause nicht gefunden werden kann. Allerdings ist es auch hierbei nur durch manuelle Überprüfung möglich, alle Frames, auf denen wirklich ein Spielfeld abgebildet ist, zu erkennen. Dennoch kommt es häufiger vor, dass bei Nahaufnahmen der Spieler oder bei Werbepausen fälschlicherweise Spielfelder angenommen werden. Aus diesem Grund fällt das Ergebnis der prozentualen Ballerkennung schlechter, als die tatsächliche Erfolgsquote, aus.

5.1. Optimierung der Parameter

Zuerst sollen verschiedene Konfigurationen von Parametern betrachtet werden, sodass in dieser Arbeit die Ballerkennung der Hauptszenarien mit einer optimierten Parameterkonfiguration untersucht werden kann. Für eine möglichst gute Kalibrierung der Parameter sollen verschiedene Videos betrachtet werden, da die Ballerkennung, je nach Videoqualität und Spielsituationen, unterschiedlich ausfallen kann. Aufgrund dessen wurden drei Videos betrachtet, die unterschiedliche Ballwechsel analysieren. Ein Video stellt einen kurzen Ballwechsel mit insgesamt 109 Bildframes dar, bei dem die Flugkurve des Balles sehr günstig erscheint, da sich der Ball visuell nicht direkt vor einem Spieler bewegt und damit nicht mit dem Spieler verschmilzt. Ein weiteres

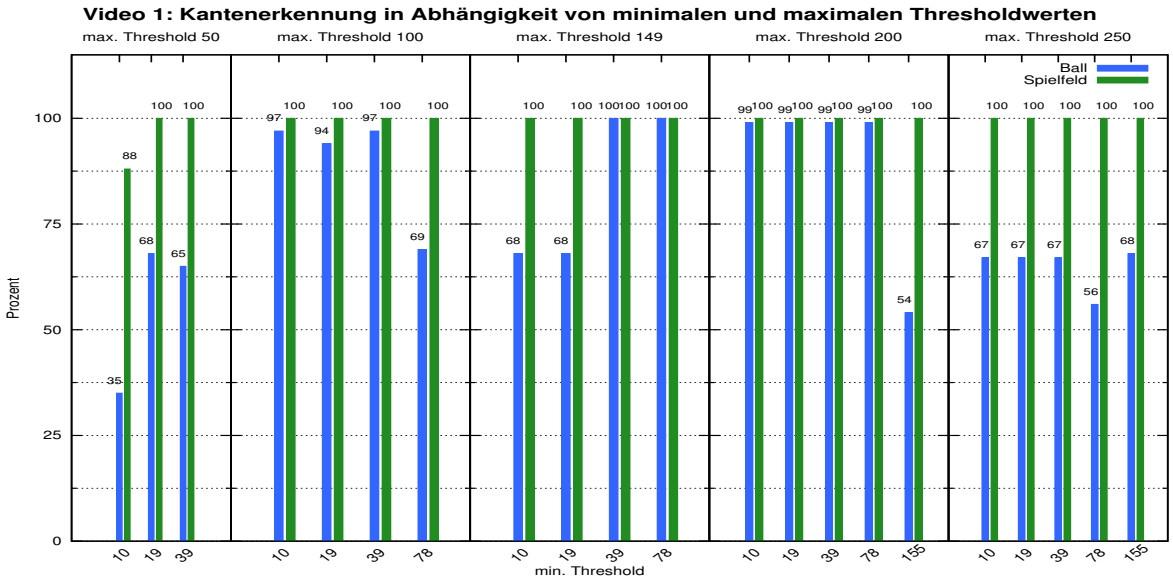


Abbildung 5.1.: Kantenerkennung in Abhängigkeit von min. und max. Thresholdwerten - Video 1

Video mit guter Videoqualität zeigt einen längeren Ballwechsel mit 637 Bildframes, bei dem die Flugkurve des Balles häufiger vor dem Spieler verläuft, sodass der Ball nicht immer erkannt werden kann. Das dritte Video dagegen enthält einen Ballwechsel von 8 Sekunden mit schlechterer Videoqualität und besteht aus insgesamt 245 Frames.

5.1.1. Analyse des Canny Edge Detectors: min. und max. Thresholdwerte

Der Canny Edge Algorithmus ermöglicht die Kantenerkennung von Bildern. Die Kantenverläufe werden genutzt, um das Spielfeld und den Ball zu erkennen. Daher beeinflusst die Effektivität des Canny Edge Detectors wesentlich die Ball- und Spielfelderkennung in dieser Masterarbeit. Daher soll als erster Schritt eine gute Konfiguration des Canny Edge Detctors gefunden werden. Dieser kann, wie in Kapitel 2.5.2.2 beschrieben, durch die beiden Parameter *minimaler* und *maximaler Threshold* angepasst werden. In den nachfolgenden Diagrammen werden die erkannten Ballpositionen und Spielfelder in Prozent und in Abhängigkeit von den eben beschriebenen Paramtern dargestellt. Wie bereits im vorherigen Absatz erwähnt, zeigte sich, dass nicht immer davon ausgegangen werden kann, dass die Fälle mit der Besten prozentualen Ball- und Spielfelderkennung auch das Beste Ergebnis liefert, da falsch klassifizierte Ballpositionen oder Spielfelder nicht automatisch festgestellt werden können. Die nachfolgenden Abbildungen werden daher vor allem genutzt, um vielversprechende Parameterkonfigurationen genauer hinterfragen zu können, sodass diese, durch manuelle Betrachtung der aufgezeichneten Videos, evaluiert werden können.

Durch Anpassung des Canny Edge Detectors kann sowohl die Ball- als auch Spielfelderkennung beeinflusst werden. Zuerst soll die Spielfelderkennung näher betrachtet werden. In Diagramm 5.1

zeigt sich, dass sich die Spielfelderkennung in Video eins, hier durch grüne Balken dargestellt, fast ausschließlich bei 100% befindet. Das Spielfeld kann folglich in fast jedem Bildframe identifiziert werden. Lediglich bei sehr kleinen Thresholdwerten von 10 und 50 für die minimalen und maximalen Thresholdwerte konnten nicht alle Spielfelder erkannt werden. Hier wurden lediglich 86% der Spielfelder erkannt. Ein ähnliches Bild zeigt sich für Video drei (siehe Diagramm 5.3, bei dem ebenfalls in den meisten Fällen 100% der Spielfelder erkannt werden konnte. Bei niedrigeren Thresholdwerten zeigte sich, dass gegenüber Video eins noch weniger Spielfelder erkannt werden konnten. Auch in Video zwei (siehe Diagramm 5.2) stellt sich diese Tendenz dar. Hier ist zu erkennen, dass grundsätzlich der Prozentwert der Spielfelderkennung bei größerem maximalen Thresholdwert höher als bei einem kleineren Thresholdwert ist. Dies lässt sich dadurch erklären, dass je höher der maximale Thresholdwert ist, desto zuverlässiger werden nur die wichtigen Kanten und damit auch die essentiellen Spielfeldkonturen erkannt. Dadurch können Bildframes bei dem kein Spielfeld zu sehen ist, besser herausgefiltert werden. In diesen Fällen können daher insgesamt weniger Spielfelder detektiert werden, wodurch das prozentuale Ergebnis der Ballerkennung höher ist. Das Ergebnis bei höheren maximalen Thresholdwerten kommt den echten Ergebnis der Spielfelderkennung näher als bei niedrigeren Werten. Bei einem maximalen Threshold von 250 bewegt sich die Spielfelderkennung zwischen 86% und 92% und fällt auf ungefähr 2% bis 80% bei maximalen Threshold von 50. Auch variiert die Spielfelderkennung bei kleineren maximalen Thresholdwerten stärker in Abhängigkeit von den minimalen Thresholdwerten. Dort ist bei größeren minimalen Thresholdwerten eine bessere Ballerkennung festzustellen als bei niedrigeren Werten. Weiterhin zeigt sich bei manueller Überprüfung der Ergebnisse, dass ein großer Abstand zwischen minimalen und maximalen Thresholdwerten dazu beiträgt, dass mehr Spielfelder erkannt werden, obwohl kein Spielfeld in den Frames dargestellt ist. In diesen Situationen werden Konturen in Frames von Nahaufnahmen erkannt, die als Spielfeld angenommen werden könnten. Auch aus diesem Grund sind die Werte der Spielfelderkennung bei kleineren minimalen Thresholdwerte geringer als bei höheren Thresholdwerten.

Bei der Ballerkennung zeigt sich in Video eins, dass die Werte, hier als blaue Balken dargestellt, bei hohem maximalen Thresholdwerten von 250 grundsätzlich bei ungefähr 67% liegen und damit kleiner als bei tieferen maximalen Thresholdwerten sind. Dies lässt sich darauf zurückzuführen, da die Kantenstärke des Balles, besonders bei schnelleren Bewegungen geringer wird, sodass diese bei einem sehr hohen maximalen Thresholdwert von 250 nicht mehr als Kante erkannt werden. Bei Video zwei und drei dagegen zeigten sich dem widersprechend gute Prozentwerte von durchschnittlich circa 50% gegenüber den Werten bei niedrigerem maximalen Threshold. Bei diesen konnten weniger als 50% der Bälle aller Bildframes identifiziert werden. Dies lässt sich durch die kleinere Anzahl an möglichen Spielfeldern erklären, als bei den Fällen mit niedrigerem Thresholdwerten, was trotz weniger erkannten Ballpositionen zu einem höheren prozentualen Ergebnis führt.

Im Allgemeinen zeigt sich in den Diagrammen, dass die Ergebnisse der einzelnen Parameterkonfigurationen sehr unterschiedlich ausfallen. Besonders in Video eins und drei lässt sich kaum eine Schlussfolgerung auf die Ballerkennung anhand der hier dargestellten Werte ziehen. Grundsätzlich zeichnet sich Video eins mit hohen Werten für die Ball- als auch Spielfelderkennung aus. Dies konnte bereits vorher vermutet werden, da der Verlauf der Flugkurve des Balles einer

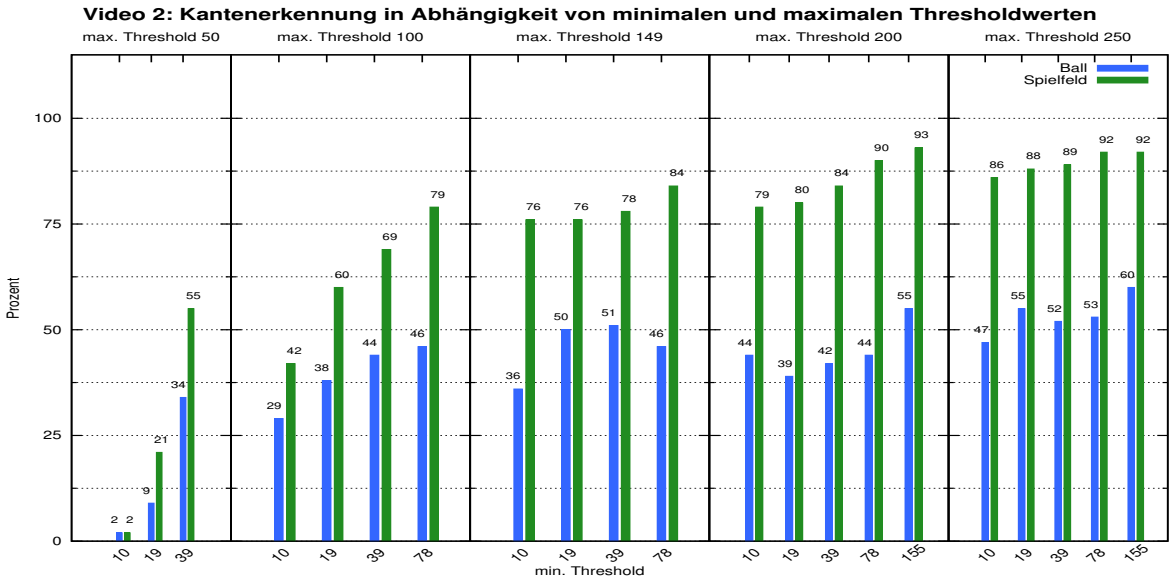


Abbildung 5.2.: Kantenerkennung in Abhängigkeit von min. und max. Thresholdwerten - Video 2

Ballerkennung in dieser Arbeit entgegenkommt, da die Flugkurve des Balles den Spieler nicht berührt oder sogar mit ihm verschmilzt. Auch aus diesem Grund variieren die Werte bei Video eins nur wenig.

Anschließend sollen Fälle genauer betrachtet werden, dessen Werte für die Ball- als auch Spielfelderkennung in allen drei Videos sehr vielversprechend sind. Aufgrund des positiven Spielverlaufs in Video eins wurden lediglich die Fälle weiter betrachtet, die in Diagramm 5.1 bei annähernd 100% für die Ballerkennung liegen. Folglich ergibt sich daraus, dass der maximale Threshold von 250 nicht mit einbezogen wird, da hier die Ballerkennung bereits in Video eins bei ungefähr 67% liegt. Dies ist, wie oben erwähnt, darauf zurückzuführen, dass durch den hohen Wert für den maximalen Threshold lediglich sehr klare Kanten detektiert werden, die aber aufgrund von sehr schnellen Bewegungen des Balles nicht immer vorzufinden sind. Es sollen auch die Fälle mit maximalen Thresholdwert von 50 und 100 nicht weiter betrachtet werden, da durch den niedrigen Threshold unwichtige Kanten mit in die Ballerkennung einbezogen werden, wodurch viele Spielfelder in den Bildframes angenommen werden, obwohl kein Spielfeld zu erkennen ist. In diesen Fällen zeigte sich ebenso, dass der Ball sehr leicht mit dem Spieler verwechselt werden kann, da der Spieler, auf Grund vieler Kanten, sehr detailliert dargestellt wird. Daher werden besonders die Konfigurationen mit maximalen Thresholdwert von 149 und 200 genauer betrachtet.

Bei maximalen Thresholdwert von 149 zeigte sich die Erkennung des Balles bei minimalen Thresholdwert von 39 sehr gut. Dort liegt der Wert in Video eins bei 100%. Auch in Video zwei zeigt sich in Diagramm 5.2 ein Prozentwert von 51 und in Video drei ein Wert von 60%. Bei genauerer Betrachtung der Videos zeigte sich aber, dass der Ball häufig mit dem Spieler verwechselt wurde und daher die Werte stark verzerrt dargestellt wurden. Dies könnte aus einem relativ

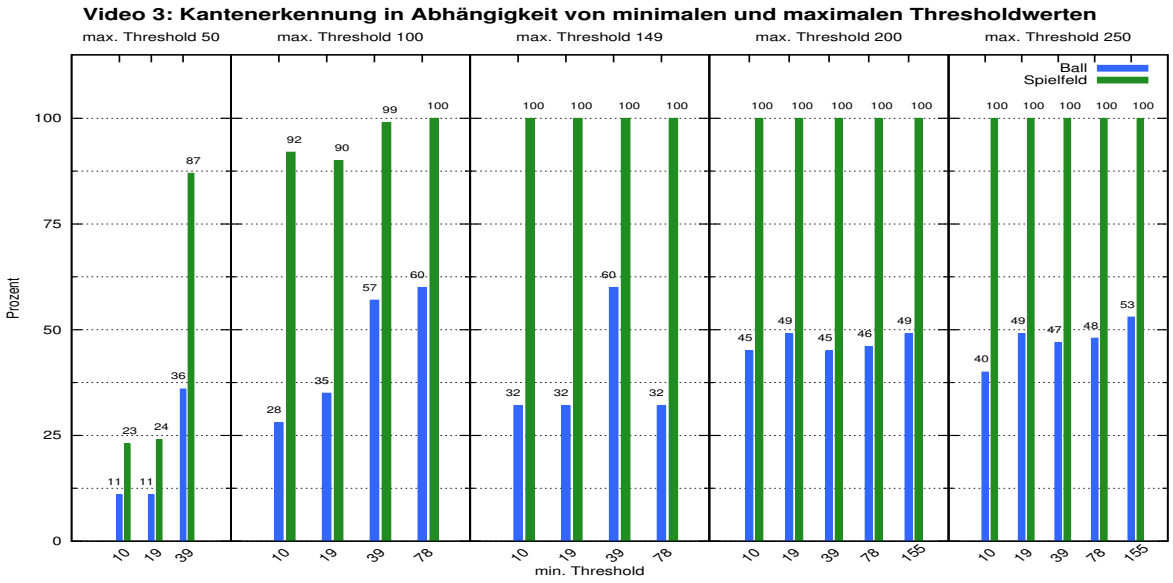


Abbildung 5.3.: Kantenerkennung in Abhängigkeit von min. und max. Thresholdwerten - Video 3

niedrigen maximalen Thresholdwert von 149 resultieren, da hier verhältnismäßig viele Kanten erkannt werden, sodass der Spieler aus vielen Konturen besteht, die fälschlicherweise mit dem Ball verwechselt werden können. Dies zeigt sich in Abbildung 5.4, in der Teile des Spielers als Ball angenommen wurden, sodass der Ball weiterhin bei dem oberen Spieler gesucht wird, obwohl der Ball bereits wieder bei dem unteren Spieler angelangt ist. Das gleiche Verhalten zeigt sich auch bei der Konfiguration mit minimalem Threshold von 78, bei dem aber auch die Werte für Video drei (siehe Diagramm 5.3) mit nur 32% stark abfallen.

Bei genauerer Betrachtung der Fälle mit einem maximalen Threshold von 200 zeigte sich für die Parameter mit 10, 19, 39 und 78 minimalen Threshold nur geringe Unterschiede in der Ballerkennung. Die Ballerkennung betrug hier ungefähr für Video eins bei 99%, für Video zwei bei 43% und bei Video drei bei 47%. Gegenüber dem maximalen Threshold von 149 waren die prozentualen Werte für die Ballerkennung allerdings niedriger, was aber darauf zurückzuführen ist, dass durch den höheren maximalen Threshold weniger unwichtige Kanten entdeckt werden, sodass der Ball seltener mit dem Spieler verwechselt wird. Bei manueller Betrachtung zeigte sich dies ebenfalls, allerdings traten weiterhin Verwechslungen zwischen Ball und Spieler auf. Insbesondere in Video drei war zu erkennen, dass sich der Ball bei niedrigem maximalen Threshold von 149 fälschlicherweise beim Spieler verfing und so die Ergebnisse verfälschte. Bei maximalen Threshold von 200 dagegen konnte der Ball weitestgehend nachverfolgt werden. Die Konfiguration mit minimalen Thresholdwert von 39 stach dabei heraus, da dort keine große Verwechslung auftrat.

Daher soll die Konfiguration von 39 für den minimalen und 200 für den maximalen Threshold in der weiteren Arbeit verwendet werden, da hier vor allem in Video eins und drei gute Ergebnisse

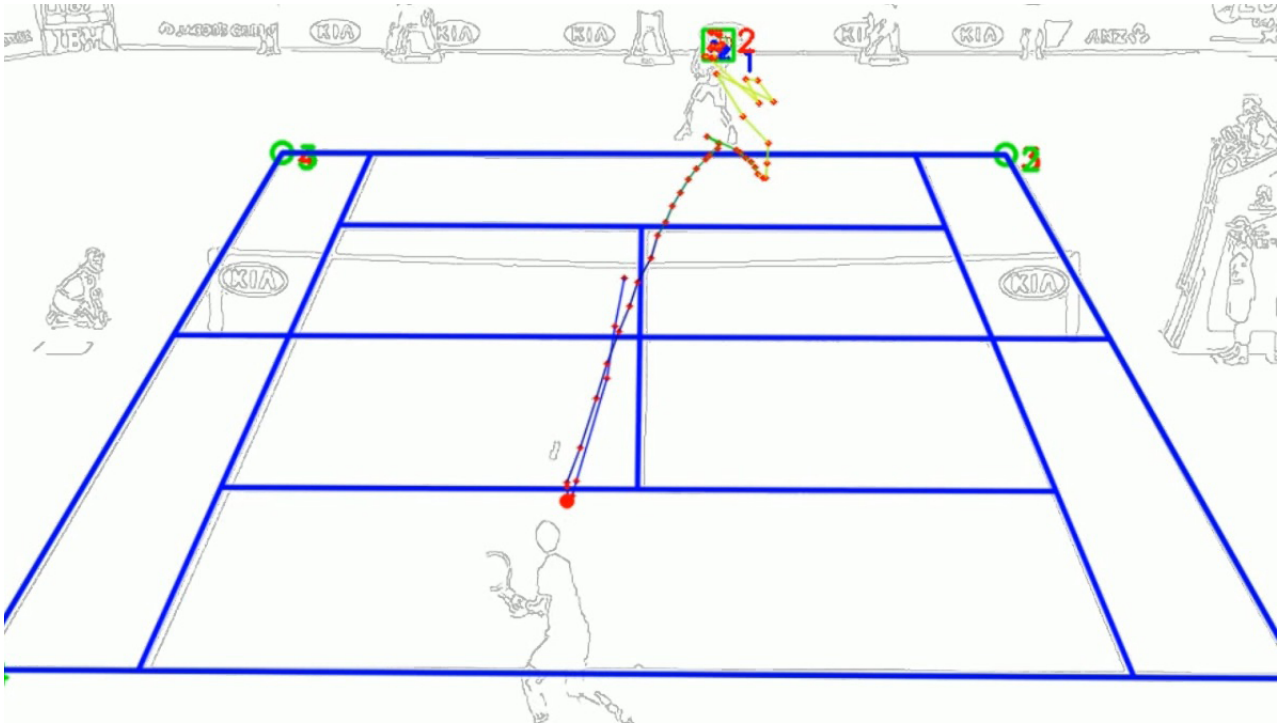


Abbildung 5.4.: Darstellung einer Verwechselung des Balles mit einem Spieler

erzielt wurden und die Ballerkennung in Video zwei im Vergleich zu den anderen Fällen sich nicht wesentlich unterschieden hat. Die relativ Breite Wahl des Bereiches von minimalen zu maximalen Threshold deutet darauf hin, dass die wichtigsten Konturen relativ genau betrachtet werden und nur ganz feine Konturen unter einem Wert von 39 ignoriert werden. Daher werden vielen Kanten in die Ballerkennung mit einbezogen, was natürlich die Ballerkennung genauer machen kann, aber auch nicht zu viele, um Störungen durch Verwechslungen zu vermeiden.

5.1.2. Analyse der Spielfelderkennung

Die Spielfelderkennung ist ausschlaggebend für die Erkennung des Balles, da nur auf Frames, auf dem das Spielfeld erkannt werden kann, der Ball gesucht wird. Durch diese Vorgehensweise können Nahaufnahmen zwischen den Ballwechseln oder Werbepausen automatisch erkannt und ignoriert werden. Infolgedessen soll als nächstes die Spielfelderkennung mit Hilfe von Parametern optimiert werden.

Neben der ausschließlichen Betrachtung der Konturen gibt es, wie in Kapitel 4.2 beschrieben, weitere Möglichkeiten die Spielfelderkennung zu verbessern. Zum einen können anhand eines **Parameters kurze Linien** der erkannten Konturen entfernt werden, sodass aus diesen ein trapezförmiges Rechteck (perspektivisch verzerrtes Spielfeld) resultiert. In dieser Analyse wird dafür der Parameter *kurze_Linien*, der die Werte 0 und 1 annehmen kann, verwendet. Weiterhin

kann durch den **Parametern *min_Linienlänge*** und ***max_Linienlänge*** beeinflusst werden, wann Linien als kurz angesehen werden. Die Linienmaße werden hierbei in Pixel angegeben, sind aber prozentual von der gesamten Breite des Bildes abhängig. Im Grunde ist es das Ziel die erkannten Konturen so einzuschränken, dass alle Spielfeldlinien zwischen den Werten dieser beiden Parameter liegen.

Nach erstmalig erfolgreicher Spielfelderkennung ist es möglich die Suche für die **Erkennung der nächsten Spielfelder einzuschränken**. Dies wird durch den Parameter *verbesserte_Felderkennung* und den Werten 0, 1 und 2 gesteuert. Beträgt der Parameter den Wert 0 so erfolgt nach erfolgreicher Spielfelderkennung im nächsten Bildframe keine Einschränkung der Suche. Die Suche findet dann wie zuvor im gesamten Bild statt. Bei dem Wert 1 wird die Suche um das bisher erkannte Spielfeld herum eingegrenzt, sodass große Ausreißer in der Felderkennung entfernt werden. Der letzte Parameterwert 2 gibt an, dass das neue Spielfeld nur an den bisher erkannten Spielfeldecken gesucht wird, da davon auszugehen ist, dass sich die Spielfeldecken von Frame zu Frame nur geringfügig verändern.

Die Diagramme der Evaluierung sind vollständig im Anhang A.1.1 zu finden, da in den nachfolgenden Szenarien sehr viele verschiedene Parameterkonfigurationen auftreten und dadurch einige Diagramme erstellt wurden. Es sollen lediglich die wichtigsten Diagramme genauer erläutert und expliziert in den nachfolgenden Abschnitten dargestellt werden.

Grundsätzlich wurden die verschiedenen Parameter ebenfalls anhand den drei zuvor vorgestellten Videos untersucht und sind in den Diagrammen dargestellt. Dabei wurde die erfolgreiche Spielfelderkennung in Prozent in Abhängigkeit der jeweiligen Parameter angegeben. Die blauen Balken geben an, wie oft das Spielfeld erkannt wurde im Verhältnis zu den Spielfeldern die überhaupt hätten erkannt werden können. Weiterhin kann angenommen werden, dass sich die Größe eines erkannten Spielfeldes normalerweise nicht in jedem Bildframe vollständig verändert, daher repräsentiert der grüne Balken eine optimierte Spielfelderkennung, bei der davon ausgegangen wird, dass das Spielfeld mindestens zehn Bildframes gültig ist. Dadurch können kurzzeitige Probleme bei der Felderkennung abgefangen und die prozentualen Werte verbessert werden. Erst nachdem der Ball in zehn aufeinanderfolgenden Frames nicht erkannt wird, zählt das Spielfeld als nicht mehr erkannt.

In den Diagrammen (siehe Anhang A.1.1) für die Analyse der Spielfelder zeigt sich bei Video eins, dass selbst ohne Optimierung die Spielfelderkennung bei 99% liegt. Die optimierte Felderkennung beträgt bereits 100%. Infolgedessen kann der Parameter *verbesserte_Felderkennung* dies nicht wesentlich optimieren. Die Diagramme unterscheiden sich daher nur minimal. Anschließend soll zuerst die Konfiguration ohne *verbesserte_Felderkennung* betrachtet werden.

In den Grafiken für Videos zwei und drei zeigt sich eine deutliche Verbesserung der Spielfelderkennung für den **Parameter *kurze_Linien*** entfernen. Ohne diese Optimierung liegt der Wert für Video zwei bei 3% (17% optimierte Spielfelderkennung). Durch das Entfernen kurzer Linien, kann die Felderkennung auf 31 bis 51% (62 bis 85% optimierte Felderkennung) verbessert werden. In Video drei wird dadurch sogar überhaupt erst eine Spielfelderkennung ermöglicht. Ohne kurze Linien zu entfernen konnte kein Spielfeld erkannt werden. Durch diese Optimierung

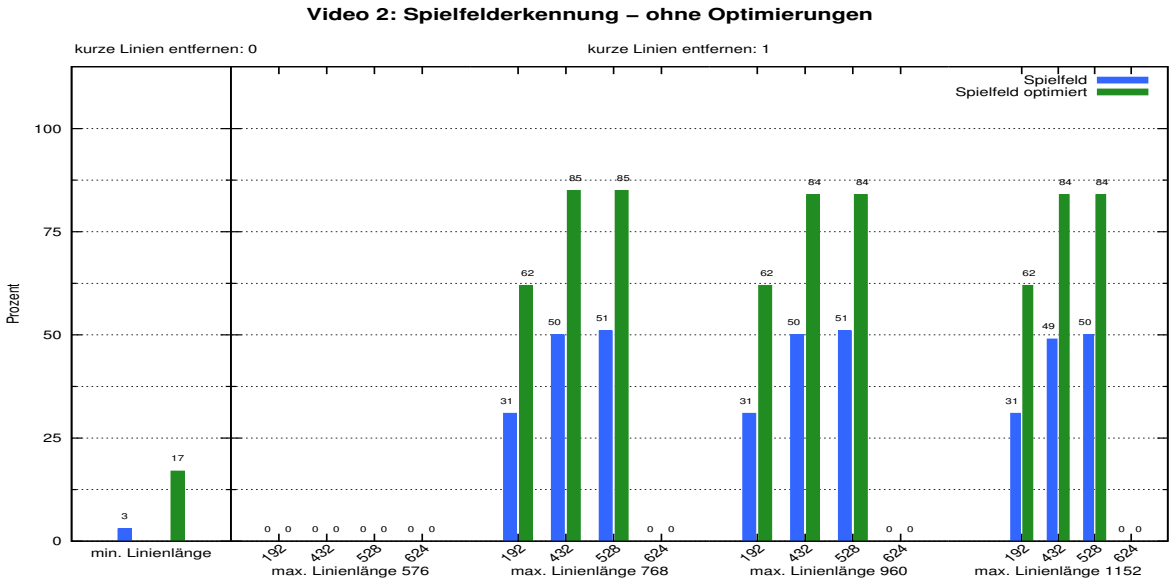


Abbildung 5.5.: Analyse der Spielfelderkennung: ohne Optimierungen nach erfolgreicher Felderkennung

werden 59 bis 65% (89 bis 93% optimierte Spielfelderkennung) erreicht.

Die Parameter für die **minimale und maximale Linienlänge** beeinflussen dabei die erlaubte Größe der Linien. Hier zeigt sich, dass Linien des Spielfeldes, die eine maximale Linienlänge von 576 überschreiten, nicht gültig sind. Daher stellt sich dieser Parameterwert als ungünstig dar, da hier die richtigen Spielfeldlinien ausgeschlossen werden. Bereits bei 768 können allerdings alle Linien erkannt werden, sodass nun auf den Parameter *minimale_Linienlänge* eingegangen werden kann. Bei Video zwei ist im Anhang A.1.1 zu sehen, dass die minimale Linienlänge kleiner als 624 betragen sollte, da sonst essentielle Linien der Spielfelder als zu kurz angesehen und damit entfernt werden. Dadurch ist keine Spielfelderkennung möglich. Weiterhin ist ein prozentualer Anstieg der Felderkennung von 31% bei einer minimalen Länge von 192 auf einen Wert von 50% bei 432 oder sogar auf 51% bei einer minimalen Linienlänge von 528 zu sehen.

In Video drei ohne weiteren Optimierungen nach erfolgreicher Spielfederkennung ist dagegen festzustellen, dass eine minimale Linienlänge von 192 keine Verbesserung, im Vergleich zu der Ausgangssituation, bei der keine kurzen Linien entfernt werden, ergibt. Erst ab 432 wird das Spielfeld in 59% der Fälle erkannt und kann bei einer minimalen Linienlänge von 528 auf 66% erhöht werden. Alles in allem zeigt sich, dass die minimale Linienlänge möglichst groß gewählt werden sollte, ohne dass wichtige Linien des Spielfeldes entfernt werden. Dies liegt, wie bei Video drei beschrieben, zwischen den Werten 432 und 528. Daher soll aus Sicherheitsgründen der Mittelwert davon verwendet werden, um Linien nicht fälschlicherweise zu entfernen. Dieser Wert beträgt 480. Auch für die maximale Linienlänge wird aus demselben Grund der Mittelwert von 768 und 1152 also 960 als geeigneten Wert für diesen Parameter gewählt.

Bei Betrachtung des **Parameters *verbesserte_Felderkennung*** zeigt sich von der Standard

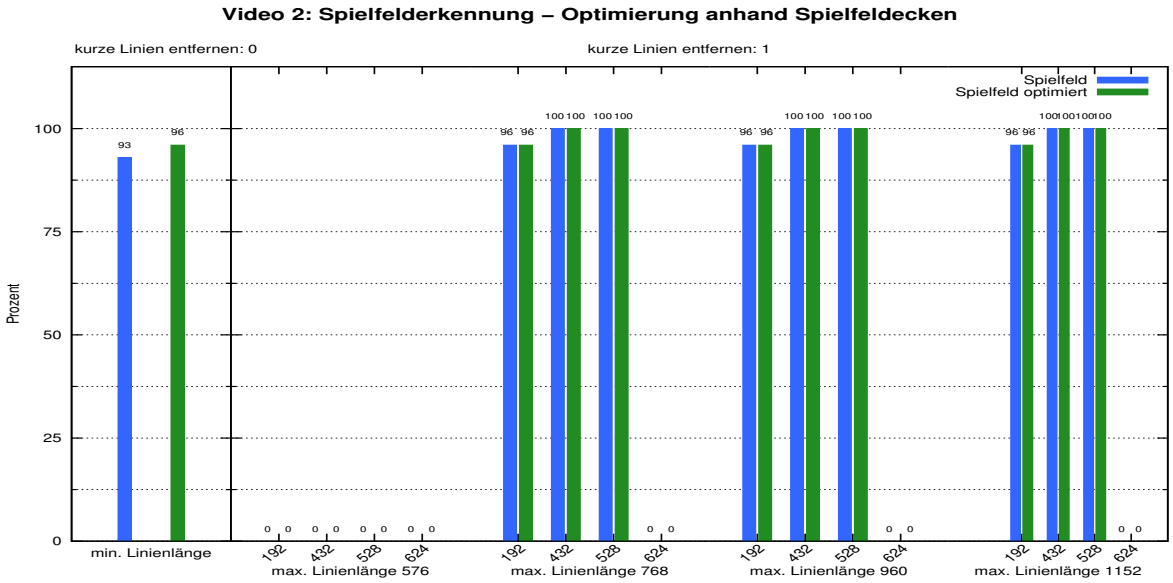


Abbildung 5.6.: Analyse der Spielfelderkennung: mit Optimierungen anhand der Spielfeldecken

Spielfelderkennung gegenüber der Optimierung anhand dem Spielfeldrand keine eindeutige Verbesserung. In Video zwei zeigt sich in Diagramm 5.5 für die Felderkennung bei einer Konfiguration von 528 minimale und 960 maximale Linienlänge ein Wert von 51% (84%). Durch die Optimierung anhand des Spielfeldrandes kann lediglich die optimierte Spielfederkennung auf 87% verbessert werden. Bei Video drei dagegen fiel die Erkennung bei gleicher Parameterkonfiguration für die Linienlänge sogar von 66% (93%) auf 58% (90%) ab.

Die Optimierung der Felderkennung anhand der Spielfeldecken dagegen bewirkt eine wesentliche Verbesserung der Spielfelderkennung (siehe Diagramm 5.6. Bei Video zwei kann der Ausgangswert bei Standard Spielfelderkennung von 51% auf 100% verbessert werden. Auch in Video drei konnte sie durch diese Optimierung auf 100% erhöht werden. Es zeigt sich, dass es sehr effektiv ist, die Suche nach dem neuen Spielfeld auf die zuvor erkannten Spielfeldecken einzuschränken, da sich das Spielfeld zum vorherigen Frame meist nur wenig verändert. Weiterhin werden dadurch Störungen von Spielern verhindert, die auf den Spielfeldlinien laufen und somit die erkannten Konturen des Spielfeldes vergrößern. Für gewöhnlich befinden sich die Spieler auf den Grundlinien, sodass bei den Spielfeldecken weniger Störungen auftreten. Aus diesen Gründen soll für die weiteren Untersuchungen die Spielfelderkennung anhand der Spielfeldecken verbessert werden, sowie kurze Linien entfernen, sodass nur Linien zwischen 480 (minimale Linienlänge) und 960 (maximale Linienlänge) als gültige Linie des Spielfeldes zählen.

5.1.3. Ballerkennung: Analyse des Suchbereiches

In den nächsten Abschnitten soll die Ballerkennung genauer untersucht werden. Es gibt verschiedene Ansätze den Ball identifizieren zu können, die im Kapitel 4.3 beschrieben wurden. Einer davon ist die Anpassung und Verschiebung eines Suchbereiches in die Richtung, in dem der Ball vermutet wird. Ein weiterer Ansatz ist den Ball anhand der Clustergröße und der Distanz zur letzten Ballposition zu bestimmen. Dabei gibt es Möglichkeiten die Ballgröße als auch die Distanz unterschiedlich festzulegen, sowie die Gewichtung dieser Größen für die Ballidentifikation zu variieren, sodass mehrere Parameterausprägungen getestet werden müssen, um die beste Parameterkonfiguration für die Ballerkennung herauszufinden.

Zuerst soll der Suchbereich genauer betrachtet werden. Hierbei geht es vor allem darum die Ballbewegung vorauszuahnen, sodass der Ball in einem kleineren Bereich des Bildes gesucht werden kann. Die Anpassung und Verschiebung des Suchbereiches kann anhand fünf Parametern gesteuert werden. Die Parameter *Berechnungsart*, *Skalierung_Breite* und insbesondere *kleinster_Suchbereich* bestimmen die Größe des Suchbereiches. Die Berechnungsart gibt an, ob sich die Breite des Suchbereiches linear oder nach einer exponentiellen Formel vergrößern soll. Die Skalierung der Breite beeinflusst ebenfalls wie stark sich die Breite verändern soll. Dadurch kann der Suchbereich je nach Situation vergrößert oder verkleinert werden, wobei er niemals die kleinste Breite des Parameters *kleinster_Suchbereich* unterschreiten darf. Kleinere Suchbereiche haben den Vorteil diesen genauer untersuchen zu können und sind performanter, da weniger Pixel betrachtet werden müssen. Der Ball kann aber schneller verloren gehen, da er sich bei plötzlichen Richtungswechseln leicht aus dem Suchbereich herausbewegen kann. Größere Suchbereiche dagegen können den Ball zuverlässiger nachverfolgen und führen dazu, dass der Ball seltener verloren geht. Anhand der nachfolgenden Analyse soll abgewogen werden, welche Parameterkonfiguration am besten für die Ballerkennung geeignet ist.

Diese soll zusätzlich die Parameter *Wiederholungsfaktor* und *Verschiebung* mit einbeziehen. Der *Wiederholungsfaktor* ist dafür da, den Ball besser wiederfinden zu können, wenn er im Suchbereich nicht mehr entdeckt wurde. Bis zu dreimal wird der Suchbereich anhand des Parameters *Wiederholungsfaktor* vergrößert (die genaue Berechnung ist in Kapitel 4.3.1 zu finden). Zuletzt kann der Suchbereich anhand des Parameters *Verschiebung* in eine bestimmte Richtung verschoben werden, in dem der Ball als nächstes vermutet wird. Dies kann anhand der Vorhersage des Kalman Filters erfolgen oder anhand der zuletzt erkannten Ballposition, da angenommen wird, dass der Ball normalerweise immer seine Bewegungsrichtung fortsetzt, abgesehen von den Treffpunkten des Balles mit dem Schläger und des Bodens.

Aufgrund der Einbeziehung von fünf Parametern müssen sehr viele Fälle getestet werden. Insgesamt ergeben sich dadurch pro Video 16 Diagramme, die im Anhang A.1.2 zu finden sind. Weiterhin kann auf Grund dieser großen Zahl an Diagramme nicht jedes Szenario einzeln betrachtet werden. Die gewonnenen Ergebnisse der Ballerkennung aus den Diagrammen sollen als Orientierung dienen potentiell gute Parameterkonfigurationen genauer untersuchen zu können. Grundsätzlich wird in den Diagrammen die Anzahl der erkannten Ballpositionen (blauer Balken) auf der Y-Achse dargestellt. Ein weiteres Kriterium über die Effektivität der Parameterauswahl

sollen die dafür notwendigen Versuche, den Ball zu finden, darstellen. Der grüne Balken gibt die Anzahl der Ballerkennungen an, die direkt im ersten Versuch gefunden wurden. Wird der Ball in einem Bildframe erst im zweiten Versuch gefunden, ist dies im gelben Balken gekennzeichnet. Erfolgt die Erkennung des Balles im dritten Versuch, wird dies anhand des roten Balken gezeigt.

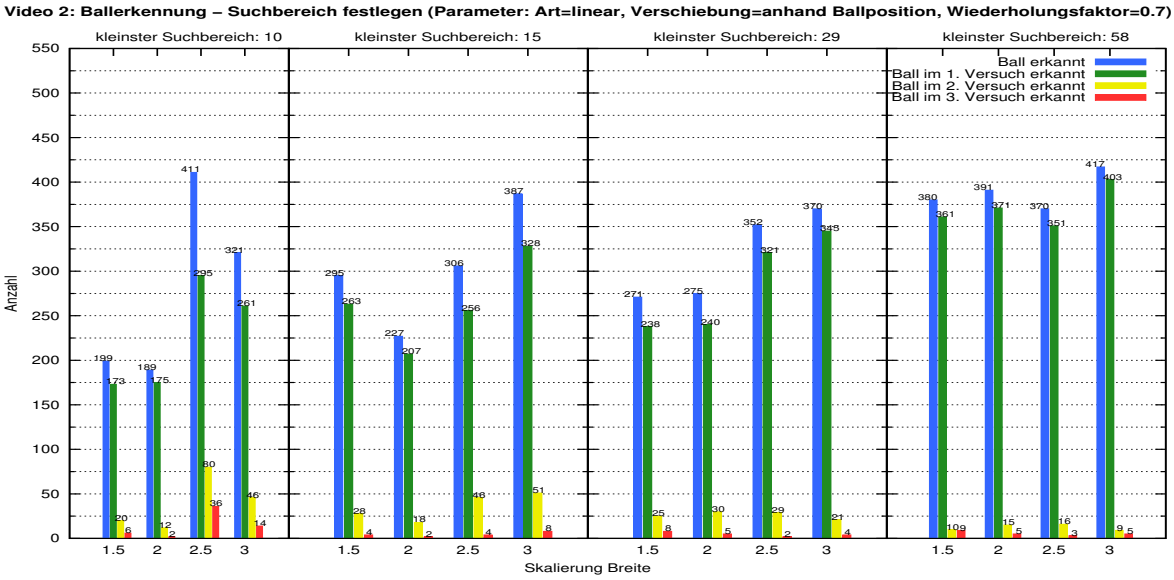


Abbildung 5.7.: Video 2: Ballerkennung - Analyse des Suchbereiches (Art = linear, Verschiebung = anhand Ballposition, Wiederholungsfaktor = 0.7)

In den Diagrammen zeigte sich, aufgrund der dargestellten Ergebnisse, kein klarer Zusammenhang der Parameter. Die Ergebnisse der Ballerkennung anhand den eben beschriebenen Parametern zeigte sich ehr zufällig. Auffällig ist bei kleineren Suchbereichen, wie beispielsweise im Suchbereich 10, dass mehr Ausreißer nach oben als auch nach unten festzustellen sind. Dieses Verhalten lässt sich durch Verwechslungen bei der Ballidentifikation mit dem Spieler erklären, bei denen nicht automatisch erkannt wird, dass das der Ball über eine längere Zeit falsch identifiziert wurde. Weiterhin zeigt sich, dass je größer der Suchbereich ist, desto gleichmäßiger scheint die Ballerkennung in Abhängigkeit des **Parameters *Skalierung_Breite*** zu sein. Dies zeigt sich beispielsweise in Diagramm 5.7. Darin ist ebenfalls zu erkennen, dass bei einem größeren Suchbereich der Ball meist im ersten Versuch gefunden wird und weniger Zweit- und Drittversuche notwendig sind.

Die manuelle Betrachtung der einzelnen Fälle zeigte, dass in kleineren Suchbereichen der Ball zwar in der Nähe des Spielers noch genau erkannt werden kann, aber der Ball, bei verschmelzen des Ballclusters mit dem Spieler, verloren geht und der Spieler beziehungsweise Körperteile von ihm häufig fälschlicherweise als Ball identifiziert werden. Je größer der Suchbereich ist, desto seltener geht der Ball verloren sowie die Falschidentifikationen mit dem Spieler nehmen ab. Allerdings kann dadurch der Ball in der Nähe des Spielers nicht mehr so genau wahrgenommen werden. Da allerdings in dieser Arbeit die exakte Position des Balles nicht immer von höchster

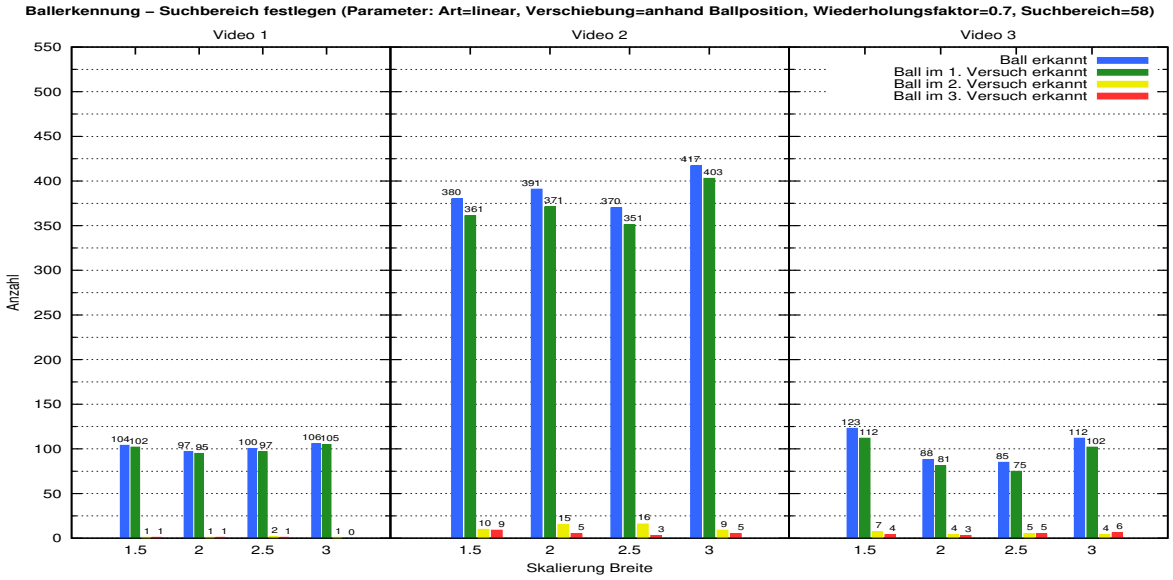


Abbildung 5.8.: Ballerkennung: Analyse des Suchbereiches (Art = linear, Verschiebung = anhand Ballposition, Wiederholungsfaktor = 0.7, Suchbereich = 58)

Wichtigkeit ist, sondern die grundsätzliche Flugkurve und grobe Positionen des Balles ausreichend sind, ist eine ungenaue Ballbestimmung bei Näherkommen des Balles zum Spieler zu vernachlässigen. Aus diesen Gründen sollen im Nachfolgenden ausschließlich die Ergebnisse des Suchbereiches mit 58 betrachtet werden.

Für einen besseren Überblick wurden neue Diagramme, ohne kleinere Suchbereiche, erstellt. Diese sind ebenfalls im Anhang A.1.2.5 zu finden. Dort wird pro Parameterkonfiguration ein Diagramm erstellt, die die Ergebnisse der drei Videos enthalten. Es sollen nun die erfolgversprechendsten Ergebnisse der Ballerkennung manuell genauer evaluiert werden.

Zu Beginn wird hierbei die Parameterkonfiguration für eine **lineare Berechnungsart** mit einer **Verschiebung, die anhand der Ballposition** durchgeführt wird, betrachtet. Dort zeigten sich gute Werte für die Ballerkennung bei einem Wiederholungsfaktor von 0.7 und einer Skalierung der Breite des Suchbereiches mit dem Faktor 3. Bei dieser Konfiguration ist in Diagramm 5.8 zu erkennen, dass für Video eins insgesamt 106 Ballpositionen erkannt wurden. 107 erkannte Ballpositionen stellt den Bestwert aller Parameterkonfigurationen für dieses Video dar. In Video zwei konnte für diese Konfiguration 417 (Bestwert für dieses Video: 451) mal der Ball identifiziert werden. Video drei zeigte sich auch mit 112 Ballerkennungen (Bestwert: 130) ebenfalls in einem guten Bereich. Bei manueller Betrachtung dieses Falles zeigte sich in diesen drei Videos, dass der Ball zwar auch mal verloren geht, aber sich nicht bei einem Spieler verfängt und somit nicht dauerhaft als Ball ansieht. Es spricht für eine gute Ballerkennung, wenn für diesen Fall viele Ballpositionen festgestellt werden konnten und das obwohl der Ball nicht immer erkannt werden konnte und für eine Weile verloren ging.

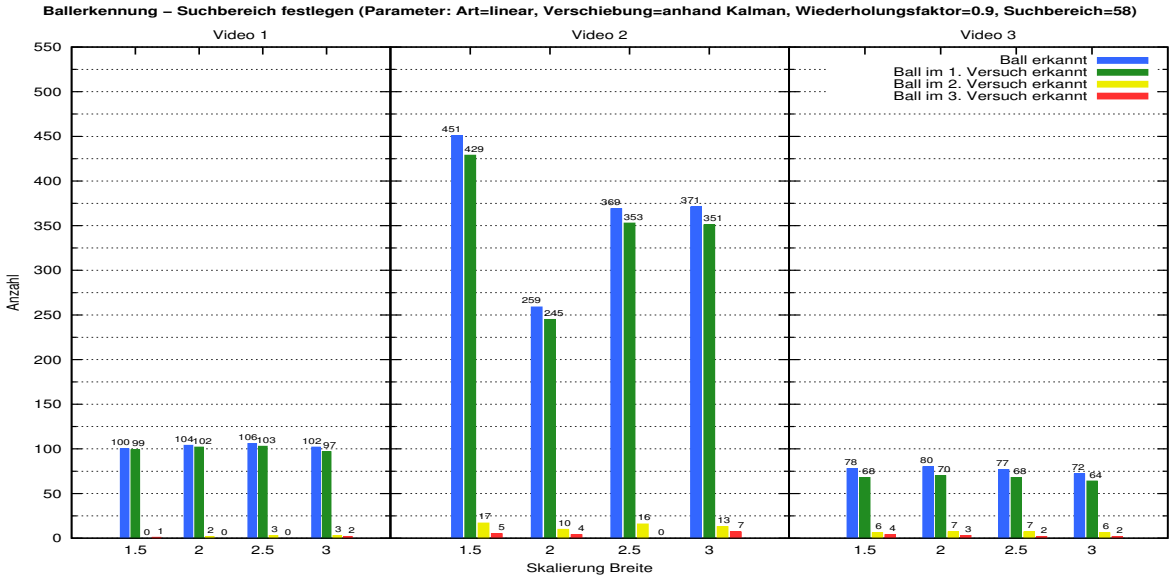


Abbildung 5.9.: Ballerkennung: Analyse des Suchbereiches (Art = linear, Verschiebung = anhand Kalman, Wiederholungsfaktor = 0.9, Suchbereich = 58)

In diesem Diagramm ist das Ergebnis für die Skalierung der Breite mit einem Faktor von 1.5 ebenfalls vielversprechend. Zwar war die Anzahl der Ballerkennung in Video eins mit 104 und auch in Video zwei mit 380 geringer als im vorherigen Fall, so konnte Video drei mit 123 erkannten Ballpositionen ein sehr gutes Ergebnis erzielen. Dies bestätigte sich für dieses Video auch durch manuelle Überprüfung. Allerdings wurde in Video zwei sichtbar, dass der Ball falsch identifiziert und daher mit dem Spieler verwechselt wurde, wodurch die Ergebnisse verfälscht wurden. Da die Ballerkennung in diesem Fall für Video eins und Video zwei deutliche Einbußen gegenüber der vorherigen Konfiguration aufweist, sollte diese Parameterkonfiguration nicht weiter betrachtet werden.

Ein sehr gutes Ergebnis für Video stellt sich in Diagramm 5.9 dar, bei dem neben einer **linearen Berechnungsformel die Kalman Filter Vorhersage für die Bestimmung der Verschiebung des Suchbereiches** verwendet wird. Bei einer Skalierung der Breite mit dem Faktor 1.5 konnte der Ball dort 451 mal identifiziert werden, was den Bestwert aller Parameterkonfigurationen für Video zwei darstellt. Auch bei manueller Überprüfung konnte dieser Fall überzeugen. Allerdings fiel die Erkennung des Balles bei Video eins mit 100 und bei Video drei sogar mit nur 78 sehr schlecht aus. Daher kommt auch diese Konfiguration nicht in Frage.

Zuletzt soll in Diagramm 5.10 ein Fall genauer betrachtet werden, bei dem die Breite des Suchbereiches anhand einer **exponentiellen Berechnungsformel** mit einem Wiederholungsfaktor von 0.9 ermittelt wird. Außerdem wird die **Verschiebung anhand der Vorhersage des Kalman Filters** vorgenommen. Dort sticht für die Skalierung der Breite mit dem Faktor 0.3 insbesondere die Anzahl der Ballerkennungen für Video drei hervor. Hier wurde der Bestwert

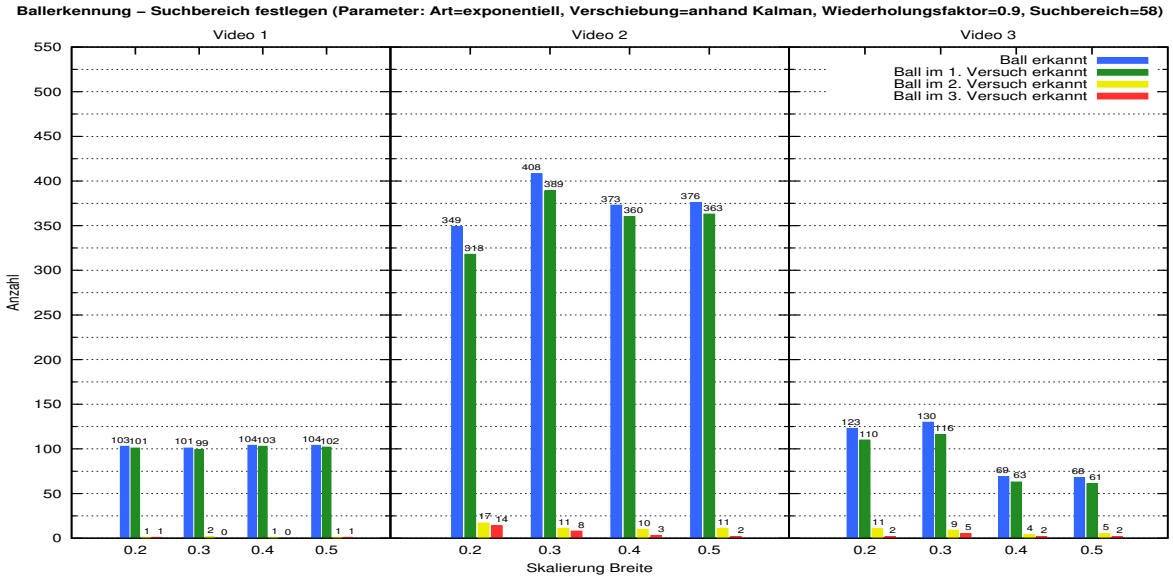


Abbildung 5.10.: Ballerkennung: Analyse des Suchbereiches (Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9, Suchbereich=58)

für Video drei mit 130 erkannten Ballpositionen erreicht. Auch Video zwei zeigt sich mit 408 Ballpositionen vielversprechend. Dies wurde durch manuelle Überprüfung dieser beiden Videos bestätigt, bei dem lediglich kleine Ausreißer und Falschidentifikationen zu finden waren. Video eins dagegen stellte sich, mit nur 101 erkannten Ballpositionen als unzufriedenstellend heraus.

Alles in allem konnte die Parameterkonfiguration, bei der die Berechnungsart linear mit einem minimalen Suchbereich von 58 Pixeln, einem Skalierungsfaktor der Breite von 3 und eines Wiederholungsfaktors von 0.7, sowie die Verschiebung des Suchbereiches anhand der zuletzt erkannten Ballpositionen durchgeführt wird, am zuverlässigsten den Ball erkennen. Dieser Fall stellte zwar keine Bestwerte auf, konnte aber in allen drei Videos mit 106 im ersten, mit 417 im zweiten und mit 112 erkannten Ballpositionen im dritten Video gute Ergebnisse erzielen, ohne das wesentliche Störungen durch Falschidentifikationen stattfanden. Aus diesen Gründen soll in den nachfolgenden Szenarien diese Parameterkonfiguration für die Anpassung und Verschiebung des Suchbereiches gewählt werden.

5.1.4. Ballerkennung: Analyse der Ballgrößen- und Distanzbestimmung

Bei der Ballerkennung in dieser Arbeit wird die Ballgröße anhand der Anzahl der Punkte, aus denen die Kontur eines Clusters besteht, festgelegt. Da diese je nach Bildbereich und Spielsituation variieren kann, sollen nachfolgend drei Varianten getestet werden, die die Ballgröße unterschiedlich ermitteln. Mit dem Parameter *Ballhistorie* kann eine der Varianten ausgewählt werden, indem er die Werte eins, zwei oder drei annimmt. Ballhistorie eins ermöglicht die Bestimmung

der Ballgröße anhand des Durchschnittes der letzten beiden erkannten Ballpositionen. Ist der Wert des Parameters *Ballhistorie* zwei so wird die durchschnittliche Größe des Balles der letzten vier Positionen als neue Ballgröße angenommen. Zuletzt soll eine Variante getestet werden, bei der die Ballgrößen der letzten vier erkannten Ballpositionen gewichtet eingehen. Bei der zuletzt genannten Variante gehen, wie in Kapitel 4.3.3 beschrieben, die zuletzt erkannten Ballgrößen stärker in die neue Ballgröße ein, als die vorherigen. Diese Variante wird durch den Wert drei des Parameters *Ballhistorie* dargestellt. Weiterhin soll in der nachfolgenden Analyse die Bestimmung der Distanz mit einbezogen werden. Hierbei beeinflusst der Parameter *Distanz_Ausgangspunkt* den Ausgangspunkt für die Distanzbestimmung. Die Distanz wird vom Ausgangspunkt zum zu untersuchenden Cluster ermittelt. Diese kann ausgehend von der Kalman Vorhersage oder auch von der zuletzt erkannten Ballposition bestimmt werden. Weiterhin kann bei Verlust des Balles angenommen werden, dass die nächste Ballposition weiter von der letzten erkannten Position entfernt ist, da der Ball sich normalerweise immer fortbewegt. Um den Ball trotzdem nicht zu verlieren, kann durch einen Faktor, der durch den Parameter *Distanzkorrekturfaktor* angegeben wird, die Distanz zum untersuchenden Cluster etwas reduziert werden, sodass ein verlorengegangener Ball nicht so stark ins Gewicht fällt und die neue Ballposition leichter wiedergefunden werden kann. Aufgrund der eben dargestellten Möglichkeiten die Ballerkennung zu beeinflussen, sollen in den nächsten Abschnitten die besten Parameterwerte für die Bestimmung der Ballgröße und Distanz ermittelt werden.

Die Diagramme dieses Szenarios sind in Anhang A.1.3 zu finden. Durch diese soll eine Konfiguration der eben beschriebenen Parameter gefunden werden, die die Ergebnisse aus der vorherigen Analyse des Suchbereiches weiter verbessern. Diese lagen für Video eins bei 106, für Video zwei bei 417 und für Video drei bei 112 erkannten Ballpositionen. Dieses Ausgangsergebnis wurde, wie auch in den Diagrammen im Anhang A.1.3 zu sehen ist, mit folgender Parameterkonfguration für die Größen- und Distanzbestimmung erreicht und sollen als Vergleich für die weiteren Optimierungen dienen:

- Parameter *Ballhistorie*: 3
- Parameter *Distanz_Ausgangspunkt*: anhand letzter Ballposition
- Parameter *Distanzkorrekturfaktor*: 5

Ausgehend von der im vorherigen Kapitel 5.1.3 bestimmten Parameter, die den Suchbereich festlegen, werden nun die Parameter für die Bestimmung der Ballgröße und Distanz untersucht. Bei Verwendung der **Vorhersage des Kalman Filters als Ausgangspunkt der Distanzbestimmung** zeigte sich sowohl für Video eins als auch für Video zwei ein deutliches Einbußen gegenüber der Vergleichskonfiguration. Lediglich in Video drei wurde für Historie drei, bei der die Ballgröße gewichtet eingeht, eine Verbesserung von 112 auf 116 erkannte Ballpositionen sichtbar.

Bei **Bestimmung der Distanz ausgehend von der letzten Ballposition** ist in Diagramm 5.11 für Video zwei eine Verbesserung gegenüber der Vergleichskonfiguration festzustellen, indem der Distanzkorrekturfaktor auf 10 erhöht wurde. Dadurch konnte die Anzahl der erkannten Ballpositionen für Historie drei von 417 auf 429, bei Historie eins sogar auf 435 erhöht werden.

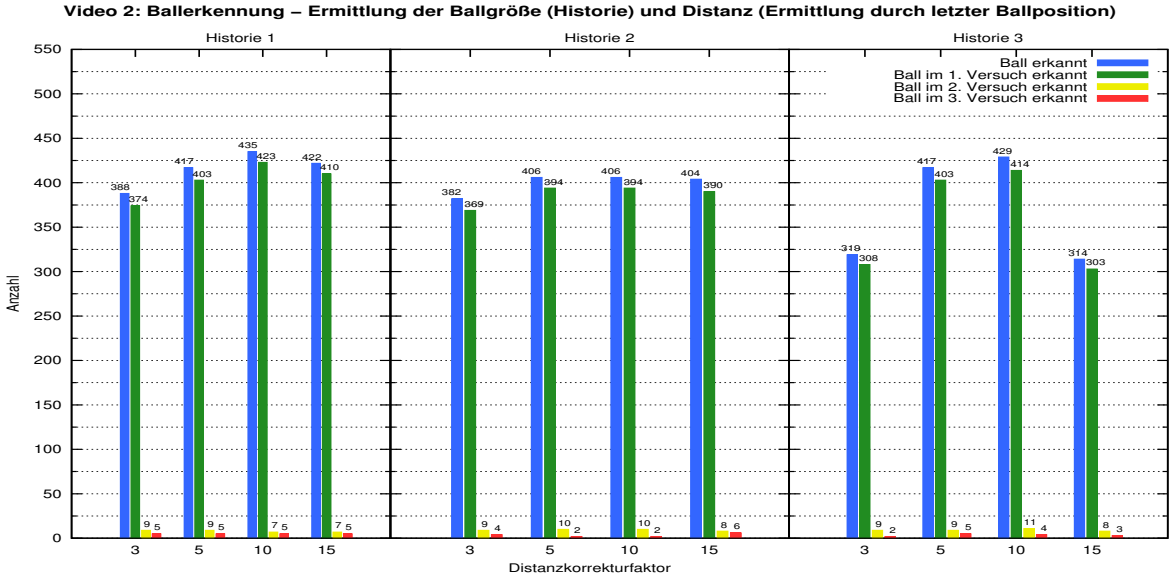


Abbildung 5.11.: Video 2: Ballerkennung - Analyse der Ballgröße (Historie) und Distanz (Ermittlung durch letzter Ballposition)

Eine Erhöhung des Distanzkorrekturfaktors könnte dazu beitragen den Ball auch nach größerer zurückgelegter Strecke, beispielsweise wenn die Ballposition in der Nähe eines Spielers verloren gegangen ist, leichter wieder zu finden. Der Parameter *Historie* bewirkt für die Distanzkorrekturfaktoren von 3 und 15 eine Verbesserung in der Ballerkennung für Historie eins. Dort konnte die Ballerkennung gegenüber der Historie drei deutlich verbessert werden. Für den Faktor 3 konnten dadurch die Anzahl der Ballpositionen von 319 (Historie drei) auf 388 (Historie eins) gesteigert werden, für den Faktor 15 sogar von 314 auf 422. Den besten Wert für Video zwei erzielte allerdings der Distanzkorrekturfaktor von 10 mit 435 ebenfalls für Historie eins. Diese Verbesserung bei Ermittlung der neuen Ballgröße anhand des Durchschnittes der letzten beiden Ballgrößen lässt sich durch eine schnellere Anpassung der aktuellen Ballgröße an veränderte Situationen im Ballwechsel erklären. Diese Situationen könnten die Ballerkennung beeinflussen, da sich der Ball im unteren Bereich des Bildes näher an der Kamera befindet und somit größer als im oberen Spielfeldbereich dargestellt wird. Auch deswegen zeigte sich in den meisten Fällen Historie drei gegenüber Historie zwei, abgesehen von den Korrekturfaktor von 3 und 15 in Video zwei, besser, da ebenfalls die zuletzt erkannte Ballgröße stärker gewichtet wird.

In Video drei (siehe Diagramm 5.12) zeigte sich für einen Distanzkorrekturfaktor von 10, sowohl für Historie eins als auch Historie drei, eine geringfügige Verbesserung von 112 der Ausgangskonfiguration (Historie drei und Distanzkorrekturfaktor von fünf) auf 116 Ballerkennungen. Hier scheint der Parameter *Historie* allerdings nur geringen Einfluss zu haben, da lediglich bei Historie zwei die Werte für die Ballerkennung gesunken ist. Im Vergleich der Historie eins und drei sind keine Unterschiede festzustellen. Manuelle Überprüfungen bestätigten die Resultate aus den Diagrammen, dass die Ballerkennung mit einem Distanzkorrekturfaktor von 10 gegenüber

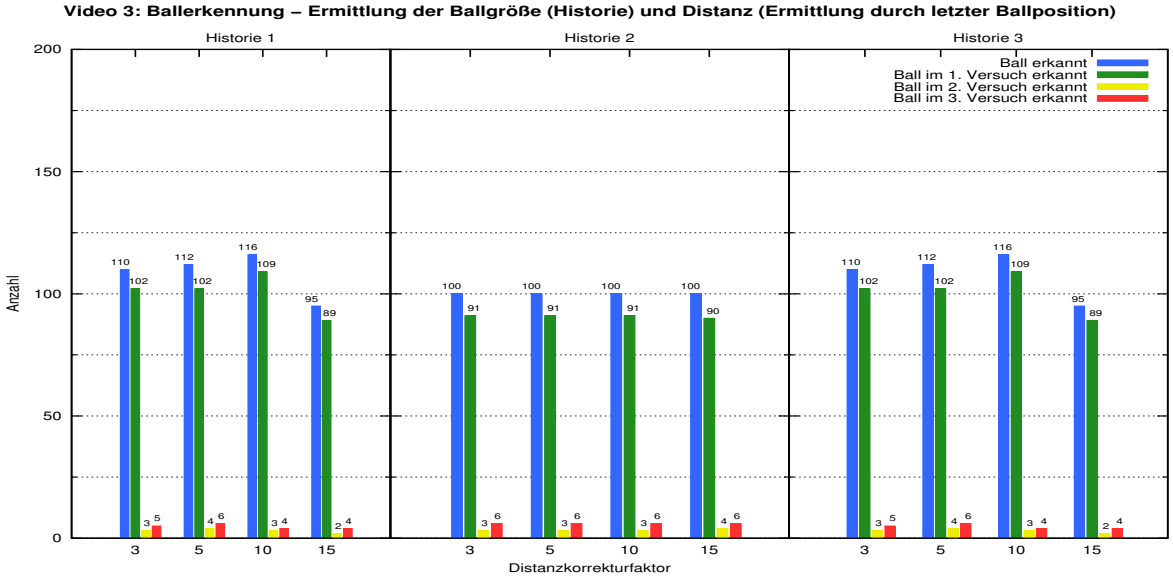


Abbildung 5.12.: Video 3: Ballerkennung - Analyse der Ballgröße (Historie) und Distanz (Ermittlung durch letzter Ballposition)

der Ausgangskonfiguration leicht verbessert werden konnte, da die Ballerkennungen von 112 auf 116 erhöht werden konnten und keine Ausreißer in Form von Fehlidentifikationen festzustellen waren. In Video eins (siehe Diagramm im Anhang A.1.4, zeigte sich dagegen bei keinen der Parameterkonfigurationen eine Veränderung und lag in allen Fällen bei 106 Ballerkennungen.

Aus diesen Gründen sollen für die Parameter für die Bestimmung der Ballgröße der Durchschnitt der letzten beiden Ballerkennungen, für die Bestimmung der Distanz als Ausgangspunkt die letzte erkannte Ballposition, sowie ein Distanzkorrekturfaktor von 10 angenommen werden. Dadurch konnten die Anzahl der Ballerkennung im Vergleich zur Vergleichskonfiguration in Video zwei um +4.3% auf 435 und in Video drei um +3.6% auf 116 Ballerkennungen erhöht werden und zeigte damit das vielversprechendste Ergebnis. In Video eins ergab sich keine Verbesserung, sodass hier weiterhin 106 Ballpositionen festgestellt wurden.

5.1.5. Ballerkennung: Analyse der Parameter für die Ballidentifikation

Die letzte Analyse, um die Parameter für die Ballerkennung zu optimieren, beschäftigt sich mit der Ballidentifikation. In dieser Masterarbeit wird der Ball anhand der Differenz zwischen aktueller Ballgröße und der zu untersuchenden Größe des Clusters, sowie der Distanz zwischen zuletzt erkannter Ballposition und der zu untersuchenden Clusterposition identifiziert. Hierbei kann anhand des Parameters *Berechnungsformel* die Art und Weise, wie diese Identifikation abläuft, beeinflusst werden. Zum einen kann die Differenz linear in die Überprüfung zur Identifikation des Balles eingehen oder aber auch exponentiell. Weiterhin können sowohl die Distanz

als auch die Ballgröße durch die Parameter *Skalierung_Distanz* und *Skalierung_Differenz* gewichtet werden. Zuletzt kann mit dem Parameter *gültiger_Bereich* Einfluss auf die Akzeptanz der Ballidentifikationen genommen werden. Je kleiner dieser Wert, desto genauer muss die Größendifferenz, sowie die Distanz von aktueller zu untersuchendem Cluster übereinstimmen, damit das untersuchte Cluster als Ball betrachtet wird. Je größer dieser Wert ist, desto toleranter wird mit der Differnz und der Distanz umgegangen.

Auch in diesem Szenario soll die Parameterkonfiguration aus Kapitel 5.1.3 für den Suchbereich als Vergleichskonfiguration angenommen werden. Die Ballerkennung soll durch die Analyse der Parameter für die Ballidentifikation weiter verbessert werden. Die Diagramme dafür sind vollständig in Anhang A.1.4 dargestellt. In den nachfolgenden Abschnitten sollen nur die wichtigsten Diagramme abgebildet und detailliert beschrieben werden.

Im Allgemeinen zeigte sich für die Parameterkonfiguration mit **linearer Berechnungsformel**, dass die Ballerkennung, gerade bei einem **kleineren gültigen Bereich**, schlechte Ergebnisse erzielt. In diesen Fällen geht der Ball leicht verloren und wird nicht wiedergefunden. Steigt der gültige Bereich auf 35% der Bildbreite an, so kann die Ballidentifikation gegenüber der Vergleichskonfiguration verbessert werden. Dies stellte sich ebenfalls bei Video eins für die Skalierung der Differenz mit dem Faktor 0.25 dar, da in diesem Fall 107 Ballpositionen erkannt wurden. Auch nach manueller Überprüfung zeigte sich, dass sich die Ballidentifikation bei Video eins als Verbesserung erwies.

Auch in Video zwei zeigte sich dieser Verlauf, dass sich bei **Vergrößerung des gültigen Bereiches** die Anzahl der Ballerkennungen erhöhte, ohne dass gravierende Störungen durch Falschidentifikationen auftraten. Die beste Ballerkennung in Video zwei mit einer linearen Berechnungsformel liegt bei 418 Ballpositionen für einen gültigen Bereich von 21% der Bildbreite. In Video drei dagegen ist die Ballerkennung deutlich schlechter als bei der Vergleichskonfiguration. Dort beträgt die größte Anzahl an erkannten Ballpositionen 91. Daher scheint die lineare Berechnungsformel nicht die beste Wahl zur Ballidentifikation zu sein. Der Parameter *Skalierung_Distanz* dagegen scheint grundsätzlich nur geringen Einfluss auf die Ballerkennung zu haben.

Bei der **exponentiellen Berechnungsformel** zeigt sich ein ähnlicher Zusammenhang des **gültigen Bereiches** wie bei der linearen Berechnungsformel. In Diagramm 5.13 zeigt sich für Video eins ebenfalls für die meisten Parameterkonfigurationen bei einem gültigen Bereich von 35% eine sehr gute Ballerkennung. Aus den bisher beschriebenen Gründen deutet es darauf hin, dass ein gültiger Bereich von 35% der Bildbreite eine gute Wahl darstellt um die Ballerkennung zu verbessern. Weiterhin sticht bei einer exponentiellen Berechnungsformel der Faktor für die Skalierung der Differenz von 0.003 heraus, mit dem ein sehr gutes Ergebnis erzielt werden konnte. Dieser ermöglichte bei Video eins 107 Ballerkennungen, die alle im ersten Versuch entdeckt werden konnten. Der Parameter *Skalierung_Distanz* beeinflusste die Ballerkennung für Video eins in keiner der jeweiligen Parameterausprägungen.

In Diagramm 5.14 wird die Anzahl der Ballerkennung für Video zwei ebenfalls bei einem **gültigen Bereich** von 35% der Bildbreite dargestellt. Darin wurde ersichtlich, dass mit einer

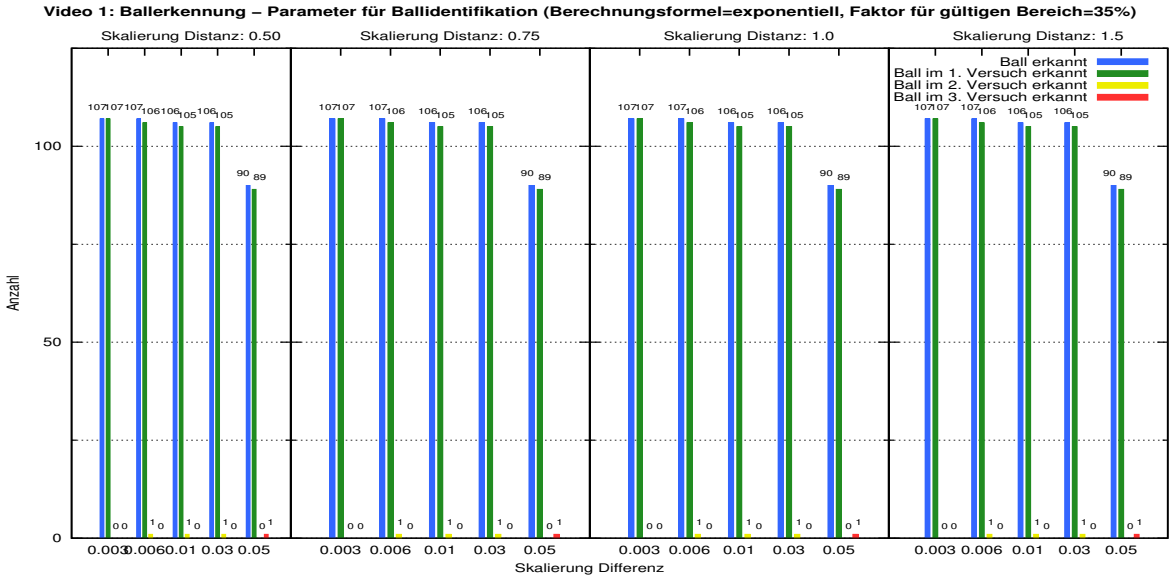


Abbildung 5.13.: Video 1: Ballerkennung - Analyse der Parameter für die Ballidentifikation (Berechnungsformel = exponentiell, Faktor für gültigen Bereich = 0.35)

exponentiellen Berechnungsformel für Video zwei die besten Ergebnisse aller Parameterkonfigurationen möglich waren. Bei einer Skalierung der Differenz mit dem Faktor 0.003 konnten unabhängig von der Skalierung der Distanz 486 Ballpositionen erkannt werden. Gegenüber der Vergleichskonfiguration ist dies ein Anstieg von +16.5%. Der Parameter *Skalierung_Distanz* zeigte lediglich für Faktor 0.5 gegenüber den anderen Konfigurationen für die Skalierung der Differenz von 0.006 und 0.03 andere Ergebnisse gegenüber den anderen Distanz Skalierungen. Es ist allerdings weiterhin kein Zusammenhang für die Skalierung der Distanz zu erkennen, da die Ergebnisse der anderen drei dargestellten Faktoren identisch sind.

Auch für Video drei (siehe Diagramm 5.15) kann die Ballerkennung mit der gleichen Parameterkonfiguration wie in den vorherigen Videos, also bei exponentieller Berechnungsformel mit einem Faktor der Differenzskalierung von 0.003, die besten Ergebnisse erzielen und erkannte gegenüber der Vergleichskonfiguration um +9.8% mehr Ballpositionen, sodass die Anzahl der Ballpositionen von 112 auf 123 gesteigert werden konnte. Dies bestätigt die Ergebnisse aus den anderen Videos.

Zusammenfassend wurde in den Diagrammen ersichtlich, dass die Anzahl der erkannten Ballpositionen für einen größeren gültigen Bereich höher ist, als bei einem kleineren gültigen Bereich. Dies ist nachvollziehbar, da der Ball auch in größeren Distanzen zur zuletzt erkannten Position oder wenn sich die Größe des zuletzt erkannten Balles gegenüber der zu untersuchenden Clustergröße weiter unterscheidet, wiedergefunden werden kann. Infolgedessen treten aber mehr Falschidentifikationen auf, besonders wenn der Ball in der Nähe des Spielers ist oder sogar vom Spieler verdeckt wurde und so nicht detektiert werden kann. Da die exakte Position des Balles

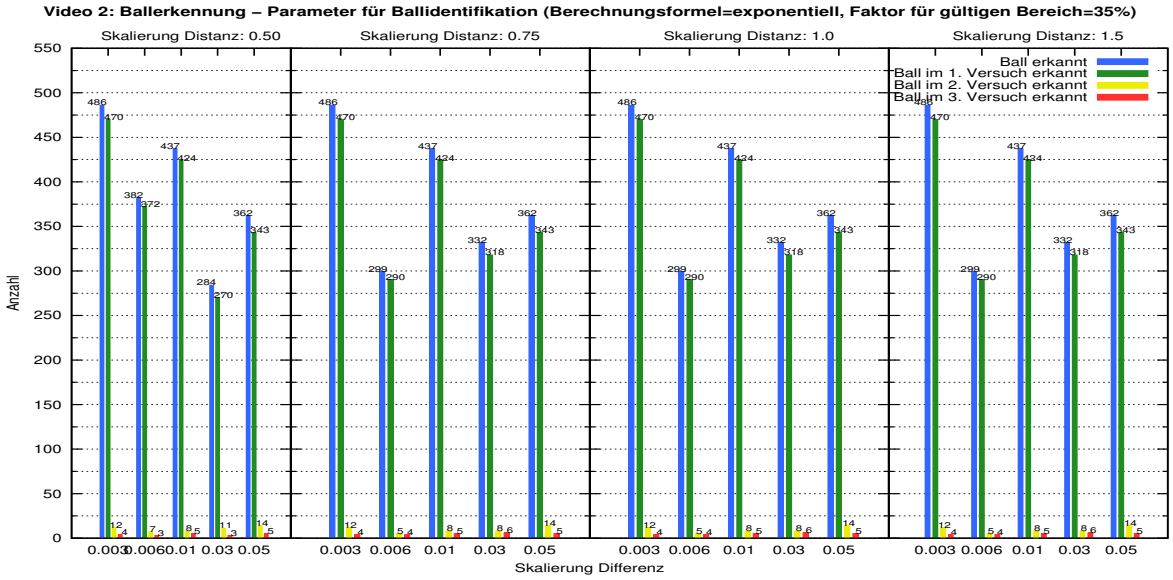


Abbildung 5.14.: Video 2: Ballerkennung - Analyse der Parameter für die Ballidentifikation (Berechnungsformel = exponentiell, Faktor für gültigen Bereich = 0.35)

nicht immer wichtig ist, kann dies aber vernachlässigt werden, sofern der Ball möglichst bald nach einer Falschidentifikation wieder bestimmt werden kann. Dazu trägt ein großer gültiger Bereich bei. Solch ein Szenario ist in Abbildung 5.16 dargestellt. Dort wird der Ball bei dem oberen Spieler kurzzeitig nicht erkannt, sodass fälschlicherweise Teile des Körpers als Ball eingestuft werden. Nach kurzer Zeit kann der Ball allerdings wiedergefunden werden. Dies zeigt sich auch etwas später bei dem unteren Spieler.

Aufgrund der Eigenschaft, dass der Ball nach Falschidentifikationen bei einem größeren gültigen Bereich besser wiedergefunden werden kann, wird in den weiteren Auswertungen eine exponentielle Berechnungsformel mit einer Skalierung der Differenz mit dem Faktor 0.003 sowie einen maximal gültigen Bereich von 35% der Bildbreite als sinnvolle Parameterkonfiguration zur Ballidentifikation angesehen. Der Parameter für die Skalierung der Distanz zeigte wenig Einfluss auf die Ballerkennung, blieb aber insbesondere bei den Faktoren für 0.75, 1.0 und 1.5 größtenteils gleich. Daher soll der mittlere Faktor für ihn gewählt werden, da in allen drei die Ballerkennung stabil stattfinden konnte und dadurch keine plötzlichen Abweichungen in der Ballerkennung zu erwarten sind.

5.2. Auswertung der Ergebnisse

Nachdem in den vorherigen Kapiteln die Parameter optimiert wurden, sollen hier die Ergebnisse dazu zusammengefasst werden, um die Parameterkonfiguration für die weitere Auswertung ersichtlich zu machen. In Tabelle 5.1 ist die endgültige Parameterkonfiguration aufgezeigt.

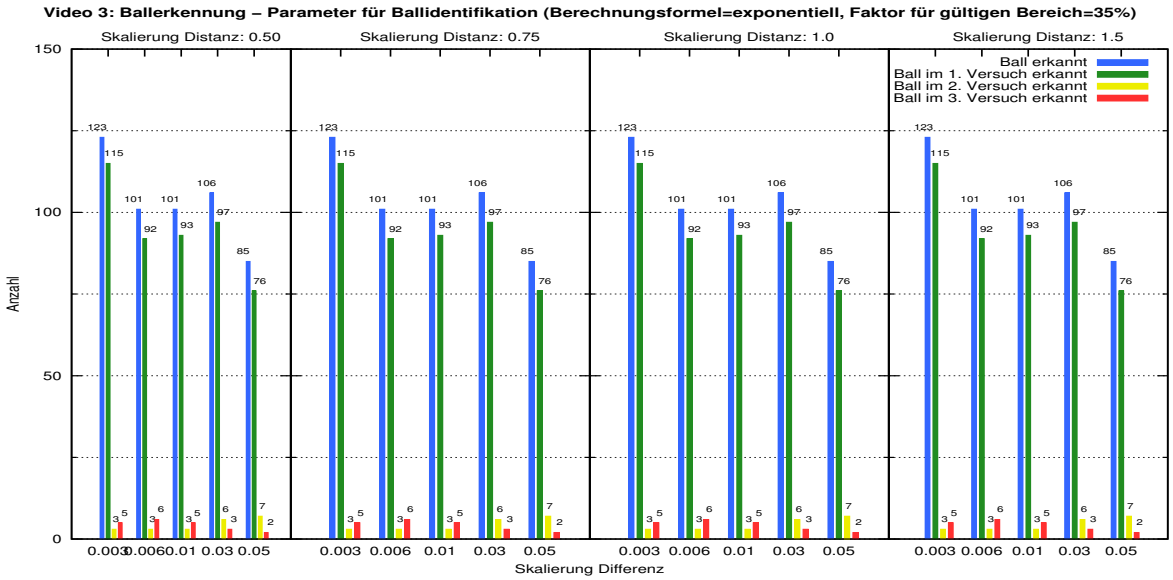


Abbildung 5.15.: Video 3: Ballerkennung - Analyse der Parameter für die Ballidentifikation (Berechnungsformel = exponentiell, Faktor gültiger Bereich = 0.35)

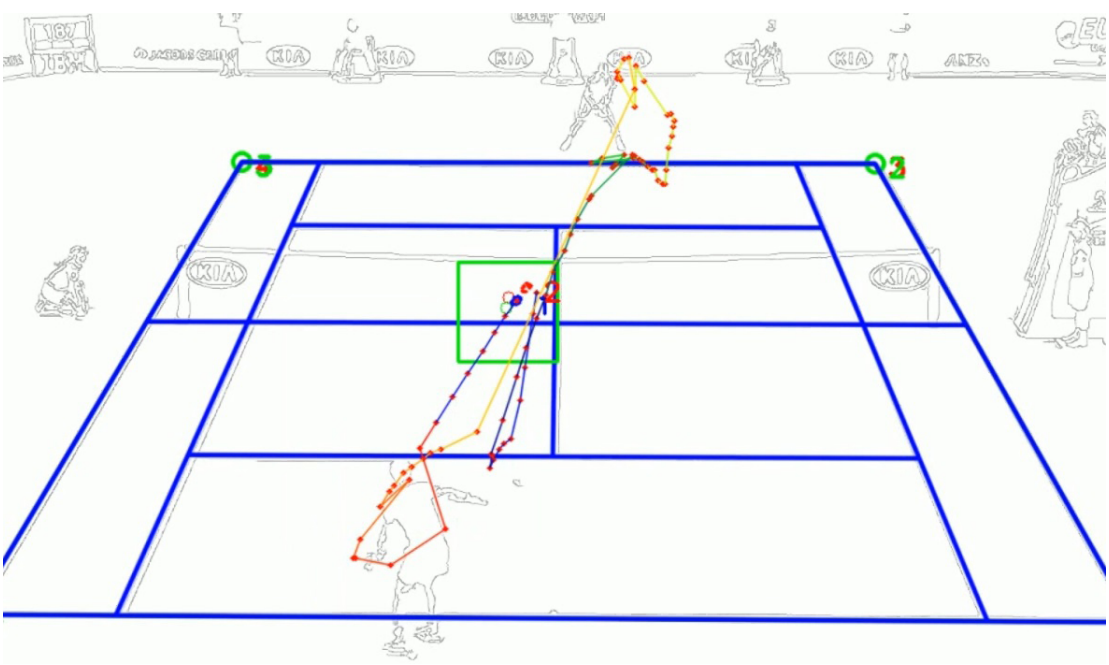


Abbildung 5.16.: Ballerkennung: Ball wird nach Falschidentifikation wiedergefunden

Parameter	Werte der Parameter
Canny Edge Detector:	
minimaler Thresholdwert	39
maximaler Thresholdwert	200
Erkennung des Spielfeldes:	
kurze_Linien	1 (kurze Linien entfernen)
min_Linienlänge	480
max_Linienlänge	960
verbessere_Felderkennung	2 (anhand Spielfeldecken)
Ballerkennung - Suchbereich:	
Berechnungsart	linear
Skalierung_Breite	3.0
kleinster_Suchbereich	58
Wiederholungsfaktor	0.7
Verschiebung	anhand letzter Ballposition
Ballerkennung - Größe/Distanz:	
Ballhistorie	1 (anhand letzten beiden Ballgrößen)
Distanz_Ausgangspunkt	anhand zuletzt erkannter Ballposition
Distanzkorrekturfaktor	10
Ballerkennung - Identifikation:	
Berechnungsformel	exponentiell
Skalierung_Distanz	1.0
Skalierung_Differenz	0.003
gültiger_Bereich	35% der Bildbreite

Tabelle 5.1.: Übersicht der Parameterkonfiguration (für die weitere Auswertung der Ballerkennung)

Grundsätzlich soll mit der nachfolgenden Auswertung verdeutlicht werden, wie **effektiv die Ballerkennung** ist und welche Umstände sie beeinflussen. Hierbei sollen zu Beginn unterschiedliche **Spielfelduntergründe** betrachtet werden, bei denen, aufgrund eines niedrigeren Kontrastes von gelbem Ball zu der Untergrundfarbe und -beschaffenheit, Auswirkungen auf die Spielfeld- und Ballerkennung erwartet werden. Weiterhin kommt es bei Fernsehaufnahmen von Tennisspielen vor, dass sich die **Kameraführung** in Form von seitlichen Bewegungen der Kamera oder von heran- oder wegzoomen verändert. Diese dynamischen Videos waren in dieser Arbeit nicht Hauptaugenmerk bei der Implementierung der Ballerkennung, sollen aber trotzdem einer statischen Kameraführung gegenübergestellt werden, sodass ersichtlich wird in wie weit die Ballerkennung auch bei dynamische Sequenzen funktioniert. Das dritte Szenario untersucht die **Qualität des Videos**. Bei guter Videoqualität ist eine bessere Ballerkennung zu erwarten. In diesen Szenarien sollen die Ergebnisse durch Angabe der Bezugsgrößen, Precision und Recall, bewertet werden, wodurch eine bessere Aussage über die Effektivität der Ballerkennung getroffen werden kann. Zuletzt soll dargestellt werden, bei wie vielen **Ballwechseln** die Flugkurve des Balles vollständig erkannt werden kann. Da hier viele Videoausschnitte, die in Summe bis zu zehn Minuten lang sind, betrachtet werden, ist der Aufwand einer Bildweisen Überprüfung, was für die Bestimmung der Precision und des Recalls notwendig wäre, zu groß. Aus diesem Grund soll angegeben werden, wie viel Prozent der untersuchten Ballwechsel richtig erkannt wurden.

Ebenfalls soll kurz auf die **Ausführungsgeschwindigkeit** eingegangen werden, die für die Ballerkennung benötigt wird. Die Ballerkennung wurde implementiert ohne das Hauptaugenmerkt auf ihre schnelle Durchführung zu legen. Daher werden auch keine Berechnungen durch die Grafikkarte oder eine parallele Ausführung einzelner Bildframes zur Beschleunigung der Bildverarbeitung realisiert. Die nachfolgenden Szenarien wurden auf einem PC mit 64-bit Windows 7 Betriebssystem, einem Intel i5-2500k Quad-Core Prozessor mit je $3.30GHz$ und $8GB$ DDR3-1334 DRAM durchgeführt.

5.2.1. Analyse der Spielfelder

In diesem Abschnitt soll der Einfluss des Spielfelduntergrundes und damit der Kontrast zwischen Hintergrund und gelbem Ball untersucht werden. Es werden hier drei Spielfelduntergründe untersucht. Ein blaues Spielfeld, bei dem davon auszugehen ist, dass der gelbe Ball aufgrund eines hohen Kontrastes zum Bodenbelag, gut erkannt werden kann. Ein Sandplatz mit hellerem, rötlichen Sandboden und ein grüner Rasenplatz, die beide eventuell durch Laufwege abgenutzt werden und somit nicht immer einen glatten Hintergrund darstellen.

Zuerst sollen verschiedene Ballwechsel bei einem blauen Spielfeld betrachtet werden. Dafür wurde ein Video mit drei kurzen Ballwechseln untersucht, das insgesamt eine Dauer von 23 Sekunden besitzt. Die einzelnen Ballwechsel wurden dabei zu einem Video zusammengeschnitten, sodass zu lange Pausen dazwischen vermieden werden. Es wurde allerdings darauf geachtet, dass dennoch ausreichend Zeit zwischen den Ballwechseln vergeht, da dieses Video eine echte Fernsehaufnahme nachahmen soll. Fernsehaufnahmen eines kompletten Tennisspiels weisen insgesamt eine deutlich längere Pausen- als echte Spielzeit auf. Um unnötig lange Ausführungsdauern bei der Ballerken-

nung zu vermeiden, wurden bei den hier betrachteten Szenarien längere Pausen entfernt. In dem analysierten Video für das blaue Spielfeld dauert der erste Ballwechsel vier Sekunden, der zweite sechs und der Dritte ebenfalls vier Sekunden. Die restlichen neun Sekunden stellen Pausen vor beziehungsweise nach dem Ballwechsel, sowie Nahaufnahmen, in denen der Ball nicht erkannt werden kann, dar.

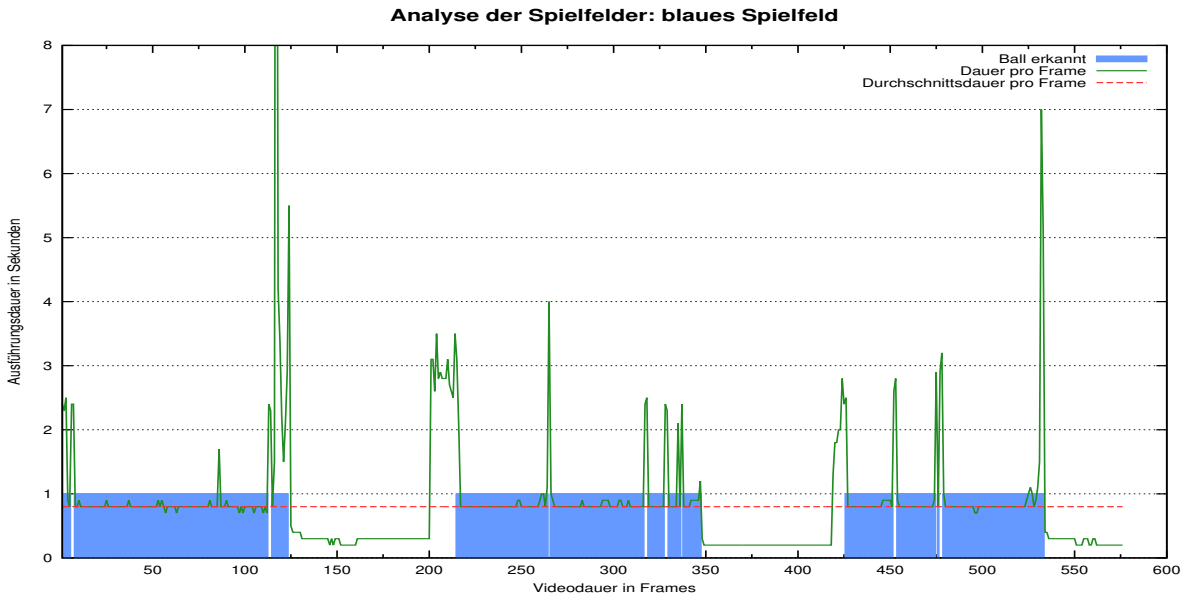


Abbildung 5.17.: Ballerkennung - Analyse der Spielfelder: blaues Spielfeld (Darstellung des Ballwechsels unter Angabe der Ausführungsdauer)

In Diagramm 5.17 ist ein zeitlicher Ablauf der Ballerkennung bei einem Video mit **blauen Spielfeld** dargestellt. Auf der X-Achse sind die Bildframes des Videos angegeben. Insgesamt besitzt das Video 577 Frames. Das Video wurde mit 25 FPS aufgenommen, sodass 25 Frames einer Sekunde entsprechen. Auf der Y-Achse ist die Ausführungsdauer der Ballerkennung in Sekunden, die pro Bildframe benötigt wird, angegeben. Die grüne Linie stellt die aktuelle Ausführungsdauer des Bildframes dar. Auf diesen Verlauf soll in Kürze genauer eingegangen werden. Die blauen Balken zeigen auf, ob im dargestellten Bildframe der Ball erkannt wurde. So zeigt sich, dass der Ball bereits ab Frame 1 bis kurz vor 125 fast durchgehend richtig identifiziert wurde. Dies stellt den ersten Ballwechsel des Videos dar. Bei Bild 6 und 7 zeigt sich eine weiße Lücke, die angibt, dass der Ball in diesen Frames nicht erkannt wurde. Zu diesem Zeitpunkt wird der Ball ausführlicher, nämlich bis zu dreimal, in einem zunehmend größer werdenden Suchbereich, gesucht, was sich im grünen Zeitverlauf zeigt. Dabei werden für die Untersuchung von Bildframe 6 und 7 jeweils 2.4 Sekunden benötigt. Auch in den ersten Bildframes ist der Suchbereich noch größer, da hierbei die Aufschlagsituation im gesamten Bild betrachtet werden muss. Erst nachdem der Ball mehrmals am Stück detektiert wurde, pendelt sich die Ausführungsdauer pro Frame bei 0.8 Sekunden ein, sofern der Ball nicht verloren geht. In Frame 86 kann der Ball nicht beim ersten Versuch erkannt werden, was sich in einem kleinen Ausschlag bei der Dauer des Frames auf 1.7

Sekunden zeigt.

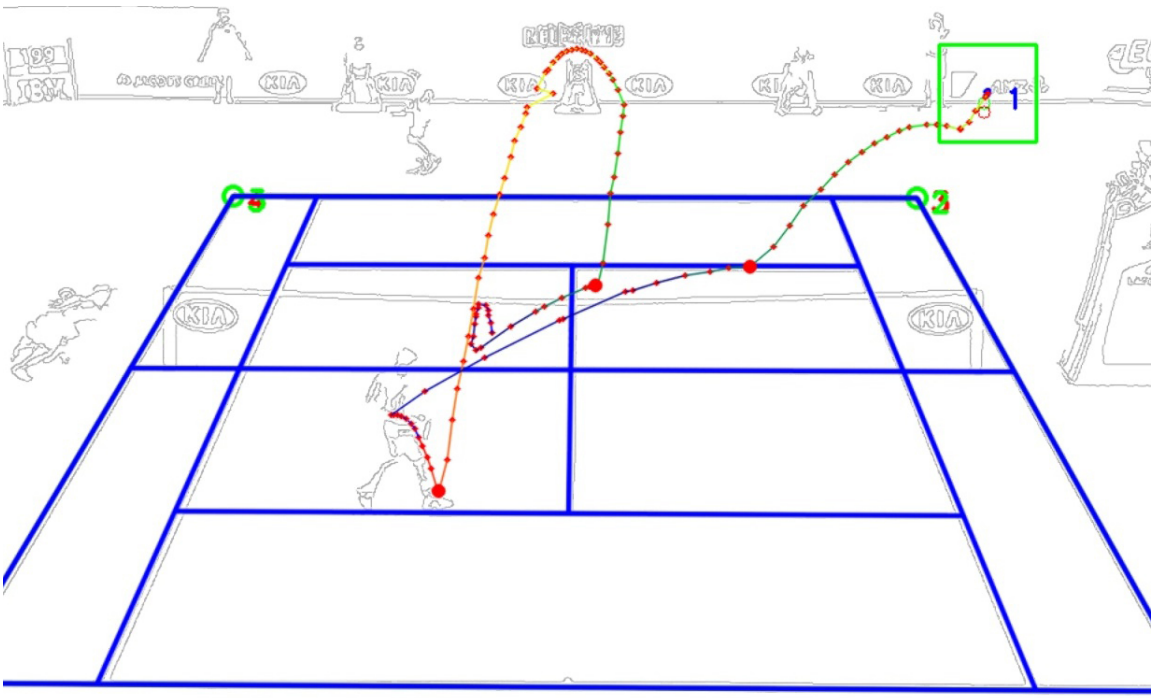


Abbildung 5.18.: Darstellung des ersten Ballwechsels bei blauen Spielfeld

Ab Bildframe 113 ist der Ballwechsel zu Ende und es erfolgt ein Bildwechsel. In Abbildung 5.18 wird die Ballerkennung des ersten Ballwechsels dargestellt. Da der Ballwechsel beendet ist und fließend zu einer Nahaufnahme gewechselt wird, werden Bilder kurzzeitig überlagert (Bilder des Spielfelds mit Bildern der Nahaufnahme), sodass die Ballerkennung zu diesem Zeitpunkt fälschlicherweise andere Bildpunkte als Ball annimmt. Dies macht der große Ausschlag der Ausführungsdauer, ab Bildframe 114 bis Frame 124, auf über $8s$ sichtbar, der aus Übersichtsgründen nicht komplett in diesem Diagramm abgebildet wurde. Diese zehn Bildframes, die hier falsch erkannt werden, stellen die zehn Frames dar, in der das Spielfeld weiterhin als gültig angesehen wird, auch wenn es nicht mehr im Bild erkannt werden kann. Solange ein gültiges Spielfeld existiert, soll ein Ball erkannt werden. Es dauert also zehn Frames bis festgestellt wird, dass die Kamera nicht mehr auf das Spielfeld gerichtet ist, sondern eine Nahaufnahme zeigt und der Ballwechsel somit zu Ende ist. Bis Bildframe 200 zeigt sich eine niedrige Ausführungsdauer von 0.3 Sekunden pro Frame, da hier, aufgrund der Nahaufnahme, kein Spielfeld erkannt wird und somit keine Ballerkennung durchgeführt werden muss.

Ab Bildframe 201 beginnt eine neue Aufschlagsituation. Es wird versucht das Spielfeld, sowie die Aufschlagsituation zu erkennen. Dafür wird ungefähr 2.8 Sekunden pro Frame benötigt. Ab Frame 215 wird der Ball erkannt und kann für den nächsten Ballwechsel nachverfolgt werden. Dieser Ballwechsel dauert bis zu Bildframe 337. Auch danach wird der Ball fälschlicherweise

in zehn Bildframes, aufgrund des fließenden Überganges des Kamerawechsels, festgestellt. Die Ausschläge der Ausführungsdauer zuvor bei Frame 317 und 328 stellen die Situation dar, bei der der Ball ins Netz geschlagen wird und dabei, aufgrund langsamerer Bewegungen, auf einer Stelle nicht immer detektiert werden kann.

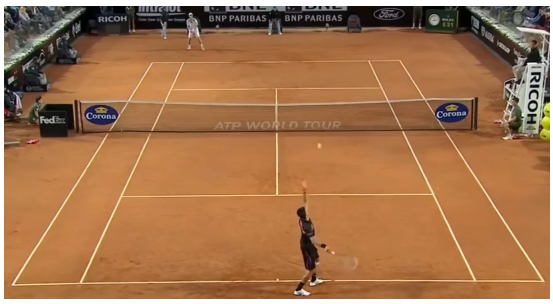
Der dritte Ballwechsel in diesem Video beginnt ab Bild 419. Der Ball wird in Bild 426 erstmals erkannt und wird wie bei den vorherigen Ballwechseln nachverfolgt. Die Ausschläge des grünen Verlaufes bei Frame 452 und bei Frame 475 stellen Situationen dar, in denen sich der Ball vor dem Spieler befindet und daher mit ihm verschmilzt. Aus diesem Grund kann er nicht erkannt werden. Alles in allem werden durchschnittlich pro Bildframe 0.8 Sekunden benötigt, um den Ball zu erkennen, was durch die gestrichelte rote Linie gekennzeichnet ist. Die automatische Ballerkennung ergab, dass der Ball in 81% der Fälle erkannt werden konnte. Dies soll anschließend anhand der Bezugsgrößen, Precision und Recall, überprüft werden.

Durch manuelle Überprüfung des Videos soll nun gezeigt werden, wie oft der Ball insgesamt hätte erkannt werden können und wie oft die Ballerkennung den Ball richtig identifiziert hat. Dafür wurde das Video Bild für Bild betrachtet, um die Bezugsgrößen, Precision und Recall, ermitteln zu können. Es ergab sich eine Precision von 88.25% und ein Recall von 80.63%. Die Precision berechnete sich aus dem Verhältnis von 308 richtig identifizierten Ballpositionen zu insgesamt 349 erkannten Ballpositionen. Die Ballerkennung hatte also 41 mal ein Objekt als Ball erkannt, das in Wirklichkeit keinen Ball dargestellt hatte. Dies ist zum einen auf den fließenden Wechsel der Kameras nach Ende des Ballwechsels hin zu einer Nahaufnahme, zum anderen aber auch auf eine falsche Erkennung des Balles in der Nähe der Spieler zurückzuführen.

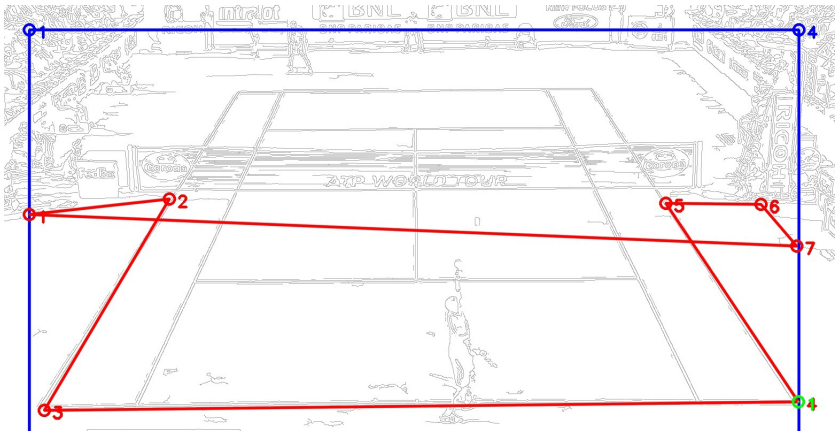
Der Recall ergibt sich aus dem Verhältnis der richtig erkannten Ballpositionen und der Anzahl der Frames in denen der Ball potentiell hätte erkannt werden können. Richtig identifiziert wurden 308 Frames. Der Ball hätte jedoch in 382 Frames erkannt werden können. Die fehlenden Frames sind entweder dadurch zu erklären, dass zuerst die Aufschlagsituation klar identifiziert werden muss, bevor eindeutig auf den Ball geschlossen werden kann. Dort ist der Ball bereits auf dem Bild zu erkennen, aber kann noch nicht sicher als Ball eingestuft werden. Zum anderen ist der Ball nicht immer während der Ballwechsel zu erkennen. Bei nicht vorhandener Bewegung im Bildframe kann der Ball beispielsweise verloren gehen, wenn beispielsweise die Flugkurve des Balles senkrecht in die Bildebene hineinführt. Auch in Spielernähe kann der Ball nicht immer erkannt werden, da er in diesem Fall mit dem Spieler verschmelzen kann.

Allerdings ist es für den grundsätzlichen Verlauf der Flugkurve nicht notwendig den Ball in jedem Bildframe zu erkennen, sodass eine Precision von 88.25% und ein Recall von 80.63% ein gutes Ergebnis für die Ballerkennung bei einem blauen Spielfelduntergrund darstellt.

Bei der Untersuchung des **Sandplatzes** stellte sich heraus, dass der Ball dort überhaupt nicht erkannt werden konnte. Allerdings kann keine Aussage über die Ballerkennung an sich und den Kontrast des Balles zum roten Sandboden getroffen werden, da bereits die Erkennung des Spielfeldes nicht möglich war. In Bild 5.19(a) ist eine Aufschlagszene des untersuchten Sandplatzes dargestellt. In dieser Arbeit wird bei der Spielfelderkennung die größte Kontur des Bildes darauf untersucht, ob es sich dabei um ein Spielfeld handelt. In diesem Video wird allerdings nicht das



(a) Sandplatz Farbbild



(b) Sandplatz Konturen

Abbildung 5.19.: Ballerkennung - Analyse der Spielfelder: Sandplatz (Problem bei der Spielfelderkennung)

gesamte Spielfeld als Kontur erfasst, sondern nur ein Teil. Dies ist in Abbildung 5.19(b) aufgezeigt. Weiterhin ist in dieser Abbildung zu erkennen, dass wesentlich mehr Kanten beispielsweise des Netzes, des Spielfeldbelags (bei dem vorderen Spieler) oder auch an den Spielfeldlinien zu entdecken sind. In der hier dargestellten Aufnahme ist die Spielfeldumgebung relativ klein, sodass neben dem Netz wenig Platz ist und deshalb die Balljungen und Schiedsrichter direkt daneben platziert sind. Diese Gründe können dafür sprechen, dass kein Spielfeld erkannt werden konnte.

Das gleiche Problem zeigte sich bei dem untersuchten Video mit **Rasenplatz**. Dort stellte sich ebenfalls dar, dass die Spielfelderkennung, die in dieser Arbeit Anwendung findet, Spielfelder nicht unter allen Umständen erkennen kann. In diesem Video ist das Problem, dass die Kamera zu nah auf das Spielfeld heranzoomt, sodass die Spielfeldecken in der Nähe des Bildrandes liegen und somit nicht in die Spielfeldanalyse mit einbezogen werden. Dies ist in Abbildung 5.20 zu sehen. Die linke untere Spielfeldecke befindet sich nicht mehr im blauen Bereich, indem nach Konturen für das Spielfeld gesucht wird. Da in diesem Video keine Spielfelderkennung möglich war, konnte auch keine Ballerkennung durchgeführt werden.

Zusammengefasst wird festgestellt, dass die in diesen Situationen unzureichende Spielfelderken-

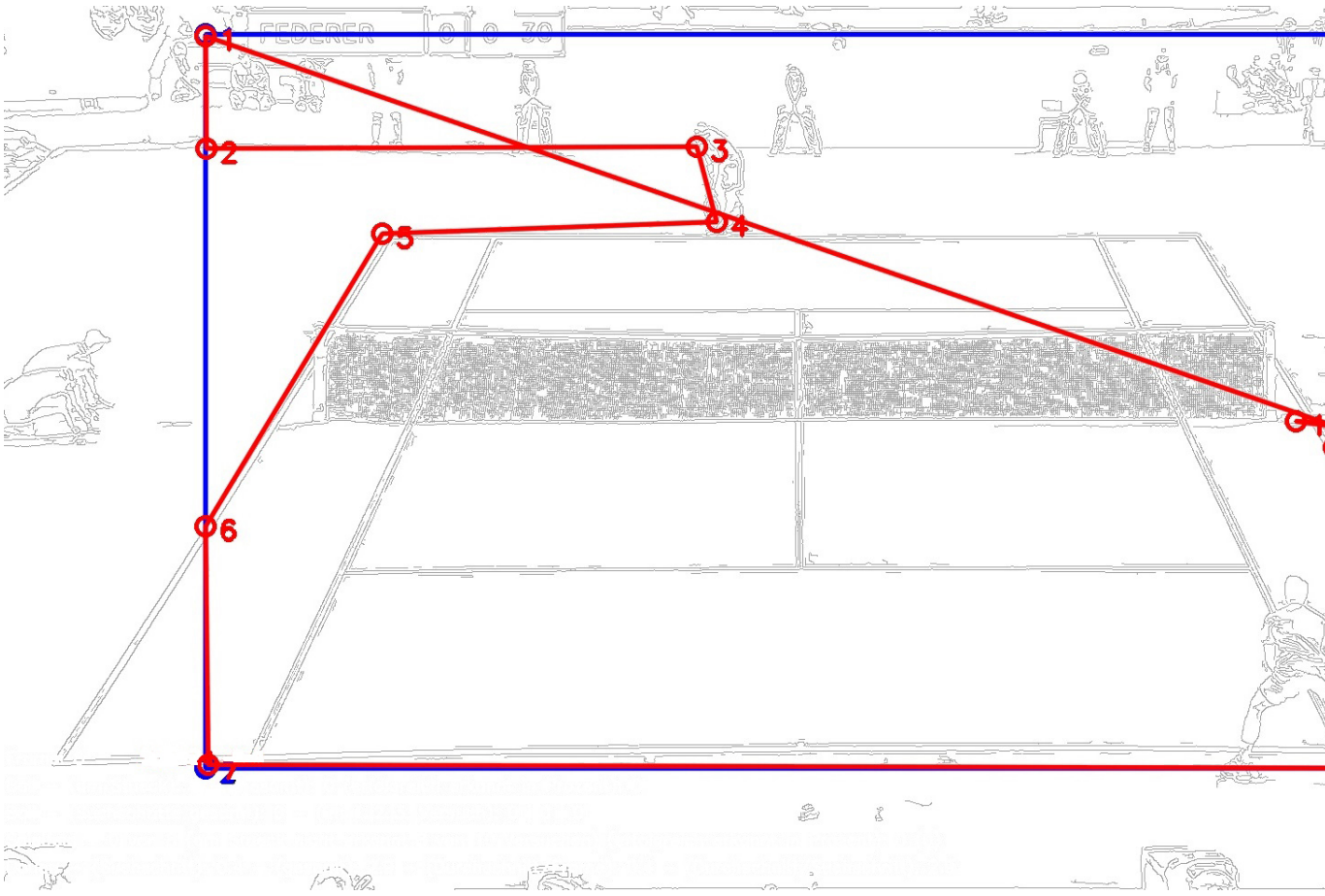


Abbildung 5.20.: Ballerkennung - Analyse der Spielfelder: Rasenplatz (Problem: Spielfeld ist sehr nah am Bildrand und wird daher nicht komplett analysiert)

nung einen Vergleich der Spielfeldbeläge verhindert. Bei einem Rasen- oder Sandplatz werden mehr Konturen festgestellt, da diese Spielfeldbeläge uneben sind und Abnutzungen durch die Laufwege aufweisen. Dadurch können die Spielfeldlinien nicht so gut wie bei einem ebenen, blauen Hartplatz erkannt werden. Ebenso zeigt sich, dass nicht jede Spielfeldumgebung gleich gut geeignet ist, was die Spielfelderkennung negativ beeinflusst. Die Ballerkennung bei blauem Spielfelduntergrund präsentierte sich allerdings mit einer Precision von 88.25% und einem Recall von 80.63% vielversprechend.

5.2.2. Analyse der Kameraführung

In diesem Abschnitt soll die Kameraführung untersucht werden. Es sollen statische und dynamische Fernsehaufnahmen betrachtet werden. Bei statischen Aufnahmen bleibt der Fokus und die Ausrichtung der Kamera gleich. Sie bewegt sich folglich nicht und es wird nicht heran- oder

weggezoomt. Dynamische Fernsehaufnahmen dagegen zeichnen sich durch Bewegungen der Kamera aus, um den Ball jederzeit möglichst mittig im Bild zu fokussieren. Außerdem kann das Bild näher an das Spielfeld heran- oder weiter vom Spielfeld weggezoomt werden. Dabei soll der Unterschied und die Problematik in diesen Situationen beleuchtet werden.

Eine **statische Fernsehaufnahme** wurde bereits im vorherigen Kapitel 5.2.1 untersucht. Dort erreichte das Video mit dem blauen Spielfelduntergrund eine Ballerkennung von 81% und erzielte dabei eine Precision von 88.25%, sowie einen Recall von 80.63%. In diesen statischen Aufnahmen kann sich komplett auf die Ballerkennung der bewegten Objekte konzentriert werden. Diese sind mit großer Wahrscheinlichkeit die Spieler und der Ball. Es existieren daher wenig Störobjekte.

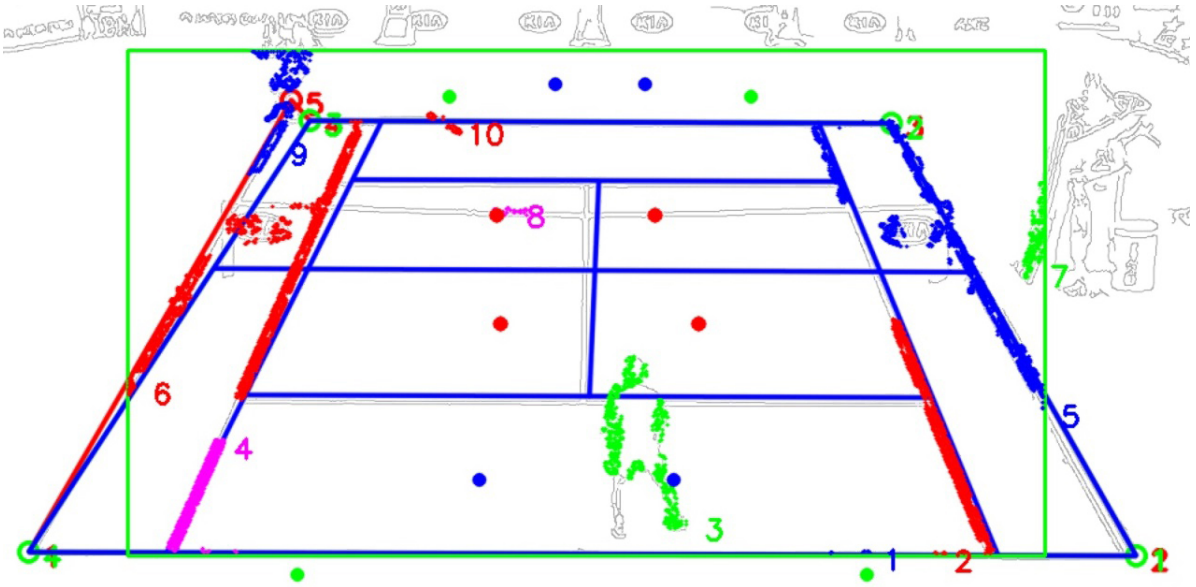


Abbildung 5.21.: Ballerkennung - Analyse der Kameraführung: dynamische Aufnahmen

Bei **dynamischen Fernsehaufnahmen** werden dagegen viele Objekte erkannt, da sich durch Veränderung der Kameraposition oder des Fokus auch feststehende Objekte in ihrer Größe oder Position verändern. Dies wird in Differenzbildern ebenfalls als Bewegung erkannt. In dieser Arbeit wurde kein Aufwand betrieben eine dynamische Erkennung des Balles zu realisieren. Daher werden fälschlicherweise zu viele Objekte festgestellt, die nun aufgrund der Bewegung der Kamera als bewegtes Objekt erkannt werden. In Abbildung 5.21 ist eine solche dynamische Situation dargestellt. Es werden neben den Spielern und dem Ball (rotes Cluster mit Nummer 10) weitere Cluster detektiert. Diese sind beispielsweise die Netzkante (Cluster 8) oder die Seitenlinien. Mit der Implementierung der Ballerkennung aus dieser Arbeit ist es dadurch nicht möglich, den Ball zuverlässig herauszufiltern. So wird der Ball in diesem 22 sekündigen Video nicht einmal richtig erkannt.

5.2.3. Analyse der Videoqualität

Die Videoqualität spielt eine große Rolle in der Bildverarbeitung und kann die Ballerkennung wesentlich beeinflussen. Dies soll in diesem Abschnitt genauer untersucht werden. Dazu wird ebenfalls das statische Video mit blauen Spielfelduntergrund, das in Kapitel 5.2.1 genauer beschrieben wurde, herangezogen. Dieses Video besitzt eine **gute Videoqualität** und erreicht damit bei der Ballerkennung eine Precision von 88.25% und einen Recall von 80.64%. Damit nur die Auswirkung der Videoqualität auf die Ballerkennung untersucht werden kann, wird dasselbe Video mit 23 Sekunden Laufzeit, in denen drei kurze Ballwechsel dargestellt werden, mit schlechterer Qualität untersucht. Dabei wurde das Orginalvideo mit niedrigerer Bitrate pro Sekunde neu gerendert und abgespeichert. Dadurch ergab sich eine Reduktion der Videogröße von $36.9MB$, um den Faktor 14.36, auf $2.57MB$.

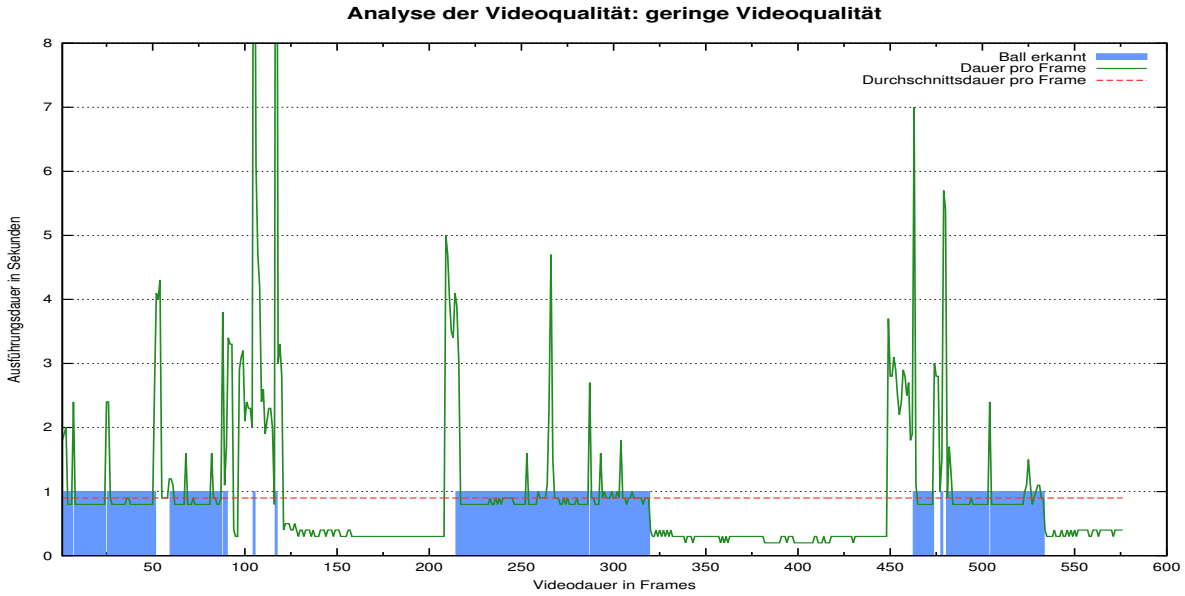


Abbildung 5.22.: Ballerkennung - Analyse der Videoqualität: niedrige Videoqualität (Darstellung des Ballwechsels unter Angabe der Ausführungsdauer)

In Abbildung 5.22 ist der Ballwechsel mit **niedrigerer Bildqualität** dargestellt. Hierbei sind, wie in Abbildung 5.17 des gleichen Videos mit besserer Bildqualität, ebenfalls drei Ballwechsel zu erkennen. Sie werden durch die blauen Balken ausgedrückt, die den erkannten Ball im jeweiligen Bildframe kennzeichnen. Im Unterschied zu dem Video mit besserer Bildqualität zeigt sich allerdings, dass die Anzahl der Ballerkennungen bei schlechterer Videoqualität geringer ist. Wie in Abbildung 5.22 zu sehen ist, geht der Ball beim ersten Ballwechsel bei Bildframe 90 verloren. Auch im zweiten Ballwechsel kann der Ball nicht bis zum tatsächlichen Ende des Ballwechsels identifiziert werden. Es kann also bereits jetzt erkannt werden, dass die Videoqualität starken Einfluss auf die Ballerkennung besitzt.

Bei schlechterer Videoqualität werden nur 46% der Ballpositionen festgestellt, gegenüber 81%

bei guter Bildqualität. Auch die Precision und der Recall sinkt deutlich bei Abnahme der Videoqualität. Es zeigt sich, wie zuvor im Diagramm beschrieben, dass zum einen der Ball seltener erkannt wird, zum anderen aber auch häufiger falsch erkannt wird. Er wird bei schlechterer Qualität nur 249 mal bei insgesamt 577 Frames detektiert. Von diesen 249 Ballerkennungen waren zudem 64 Ballpositionen falsch, sodass der Ball lediglich 185 mal richtig identifiziert wurde. Dadurch liegt die Precision bei schlechterer Videoqualität folglich bei 74.3% und der Recall bei 48.05%. Es zeigt sich also, dass der Ball aufgrund unzureichender Details der Kantenverläufe der Objekte schwerer vom Spieler zu unterscheiden ist, was vermehrt zu Fehlidentifikationen führt.

5.2.4. Analyse der erkannten Ballwechsel

Zum Abschluss soll die Ballerkennung auf ihre Effektivität hin untersucht werden. In dieser Arbeit geht es vor allem darum, die groben Positionen des Balles beim Schlagtreffpunkt des Spielers oder bei Bodenkontakt zu erfassen. Daher ist es nicht notwendig, den Ball in jedem Bildframe zu erkennen. Es genügt, wenn die grundsätzliche Flugkurve des Balles nachverfolgt und somit der Verlauf des Ballwechsels ersichtlich wird. Der Ball darf im Ballwechsel also nicht in zu vielen aufeinanderfolgenden Frames verloren gehen, damit keine Schläge ausgelassen werden. Sollte der Ball einmal für längere Zeit verloren gehen und später erst wiederentdeckt werden, so zählt der Ballwechsel als nicht erkannt. Dadurch kann der Verlauf des Ballwechsels nicht im Ganzen nachvollzogen werden, wodurch unter Umständen keine richtigen Schlüsse für eine spätere Taktikanalyse gezogen werden können.

Die **Effektivität der Ballerkennung** soll durch Angabe der **erkannten Ballwechsel** in Prozent dargestellt werden. Dafür werden zwei Videos betrachtet, die mehrere Ballwechsel enthalten. Es wurden zwei Videos gewählt, da jedes Video eine unterschiedliche Qualität und Spielsituation mit sich bringt. Dadurch kann eine allgemeinere Aussage über die Effektivität der Ballerkennung getroffen werden. Diese Videos wurden zwar zurechtgeschnitten, allerdings wurden nur längere Pausen zwischen den Ballwechseln entfernt, da ohne diese Maßnahme die Ballerkennung wesentlich länger dauern würde. Zwischen den einzelnen Ballwechseln sind nur kurze Pausen von $1-2$ Sekunden, sodass die Laufzeit der Videos fast der effektiven Laufzeit der Ballwechsel entspricht.

Das erste Video, das untersucht werden soll, hat eine Länge von 609 Sekunden. Es werden in circa 10 Minuten insgesamt 52 Ballwechsel analysiert, wovon 29 Stück nicht länger als sieben Sekunden andauern. 18 Ballwechsel dagegen sind länger als sieben Sekunden. Gerade bei den längeren Ballwechseln ist zu erwarten, dass sie schwieriger komplett zu erkennen sind, da mehr Situationen auftreten, in denen der Ball sich in der Nähe des Spielers befindet und so leichter verloren gehen kann. Weiterhin gibt es fünf Ballwechsel, bei denen das Spielfeld nicht aus der Vogelperspektive betrachtet wird und somit durch die Ballerkennung in dieser Arbeit nicht behandelt werden können. Das Video besitzt eine bessere Videoqualität gegenüber dem zweiten Video, das eine Länge von 530 Sekunden besitzt. Es enthält in knapp 9 Minuten Laufzeit 42 Ballwechsel, wovon über die Hälfte lange Ballwechsel darstellen. Hier stehen insgesamt 23 lange Ballwechsel 19 kurzen gegenüber. Durch diese beiden Videos mit unterschiedlicher Videoqualität soll eine grundlegende Aussage über die Effektivität der Ballerkennung getroffen werden, indem

insgesamt 94 Ballwechsel mit unterschiedlicher Videoqualität und Spielsituationen betrachtet werden.

Im **ersten Video mit guter Bildqualität** konnten insgesamt 28 der 52 Ballwechsel erkannt werden. 24 Ballwechsel konnten folglich nicht erkannt werden, von denen allerdings 5 Ballwechsel aus einer Nahaufnahme hinter dem Spielfeld aufgenommen wurden. Bei diesen Ballwechseln war das Spielfeld nicht komplett abgebildet und konnte daher nicht gefunden werden. Dadurch war bereits im Voraus klar, dass der Ball bei einer solchen Perspektive nicht detektiert werden kann, sodass diese aus der untersuchten Menge der Ballwechsel ausgeschlossen werden kann. In dieser Arbeit sollen lediglich Fernsehaufnahmen von Tennisspielen aus der Vogelperspektive, die aus einer Position hinter dem Spielfeld aufgenommen wurden, betrachtet werden. Dadurch zeigt sich für Video eins eine erfolgreiche Erkennung der Ballwechsel in 59.57% aller Fälle. Weiterhin kann festgestellt werden, dass von insgesamt 29 Ballwechseln, die kürzer als 7 Sekunden sind, 19 Stück erkannt wurden. Dies stellt eine Erfolgsquote bei der Erkennung kurzer Ballwechsel von 65.52% dar. Von den 18 langen Ballwechseln konnten nur 9 erkannt werden, da hier die Wahrscheinlichkeit höher ist, dass sich die Ballerkennung bei Annäherung von Ball und Spieler verfängt und der Ball dadurch verloren geht. Bei den insgesamt 24 nicht erkannten Ballwechseln zeigt sich, dass 5 Ballwechsel aufgrund einer ungünstigen Kameraperspektive nicht erkannt werden konnten. Weiterhin wurde teilweise das Spielfelder nicht gefunden, da beispielsweise vom Sender in die Fernsehübertragung des Tennisspieles ein Werbeblock eingeblendet wurde, sodass die Kontur des Feldes zu diesem Zeitpunkt überdeckt war und daher nicht erkannt werden konnte. Auch zeigten sich vereinzelt Störungen in der Ballerkennung durch kurzzeitige dynamische Kameraführung oder aber auch Störungen durch Bewegungen von Linienrichtern, Verwechslungen des Balles mit dem Spieler oder eines Aufschlages mit hoher Geschwindigkeit.

In Video zwei konnte diese Erkennungsquoten nicht erreicht werden. Dort wurden von 42 Ballwechseln lediglich 11 richtig erkannt. Dies bedeutet, dass nur 26.19% der Ballwechsel vollständig identifiziert wurden. Zudem zeigte sich, dass vorwiegend kurze Ballwechsel erkannt wurden. 8 der insgesamt 11 erkannten Ballwechsel waren kürzer als 7 Sekunden. Insgesamt wurden 19 kurze Ballwechsel untersucht, was immerhin einer Quote von 42.11% entspricht. Von den 23 Ballwechseln, die länger als 7 Sekunden andauern, wurden lediglich 3 vollständig erkannt. Bei diesem **Video mit schlechter Bildqualität** zeigte sich, ähnlich wie im vorherigen Video, dass ebenso Bewegungen der Linienrichter, schnelle Aufschläge, Probleme bei der Spielfelderkennung oder Verwechslungen des Balles mit dem Spieler die Ballerkennung stören. Es ließ sich aber feststellen, dass der Ball vermehrt mit Teilen des Spielers verwechselt wurde. Auch ging der Ball häufiger scheinbar grundlos verloren. Häufig zeigte sich, dass im oberen Spielfeldbereich der Ball leichter verloren geht, da hier der Ball durch die weitere Entfernung zur Kameraposition, kleiner dargestellt wird, was sich bei schlechterer Bildqualität verstärkt auswirkt.

Da beide Videos unterschiedliche Videoqualität und Spielfeldsituationen darstellen, kann durch sie eine **allgemeine Aussage über die Ballerkennung** getroffen werden. Von insgesamt 94 Ballwechseln aus beiden Videos wurden 39 erkannt. Von den 55 nicht erkannten Ballwechseln war für 5 bereits im Vorhinein klar, dass aufgrund einer in dieser Arbeit nicht betrachteter Kameraperspektive, diese nicht identifiziert werden können. Es konnten daher insgesamt 43.82%

	Ballwechsel	erkannt	kurz (erkannt)	lang (erkannt)	falsch
Video 1	52	28 (59.57%)	29 (19, 65.52%)	18 (9, 50%)	5
Video 2	42	11 (26.19%)	19 (8, 42.11%)	23 (3, 13.04%)	0
Insgesamt	94	39 (43.82%)	48 (27, 56.25%)	51 (12, 23.53%)	5

Tabelle 5.2.: Ergebnisse der Ballwechselanalyse

aller Ballwechsel komplett erkannt werden. Die Ergebnisse der Ballwechselanalyse wurden in der Tabelle 5.2 zusammengefasst.

5.2.5. Analyse der Ausführungsgeschwindigkeit

Bei Betrachtung der Ausführungsgeschwindigkeit der Ballerkennung zeigte sich für das eben betrachtete Video eins bei der Ballwechselanalyse eine durchschnittliche Dauer pro Bildframe von 1.2 Sekunden und für Video zwei eine Dauer von 1.3 Sekunden. Insgesamt wurde also für Video eins, das insgesamt 15096 Bildframes besitzt, ungefähr 5 Stunden für die Ballerkennung benötigt. Video zwei benötigte mit 15851 Frames sogar fast 6 Stunden.

Die durchschnittliche Ausführungsdauer pro Frame von über 1.2 Sekunden scheint allerdings der stark zugeschnittenen Videos geschuldet zu sein. In den Videos eins und zwei aus Kapitel 5.2.4 wurden die Pausenzeiten zwischen den Ballwechseln soweit wie möglich auf ein Minimum reduziert, damit möglichst viele Ballwechsel in einer möglichst kurzen Zeit untersucht werden können. Da die Ballerkennung zu Beginn und während des Ballwechsels deutlich länger als in den Pausen zwischen den Ballwechseln dauert, liegt die Ausführungsdauer pro Frame für diese beiden Videos verhältnismäßig hoch. Das Video mit dem blauen Spielfeld aus Abschnitt A.1.1 besitzt eine durchschnittliche Ausführungsdauer von 0.8 Sekunden pro Frame. Der zeitliche Verlauf dafür ist in Abbildung 5.17 dargestellt. Darin ist zu erkennen, dass zwar für die Ballerkennung während eines Ballwechsels mehr Zeit benötigt wird, aber durchaus auch längere Pausen vorkommen, die die durchschnittliche Ausführungszeit senken. Dies ist aufgrund einer geringen Ausführungsdauer von 0.3 Sekunden pro Frame in den Pausen zu erklären. Daraus folgt, dass die durchschnittliche Ausführungsdauer eines Videos stark von dem Verhältnis zwischen der effektiven Spielzeit des Tennisspiels und der Länge der Pausen zwischen den einzelnen Ballwechseln abhängt. Bei einer Fernsehaufnahme eines Tennisspieles liegt, laut Kovacs et al. [7], der Anteil des laufenden Ballwechsels zu den Pausen dazwischen bei einem Verhältnis von $1:2$ bis $1:5$. Die Pausen wirken sich also bis zu 5-mal stärker auf die durchschnittliche Ausführungsdauer der Ballerkennung aus als die Ausführungsdauer während des Ballwechsels. Daher scheint für die Ballerkennung eine durchschnittliche Ausführungsdauer pro Bildframe von 0.8 Sekunden als durchaus realistisch. Tendenziell sollte diese noch geringer sein. Im Detail hängt sie auch davon ab, wie gut der Ball erkannt werden kann. Wird der Ball schnell detektiert, liegt die Ausführungsdauer pro Frame, laut Abbildung 5.17, bei ungefähr 0.8 Sekunden. Wird der Ball nicht direkt erkannt oder geht sogar verloren, wird wesentlich mehr Zeit für den Bildframe benötigt, da der Ball bis zu dreimal pro Bildframe gesucht wird. Auch am Anfang des Ballwechsels wird

für die Erkennung der Aufschlagsituation zwischen 2 und 3 Sekunden pro Frame benötigt. Erst nach Identifizierung des Balles pendelt sich die Ballerkennung pro Bildframe bei 0.8 Sekunden ein.

6. Zusammenfassung und Diskussion

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst und interpretiert. Weiterhin sollen die Aspekte der Arbeit, die vielversprechend sind, aber auch diejenigen die Probleme verursachen und für die Spielfeld- oder Ballerkennung ungeeignet sind, beleuchtet werden. Dafür werden Vorschläge gemacht, die diese Probleme beheben und dadurch die Ballerkennung weiter verbessern könnten.

6.1. Zusammenfassung der Arbeit

Diese Arbeit beschäftigte sich mit der **automatischen Ballerkennung bei Fernsehaufnahmen im Tennis**, da dies für viele Sportler und Trainer eine große Hilfe sein kann, um ihre beste Leistung zeigen und so ihre Ergebnisse verbessern zu können. Der Ansporn nach Perfektionismus und die Leistungsorientierung der Spieler und Trainer ist durch kontinuierlich steigende Einnahmemöglichkeiten im Sport zu erklären. Eine automatisierte Ballerkennung würde eine vereinfachte Möglichkeit bieten eine Taktikanalyse von einem potenziellen Gegner zu erstellen, um dadurch perfekt auf diesen vorbereitet zu sein. Folglich war es das Ziel dieser Arbeit eine automatisierte Ballerkennung zu implementieren, die es ermöglichte den Ball zum Schlagzeitpunkt oder bei Bodenkontakt zu lokalisieren. Diese Positionsbestimmung muss nicht auf den Zentimeter genau erfolgen, da für eine grundlegende Taktikanalyse, die beispielsweise die Schlagabfolgen der Spieler analysiert, auch eine ungefähre Position des Balles ausreicht.

Eine große **Herausforderung** stellte dabei die Vielzahl an Parametern bei Fernsehaufnahmen dar, die die Ballerkennung stark beeinflusst. Zu diesen Parametern zählt zum einen die Qualität der Fernsehaufnahmen, die üblicherweise schlechter ist als bei der Analyse eines Tennisspiels, bei der ein eigenständiger Aufbau mit exakt platzierten Kameras existiert. Hinzu kommt, dass sich auch die Kameraposition in den verschiedenen Tennisstadien unterscheidet, sodass die Blickwinkel der Kameraperspektiven voneinander abweichen können. Zum anderen wechselt je nach Stadion die Spielfeldumgebung und der -untergrund. Beeinflussend auf die Ballerkennung wirkt sich dabei auch der vorhandene Platz zwischen Netz und Schiedsrichterstuhl beziehungsweise zwischen Spielfeldbegrenzung und Zuschauerränge aus. Weiterhin können sich je nach Tageszeit und Wetter die Lichtverhältnisse durch Sonne und damit verbundene Schatten verändern.

Aus diesen beeinflussenden Parametern wurden einige herausgesucht, deren Einfluss sich auf die spätere Auswertung gravierend auswirken könnte und wurden genauer betrachtet. Dazu gehören die unterschiedlichen **Bodenbeläge**, die **Kameraführung** und die **Videoqualität**. Mittels dieser Hauptszenarien sollte eine Aussage über die Anwendbarkeit der hier verwendeten

Spielfeld- und Ballerkennung getroffen werden. Dafür wurden in diesen Szenarien die Bezugsgrößen Precision und Recall ermittelt, die eine Aussage über die **Effektivität der Ballerkennung** ermöglichen.

Bevor diese grundlegenden Szenarien genauer evaluiert werden kann, war es notwendig die Effektivität der Ballerkennung zu optimieren. Dabei spielten bereits die Auflösungen der Fernsehaufnahmen eine wichtige Rolle. In dieser Arbeit wurden Fernsehaufnahmen mit einer Auflösung von $1920x1080$ betrachtet. Gegebenenfalls wurden Videos in diese Auflösung konvertiert. Die Auflösung der Videos spielt vor allem bei der Wahl der Parameterkonfiguration eine große Rolle, die wiederum die Spielfeld- und Ballerkennung beeinflusst. Diese wurde für die gegebene Auflösung anhand drei verschiedener Videos optimiert, um ein möglichst aussagekräftiges Ergebnis zu erhalten. Durch diese **Optimierung der Parameter** konnte die Anzahl der Ballerkennungen im ersten Video von 106 auf 107, im zweiten von 417 auf 486 und im dritten von 112 auf 123 verbessert werden. Aus dieser Analyse ergab sich die Parameterkonfiguration, die in Tabelle 5.1 im Kapitel 5.2 Evaluierung dargestellt ist. Diese wurde anschließend für die weitere Untersuchung der Hauptszenarien verwendet.

Das erste Hauptszenario stellt die **Untersuchung des Spielfeldbelags** dar. Dort zeichnete sich die Ballerkennung bei einem blauen Hartplatz mit einer Precision von 88.25% und einem Recall von 80.63% aus. Es zeigte sich folglich, dass 88.25% aller Ballerkennungen richtig waren und insgesamt 80.63% aller Ballpositionen, die im Video vorgekommen sind, erkannt wurden. Dies zeigt, dass die Ballerkennung mit einem guten Kontrast zwischen gelben Ball und blauen Spielfelduntergrund, sehr effektiv ist. Bei dem Vergleich von Aufnahmen mit verschiedenen Untergründen konnten keine Ergebnisse für die Ballerkennung auf Rasen- und Sandplatz erzielt werden. Es wurde festgestellt, dass keine Aussage über die Ballerkennung bei anderen Spielfelduntergründen möglich war, da unter diesen Umständen bereits das Spielfeld nicht erkannt wurde, sodass überhaupt keine Ballerkennung stattfinden konnte. Die Schwierigkeiten bei der Spielfelderkennung könnten gegebenenfalls daraus resultieren, dass durch einen unebenen und abgenutzten Spielfelduntergrund, wie es bei Rasen- und Sandplätzen üblich ist, mehr Konturen entdeckt werden, was zu einer schlechteren Erkennung der Spielfeldlinien führt. Ein weiterer Grund könnte die veränderte Spielfeldumgebung gewesen sein. Gespielt wurde, im Falle eines anderen Bodenbelags, auch in einem anderen Stadion, bei der das Spielfeld in den Fernsehaufnahmen unter Umständen stärker fokussiert wurde und dadurch der Abstand zwischen Spielfeldecke und Bildrand geringer war. Auch die Distanz zwischen Netz und Balljungen sowie Schiedsrichterstuhl variierten, sodass die hier implementierte Spielfelderkennung fehlgeschlagen ist.

Die **Kameraführung** wirkt sich ebenfalls stark auf die Ballerkennung aus. Hier wurden statische Fernsehaufnahmen, bei denen sich die Kameraposition und der -fokus nicht verändern, und dynamische Fernsehaufnahmen, bei denen sich die Kamera bewegt und auch näher heran oder weiter weg gezoomt wird, miteinander verglichen. Das Video mit blauem Spielfeld stellte eine statische Videoaufnahme dar und konnte, wie bereits oben beschrieben, mit einer Precision von 88.25% und einem Recall von 80.63% gute Ergebnisse erzielen. Bei dynamischen Fernsehaufnahmen zeigte sich allerdings, dass ebenfalls keine Ballerkennung möglich war, da durch die

dynamische Kameraführung auch feststehende Objekte, wie das Netz oder die Spielfeldlinien, als Bewegung erkannt wurden. Dadurch wurden sehr viele Störobjekte erkannt, wodurch keine eindeutige Identifikation des Balles möglich war.

Als nächstes soll auf die **Videoqualität** eingegangen werden. Um einen aussagekräftigen Vergleich zu ermöglichen wurde das Video mit blauem Spielfeldbelag mit niedrigerer Bitrate neu gerendert, sodass die Dateigröße von $36.9MB$ um den Faktor 14.36 auf $2.57MB$ reduziert wurde. Dadurch konnte bei dem gleichen Video mit schlechterer Bildqualität nur eine Precision von 74.3% und ein Recall von 48.05% erzielt werden. Es ist zu erkennen, dass sowohl der Ball seltener erkannt (48.05% gegenüber 80.63%) als auch häufiger falsch identifiziert wurde (74.3% gegenüber 88.25%). Dies kann auf weniger Details in den Kantenverläufen zurückgeführt werden, wodurch einzelne Objekte schwerer voneinander zu unterscheiden sind. Dadurch treten vor allem Probleme auf, wenn der Ball dem Spieler näher kommt, weshalb der Ball hier häufiger falsch identifiziert wurde. Auch zeigte sich, dass besonders im oberen Bildbereich der Ball häufiger nicht erkannt werden konnte. In diesem Bereich ist der Ball weiter von der Kamera entfernt und wird daher kleiner dargestellt als im unteren Bildbereich. Aus diesem Grund geht der Ball dort öfters verloren.

Ein weiteres Szenario soll die grundsätzliche **Effektivität der Ballerkennung** aufzeigen. Dabei sollte untersucht werden, wie gut die Ballerkennung verschiedene Ballwechsel erkennen kann. Die Ballwechsel, sowie insbesondere deren Flugkurve und Ablauf der Schlagabfolgen sind für eine Taktikanalyse besonders relevant. Für eine komplette Erkennung eines Ballwechsels ist es notwendig zum einen keinen Schlag auszulassen und zum anderen die Position des Balles bei Schlagtreffpunkt und Bodenkontakt zumindest grob zu erkennen. Hier wurden zwei Videos mit unterschiedlicher Bildqualität betrachtet, um eine allgemeinere Aussagekraft zu erhalten. Aus diesen wurden einige Ballwechsel herausgenommen und die Ballerkennung darauf angewandt.

In Video eins konnten bei **guter Bildqualität des Videos** von den 52 untersuchten Ballwechseln, insgesamt 59.57% vollständig registriert werden. Dabei konnten vor allem die kürzeren Ballwechsel, die nicht länger als 7 Sekunden andauerten, gut erkannt werden. Von insgesamt 28 kurzen Ballwechseln konnten 19 nachverfolgt werden. Im Falle der längeren Ballwechsel wurden immerhin noch 9 von 18 vollständig erkannt. Bei 5 Ballwechseln war bereits im Vorhinein klar, dass eine Erkennung nicht möglich sein würde, da hier der Ballwechsel nicht aus einer Vogelperspektive aufgenommen wurde, was in dieser Arbeit als Voraussetzung angenommen wurde.

Das zweite **Video mit schlechterer Videoqualität** enthielt 42 Ballwechsel von denen insgesamt 11 richtig erkannt wurden. Dies entspricht einer Identifizierung der Ballwechsel in 26.19% aller Fälle. Hier zeigten sich vor allem Probleme bei längeren Ballwechseln. Dort konnten lediglich 3 von 23 nachverfolgt werden. In beiden Videos mit insgesamt 94 Ballwechseln wurden davon **durchschnittlich** 43.82% vollständig erkannt.

6.2. Diskussion der Ergebnisse

In den nächsten Abschnitten soll genauer auf die Ergebnisse der unterschiedlichen Ansätze für die Spielfeld- und Ballerkennung eingegangen und anschließend diskutiert werden. Hier sollen zum einen vielversprechende Ergebnisse als auch auftretende Probleme erläutert werden. Weiterhin sollen für diese Probleme Lösungsvorschläge gegeben werden, um die Spielfeld- und Ballerkennung weiter zu verbessern. Die in dieser Arbeit verwendeten Ansätze werden Schritt für Schritt genannt. Am Ende soll ein Fazit gezogen werden, ob die Ballerkennung vielversprechend ist oder ob sie in dieser Form für eine Taktikanalyse nicht geeignet ist.

Ein Ansatz, der wesentlicher Bestandteil der Ballerkennung ist, ist die Bewegungen im Bild durch **Differenzbilder und anschließender Kantenerkennung mit Hilfe des Canny Edge Algorithmuses mit Kantennachverfolgung** zu erkennen. Dadurch kann die Objekterkennung anhand ihrer Konturen erfolgen, sodass die Objekte, wie das Spielfeld sowie die Spieler und der Ball, leicht und bereits in ihrer ursprünglichen Form identifiziert werden können, weshalb dieser Ansatz vielversprechend ist. Allerdings sollte beachtet werden, dass nicht jede Kontur erkannt werden kann. In Abhängigkeit der Intensität der Pixel können die Kanten und Konturen von Objekten besser oder schlechter bestimmt werden. Ist die Intensität der bewegten Bildpunkte zu gering, so wird keine Kontur erkannt, da sie unter einem Schwellwert liegt. Dies kann bei schlechter Bildqualität oder auch bei geringem Kontrast von Ball zu Hintergrund, beispielsweise durch Sonneneinstrahlung, vorkommen. Ein sehr schnell fliegender Ball kann nur verschwommen aufgenommen werden, sodass dieser nur mit einer geringen Pixelintensität registriert wird. Nur durch Kantennachverfolgung kann nicht garantiert werden, dass die bewegten Objekte in allen Fällen detektiert werden. Um diesem Umstand entgegenzuwirken, könnte das Differenzbild durch Thresholding (siehe Kapitel 2.1.3) optimiert werden. Hierbei würden Pixel mit geringer Intensität, die über dem vorgegebenen Threshold liegen, verstärkt, sodass auch die Kanten dieser Pixel erkannt werden können. Dadurch könnte die Zuverlässigkeit der Konturerkennung gesteigert werden.

Die **Spielfelderkennung** in dieser Arbeit zeigte in einigen Fällen sehr gute Ergebnisse. Sie fand anhand der **Konturen des Spielfeldes** statt, sodass je nach Spielfeldumgebung und Spielsituation das Spielfeld bereits als zusammenhängende Kontur erkannt wurde. Weiterhin konnte durch Entfernen von Störobjekten, wie der Spieler, die auf der Grundlinie stehen oder das Überstehen des Netzes an der Seitenlinie, die Spielfelderkennung verbessert werden. Es stellte sich allerdings heraus, dass es nicht immer eine Kontur gibt, die das komplette Spielfeld miteinschließt. Dies kann beispielsweise vorkommen, wenn relativ wenig Platz neben dem Spielfeld existiert, sodass sich die Balljungen und der Schiedsrichter sehr nah am Spielfeld befinden. Dadurch kann das Spielfeld lediglich auf einer Feldhälfte detektiert werden, da dessen Kontur in dieser Situation beispielsweise durch den Balljungen unterbrochen wird. Auch wurde festgestellt, dass die Spielfeldlinien bei einem unebenen oder abgenutzten Untergrund schlechter zu erkennen sind. Es kann damit also nicht garantiert werden, dass jedes Spielfeld erkannt wird, was jedoch Voraussetzung für eine weitere Ballerkennung ist.

Um in der Zukunft eine zuverlässigere Spielfelderkennung zu ermöglichen, ohne dass das Umfeld

des Spielfeldes eine große Rolle spielt, sollte eine Erkennung mittels **Hough Transformation** in Betracht gezogen werden. Damit können alle wichtigen Linien in einem Bild bestimmt werden. Dies wurde zu Beginn dieser Arbeit versucht umzusetzen, wurde allerdings verworfen, da die Spielfeldlinien nicht immer durchgängig erkannt wurden. Außerdem wurden die Spielfeldlinien nicht als eine Linie betrachtet, sondern bestanden aus vielen nebeneinanderliegende Linien, sodass diese zu einer Linie verbunden und die noch existierenden Lücken geschlossen werden müssten. Weiterhin müssten aus einer Vielzahl von Linien diejenigen identifiziert werden, die zum Spielfeld gehören. Sofern dafür eine Lösung gefunden wird, hätte dieser Ansatz den Vorteil, dass Objekte auf den Linien, wie ein Spieler, die Spielfelderkennung nicht stören, sodass sie zuverlässiger ausgeführt werden könnte.

Die **Ballerkennung** konnte in einigen Fällen vielversprechende Ergebnisse erzielen. Dafür war eine gute Erkennung des Spielfeldes notwendig, da der Ball erst dann gesucht werden kann, wenn das Spielfeld identifiziert wurde. Weiterhin ist eine gute Videoqualität mit statischer Kameraführung vorteilhaft, um den Ball mit der in dieser Arbeit eingesetzten Implementierung zuverlässig erkennen zu können. Ausschließlich in Spielernähe zeigte sich, dass der Ball in manchen Situationen mit Körperteilen des Spielers verwechselt wurde und folglich zu dem Zeitpunkt nicht erkannt werden konnte oder der Ball in dem Ballwechsel komplett verloren ging. Dies resultierte daraus, dass für die Ballerkennung lediglich die **bewegten Bildpunkte in einem Suchbereich** geclustert wurden und der Ball anhand der **Clustergrößen und der Distanz** gegenüber der zuletzt erkannten Ballposition identifiziert wurde. Dadurch kann nicht sichergestellt werden, dass es sich bei dem erkannten Cluster um den Ball handelt, auch wenn sich die Größendifferenz und die Distanz gegenüber der vorherigen Ballposition nur wenig unterscheiden und damit im akzeptierten Bereich liegen.

Um diese Falschidentifikationen zu vermeiden, könnte die Ballerkennung durch ein **Pattern Matching** Verfahren[1] oder durch Farbüberprüfungen der Objekte verbessert werden, sodass nicht ausschließlich die Größe und die Distanz der Cluster entscheidend sind, sondern auch die Form und Farbe des Objektes die Erkennung mit beeinflussen.

Auch die Ballposition konnte anhand dieser beiden Merkmale nicht immer genau bestimmt werden, da der Ball in Spielernähe häufiger falsch identifiziert oder erst gar nicht erkannt wurde. Eine Lösung dafür könnte sein, die zuvor erkannte Flugkurve mit einer **Physik Engine**[2] weiterzuberechnen, sodass der Ball in Spielernähe zwar nicht durch die Bildverarbeitung erkannt wird, aber die nächsten Ballpositionen durch eine Berechnung der Balltrajektorie bestimmt werden können. Sofern der Ball verloren geht, ist eine neuerliche Erkennung mittels der in dieser Arbeit eingesetzten Ballerkennung unumgänglich, da die Flugkurve nur bis zum nächsten Schlag vervollständigt werden kann. Diese Identifikation könnte in Netznähe erfolgen, da der Ball diesen Bereich auf jeden Fall durchqueren muss, sofern der Ballwechsel nach dem letzten Schlag fortgesetzt wurde. Dieses Vorgehen hilft auch bei schlechterer Bildqualität der Fernsehaufnahmen, in denen der Ball häufiger ohne ersichtlichen Grund verloren geht.

[1] Pattern Matching Verfahren: sucht ein Objekt anhand eines vorgegebenen Musters (engl. pattern), indem es Merkmale des Musters mit dem Objekt vergleicht. [15, S.380]

[2] Physik Engine: dient zur realistischen Simulation von physikalischen Systemen. [3]

Ein anderer Ansatz die Ballpositionen in Spielernähe besser erkennen zu können ist, die **Bewegung des Spielers mitzuverfolgen**, sodass sich der Schnittpunkt der Ballflugkurve mit der Laufrichtung des Spielers als Treffpunkt festlegen lässt. Auch durch die **Analyse der Tonspur** könnte der Zeitpunkt des Schlages oder des Bodenkontaktes des Balles registriert werden. Diese sollten sich durch einen kürzeren und lauteren Knall von den Hintergrundgeräuschen abheben.

Ein Problem bei der Ballerkennung ist, dass durch die 2D Fernsehaufnahmen die erkannte **Position des Balles im Bild verzerrt** dargestellt wird. In Abhängigkeit der Höhe des Balles über dem Boden weicht dieser mal mehr, mal weniger von seiner realen Position auf dem Spielfeld ab. Berührt der Ball im betrachteten Zeitpunkt den Boden, so entspricht dies seiner wirklichen Position. Befindet er sich in gewisser Höhe darüber, so wird aufgrund der perspektivischen Ansicht eine andere Position angenommen. Um diese Abweichung zu dem für die Taktikanalyse wichtigen Schlagzeitpunkt zu korrigieren, kann der **Bodenkontaktpunkt des Spielers** (also die Füße) als tatsächliche Ballposition herangezogen werden. Hierbei wird davon ausgegangen, dass sich der Ball beim Schlagzeitpunkt in unmittelbarer Nähe des Spielers befindet. Dadurch kann die fehlende Höhenangabe des Balles herausgerechnet und der Ball auf eine XY-Ebene des Spielfeldes projiziert werden, sodass die reale Spielfeldposition des Balles zum Schlagzeitpunkt festgestellt werden kann.

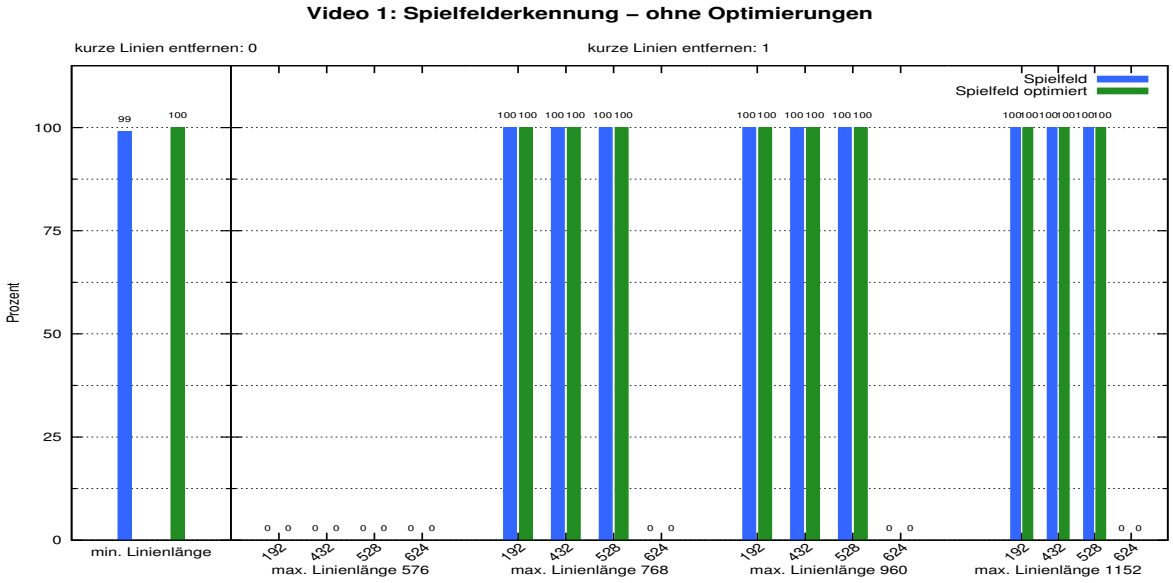
Eine weitere Möglichkeit die Genauigkeit der Ballerkennung zu verbessern ist es, anstelle der Verwendung von Differenzbildern lediglich den **Hintergrund vom aktuellen Bildframe zu entfernen**, sodass einzig die bewegten Spieler und Bälle als relevante Objekte übrig bleiben. Bei Differenzbildern wird mit Hilfe der pixelweisen Subtraktion zweier Bildframes auf Bewegungen geschlossen. Dabei werden die Bewegungen von Spieler und Ball doppelt dargestellt, was die Ballerkennung stören kann. Wenn lediglich der Hintergrund von einem einzigen Bildframe entfernt wird, bleiben die Spieler und der Ball in ihrer ursprünglichen Form erhalten, ohne doppelte Darstellung. Dies würde einer genaueren Ballerkennung zu Gute kommen.

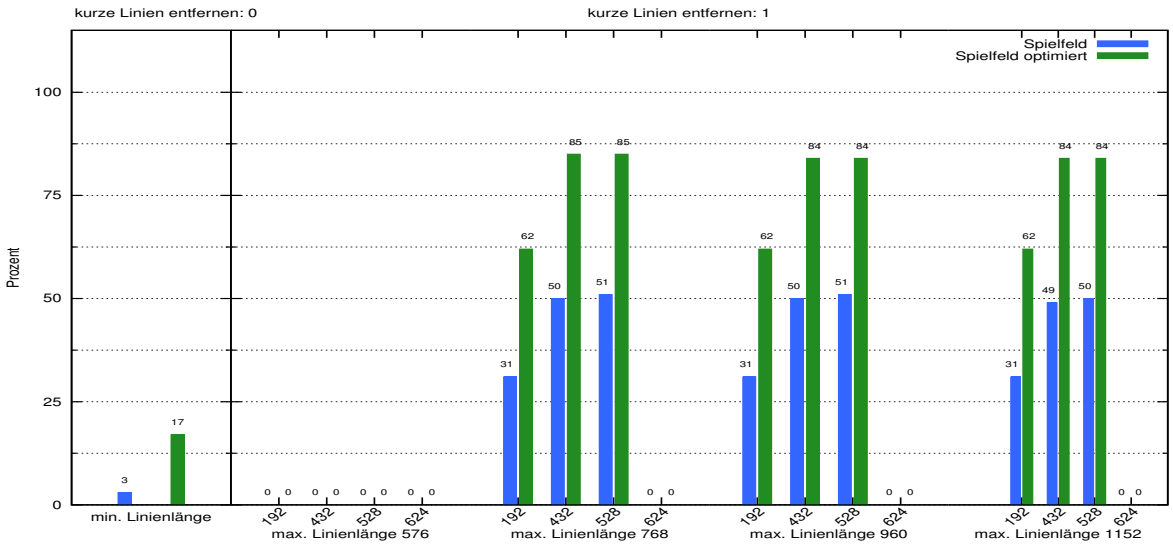
Zusammenfassend kann gesagt werden, dass die Ballerkennung vielversprechend erscheint, insbesondere bei statischer Kameraführung, guter Bildqualität, hohem Kontrast zum Spielfelduntergrund und sofern das Spielfeld gut erkannt werden konnte. Da jede Spielfeldumgebung ihre eigenen Charakteristika besitzt und sich diese von Turnier zu Turnier unterscheiden, kann es bei der Spielfelderkennung zu Problemen kommen, was eine Ballerkennung unmöglich macht. Daher sollte es in einem nächsten Schritt das Ziel sein die Spielfelderkennung zuverlässiger zu gestalten, sodass die Ballerkennung in allen Spielsituationen angewandt werden kann. Darüber hinaus wurde festgestellt, dass es Verbesserungsbedarf bei der Erkennung des Balles gibt, da ansonsten der Ball, auch aufgrund von Falschidentifikationen vor allem in Spielernähe, nicht genau erkannt werden kann. Die Ballpositionen zum Schlagzeitpunkt und bei Bodenkontakt sind für weitere Taktikanalysen besonders interessant. Sofern die Spielfelderkennung zuverlässiger wird und die Ballpositionen in Spielernähe genauer bestimmt werden können, steht einer automatischen Ballerkennung zur Taktikanalyse nichts mehr im Weg. Ohne diese Verbesserungen eignen sich nur wenige Fernsehaufnahmen, sodass keine aussagekräftige Taktikanalyse möglich ist.

A. Anhang

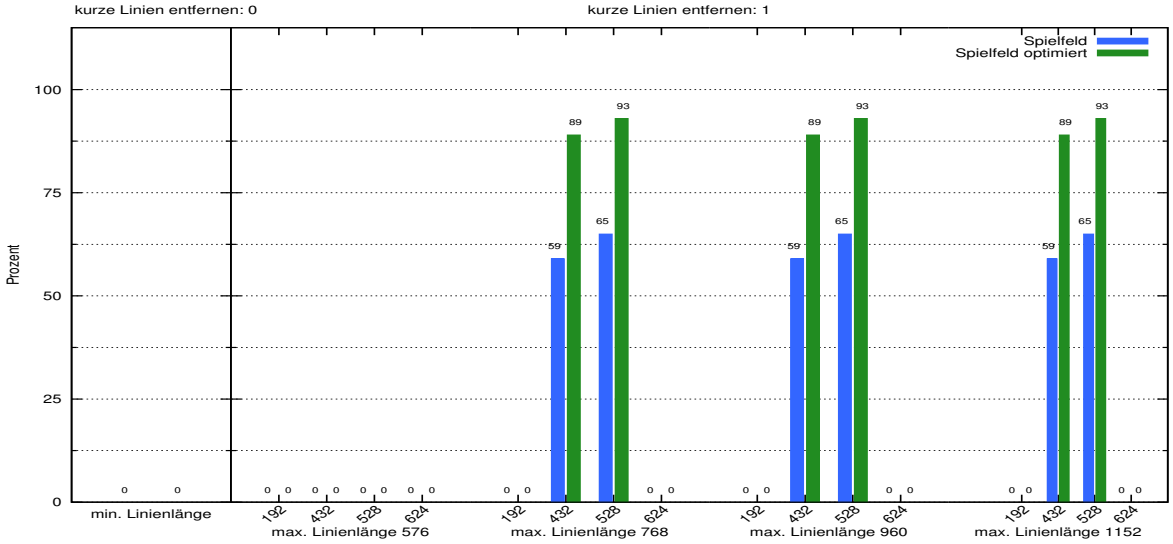
A.1. Diagramme

A.1.1. Analyse der Spielfelderkennung

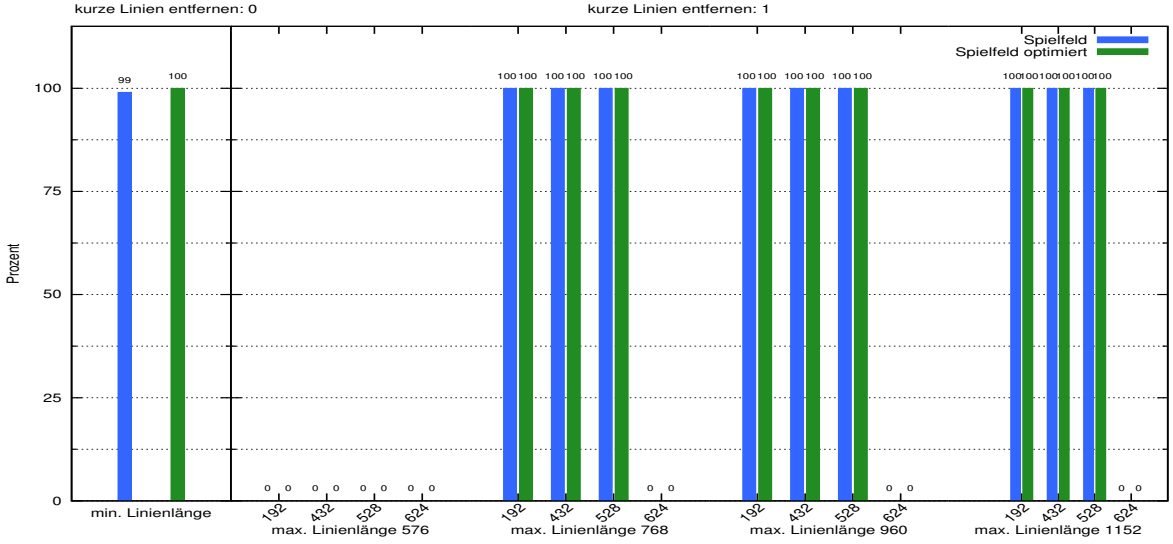




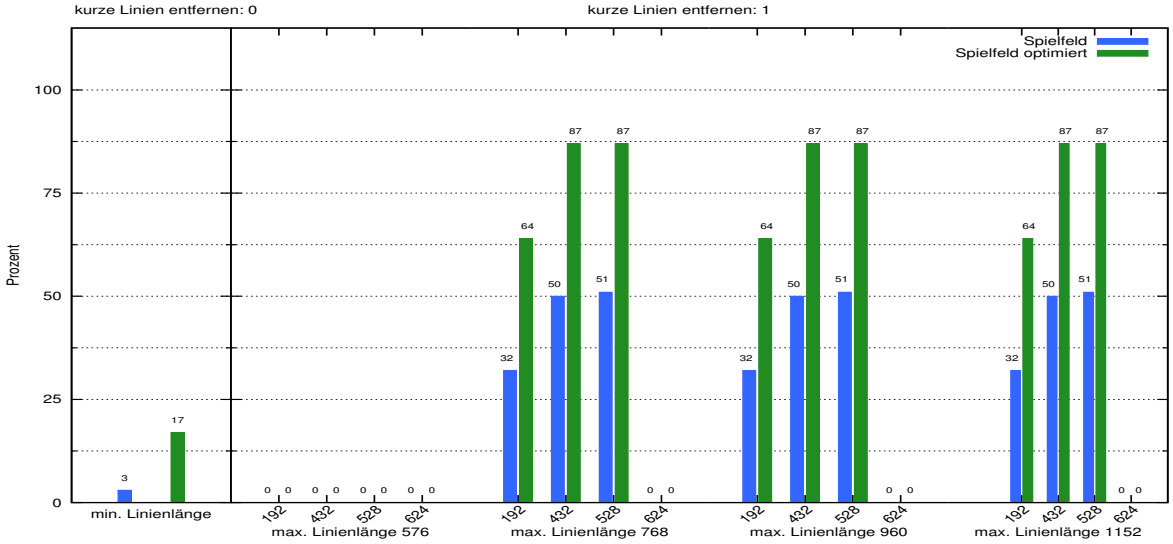
Video 3: Spielfelderkennung – ohne Optimierungen



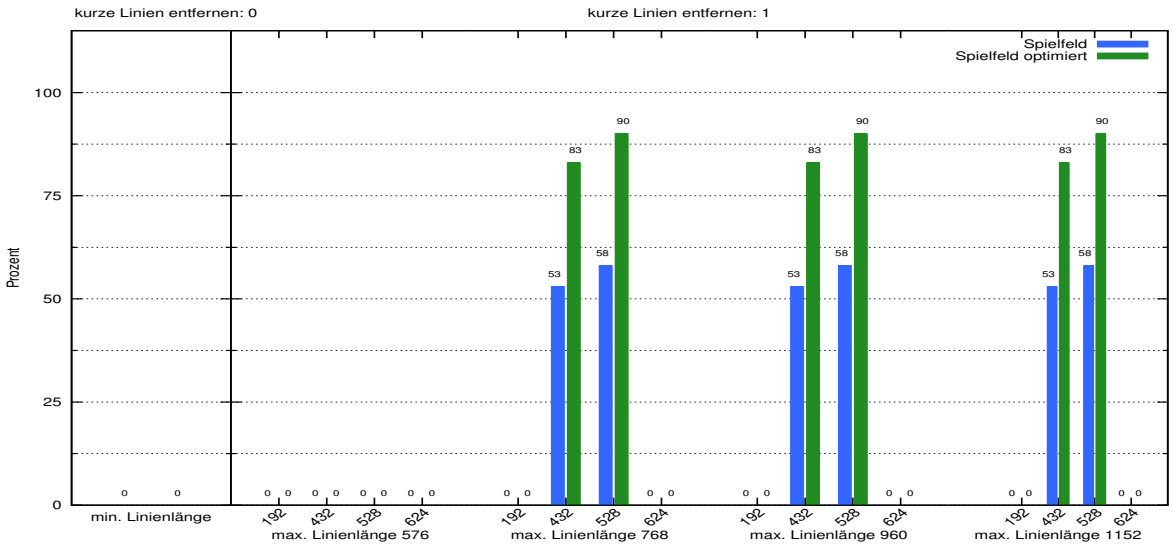
Video 1: Spielfelderkennung – Optimierung anhand Spielfeldrand



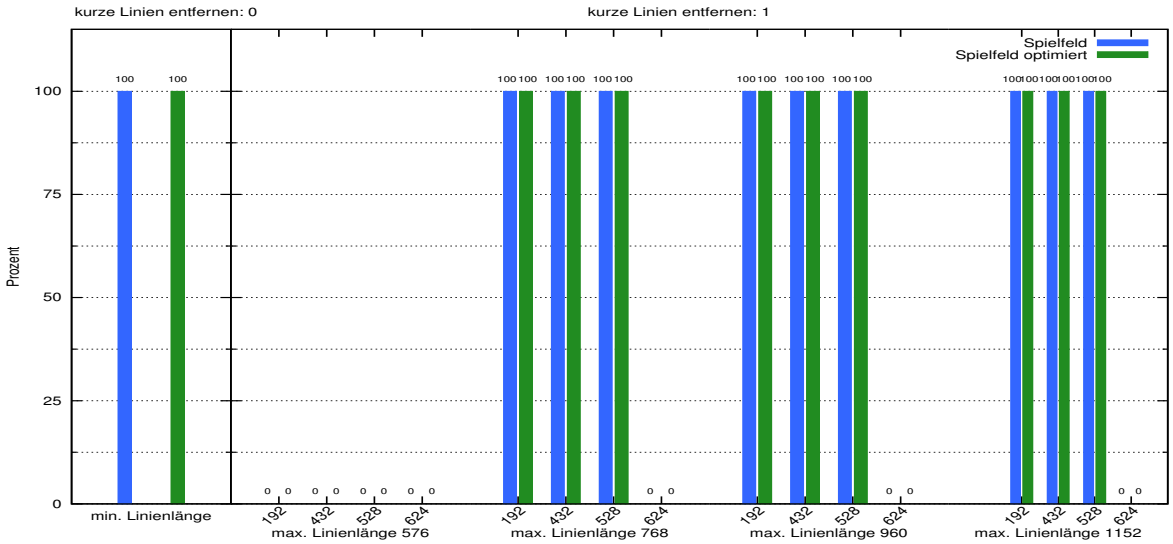
Video 2: Spielfelderkennung – Optimierung anhand Spielfeldrand



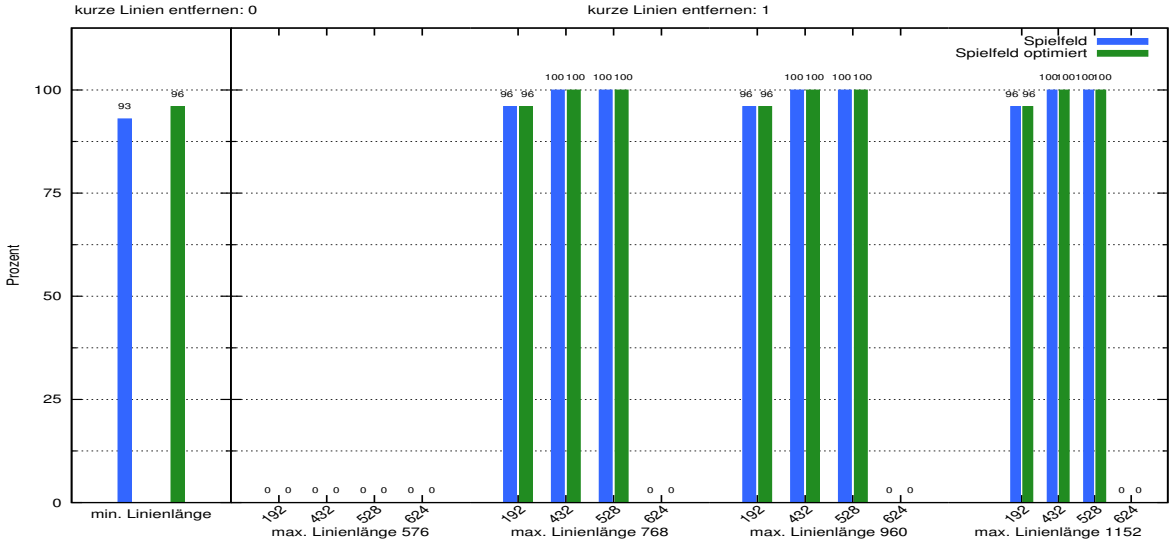
Video 3: Spielfelderkennung – Optimierung anhand Spielfeldrand

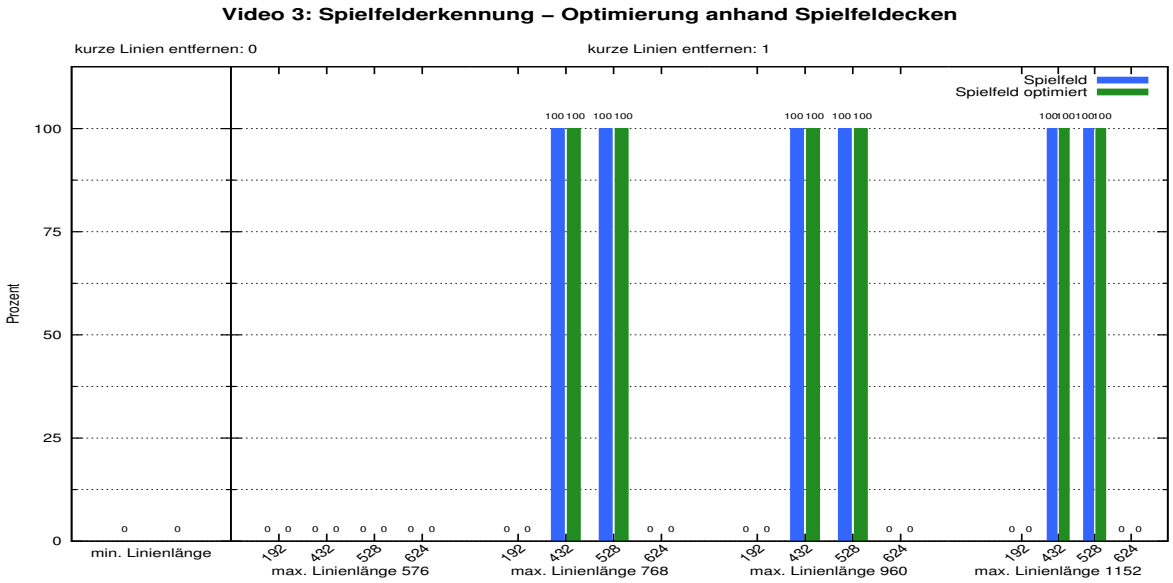


Video 1: Spielfelderkennung – Optimierung anhand Spielfeldecken



Video 2: Spielfelderkennung – Optimierung anhand Spielfeldecken

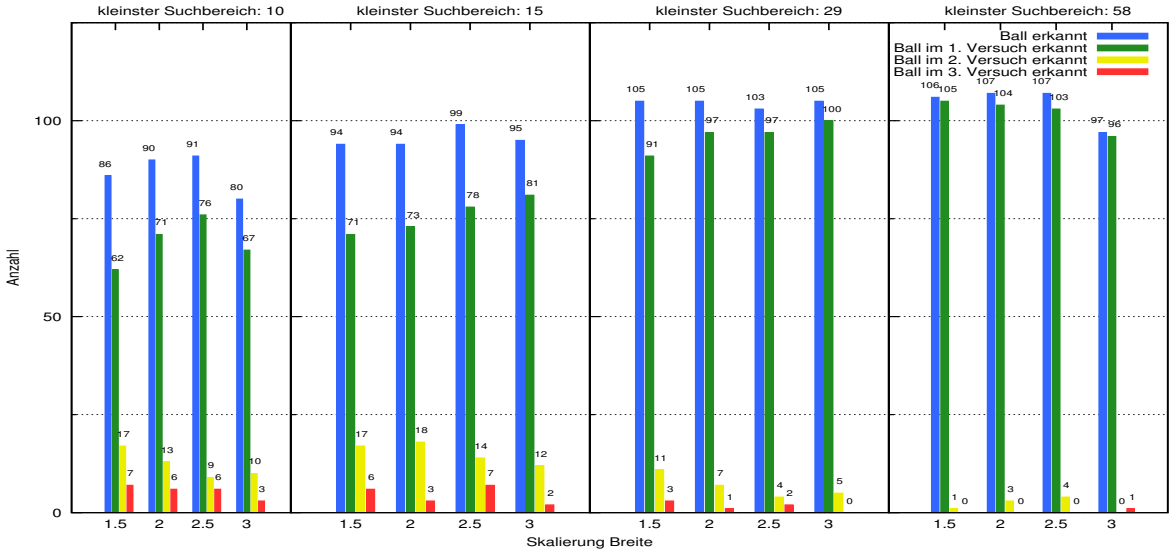



Video 3: Spielfelderkennung – Optimierung anhand Spielfeldecken
kurze Linien entfernen: 0
kurze Linien entfernen: 1
Spielfeld
Spielfeld optimiert
Prozent
100
75
50
25
0
min. Linienlänge
max. Linienlänge 576
max. Linienlänge 768
max. Linienlänge 960
max. Linienlänge 1152
192
432
528
624

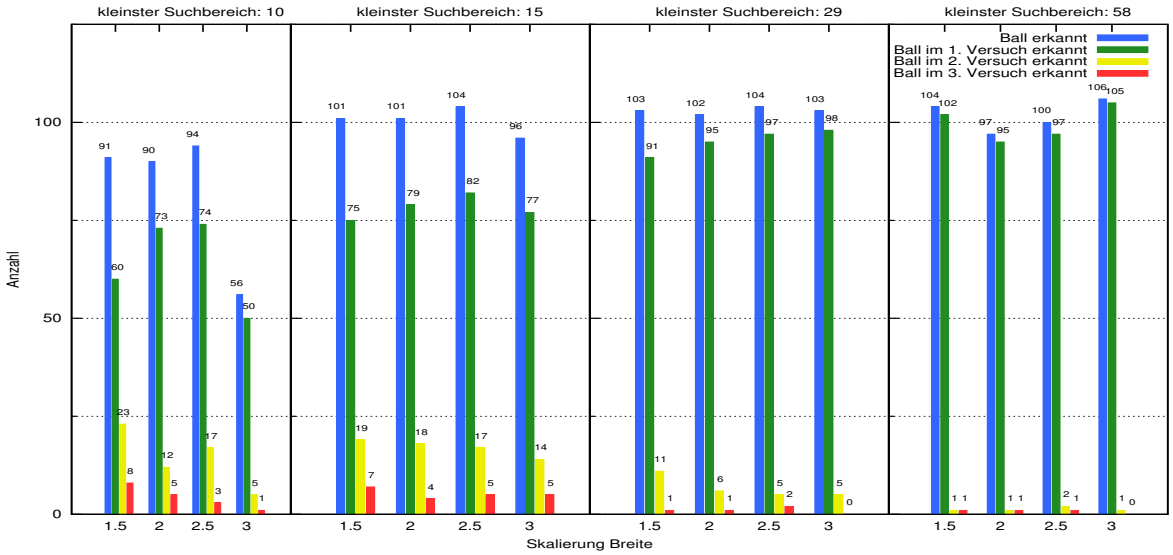
A.1.2. Ballerkennung: Analyse des Suchbereiches

A.1.2.1. Parameter: Art = linear, Verschiebung = anhand Ballposition

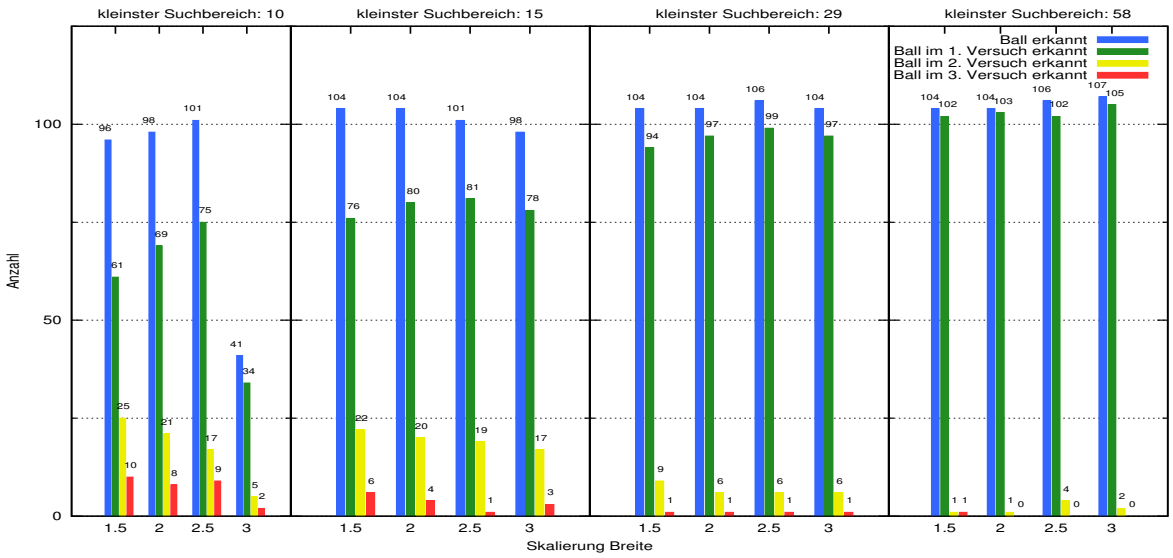
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.5)



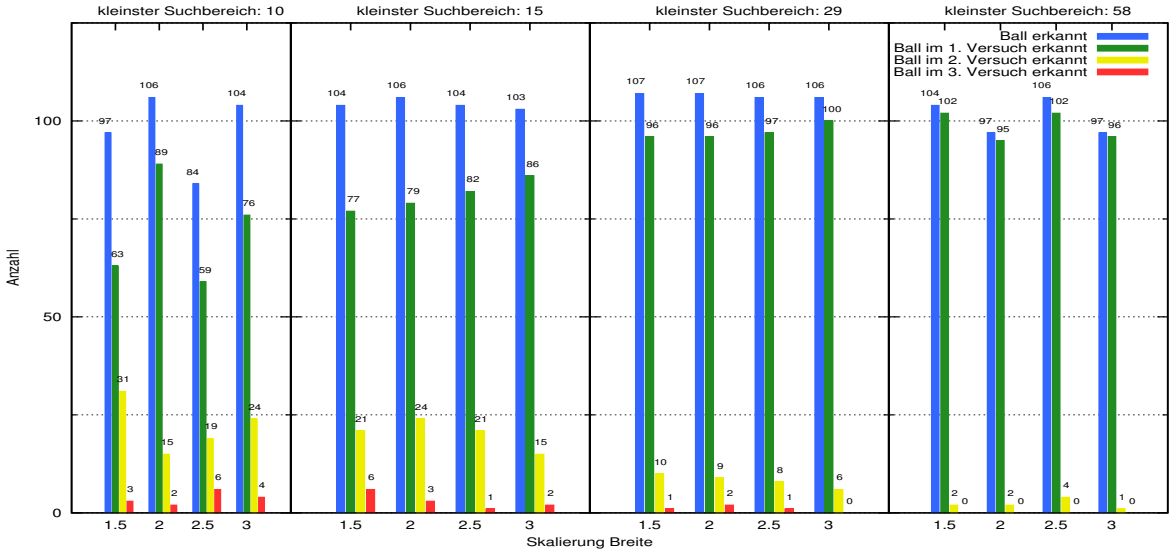
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.7)



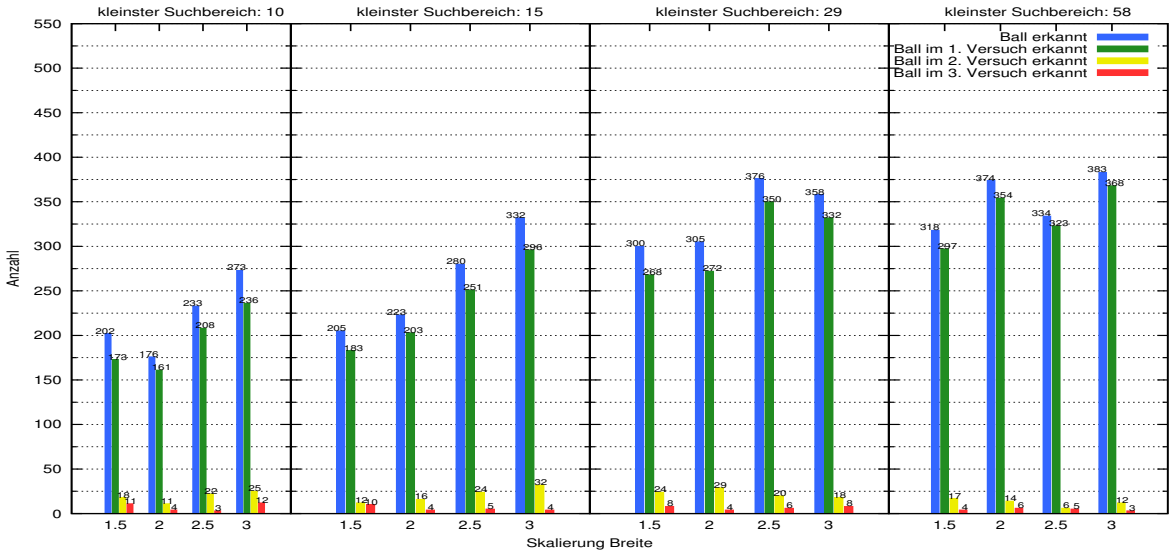
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9)



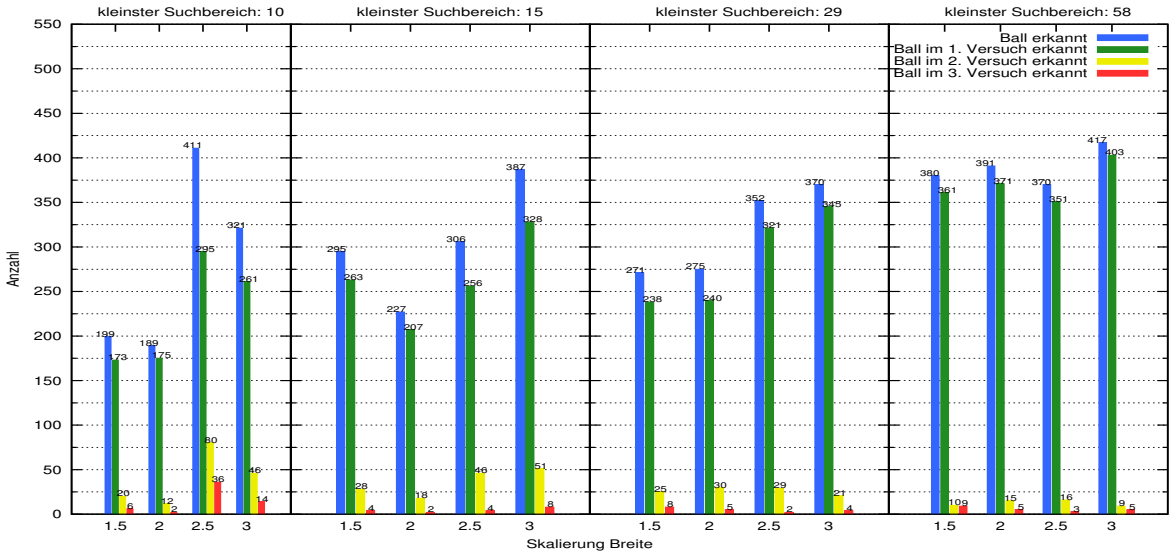
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3)



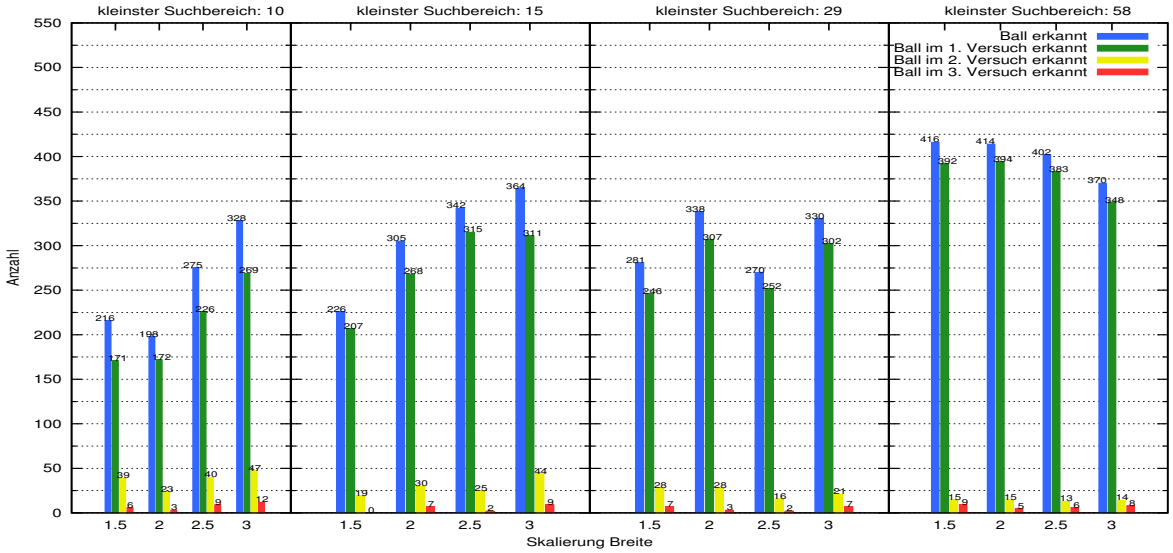
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.5)



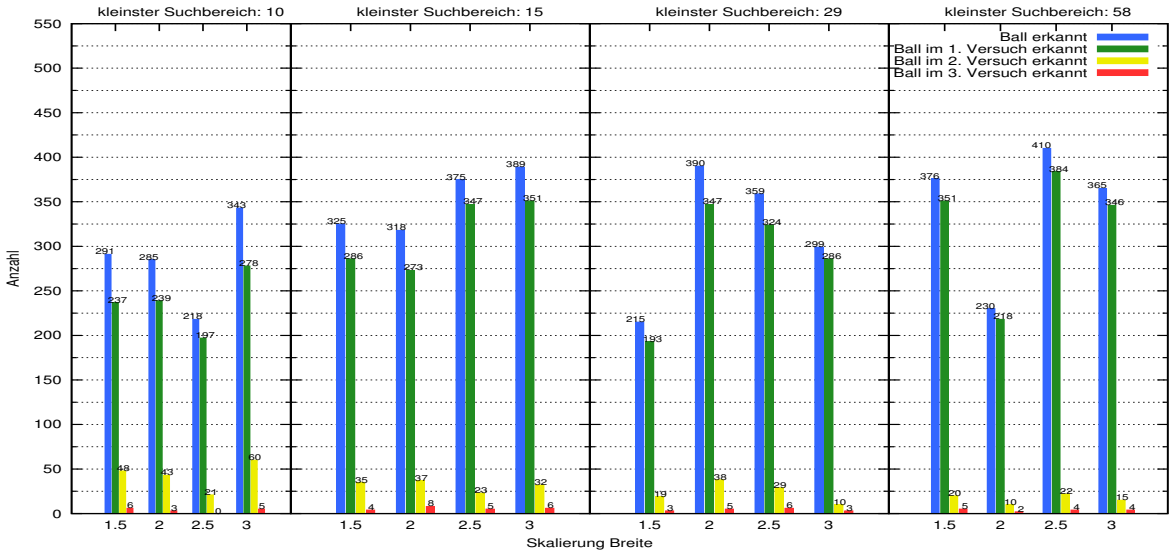
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.7)



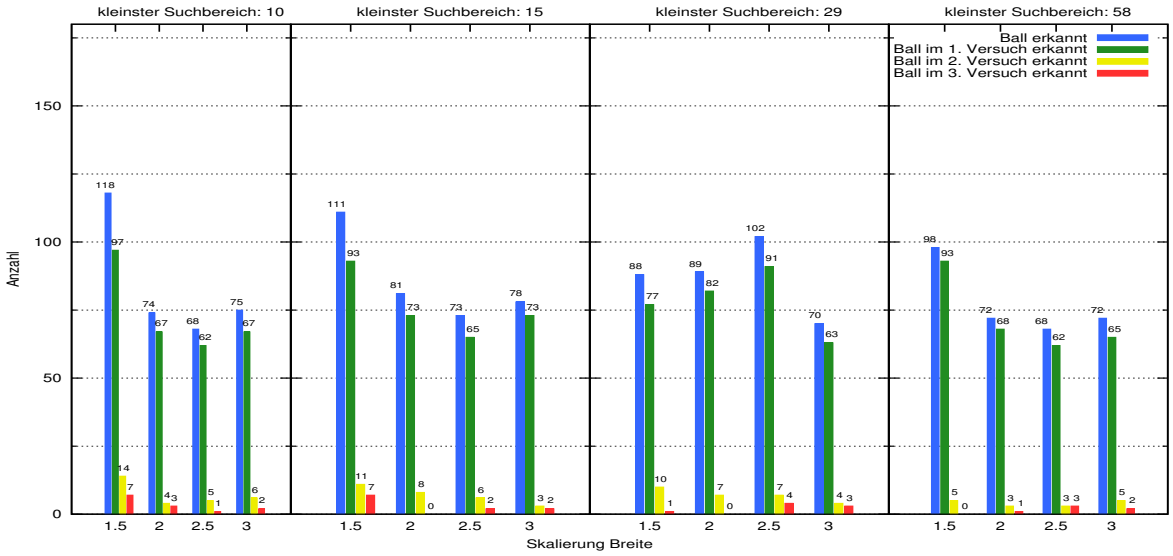
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9)



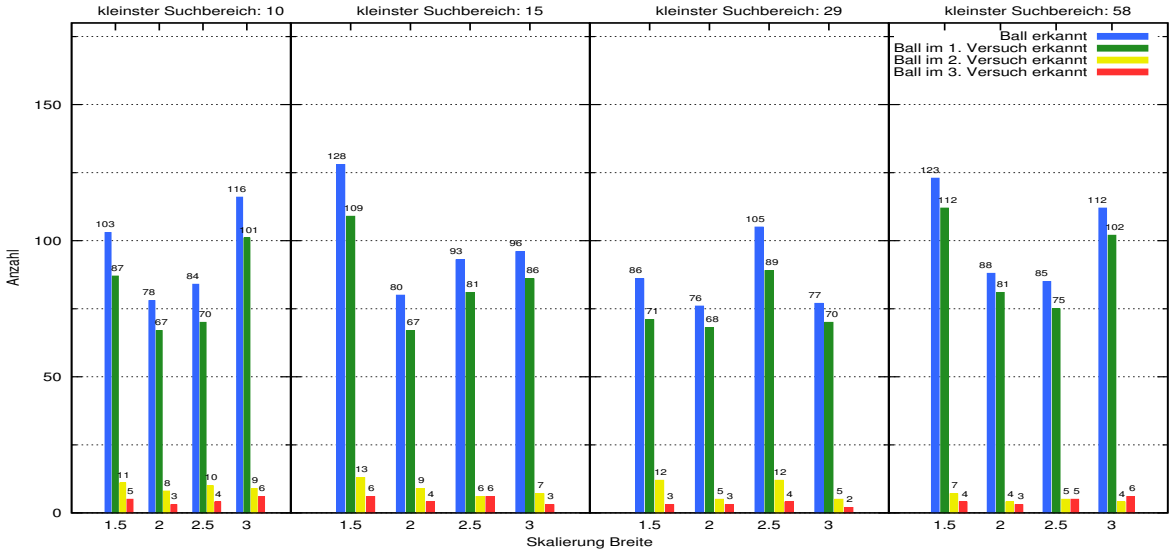
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3)



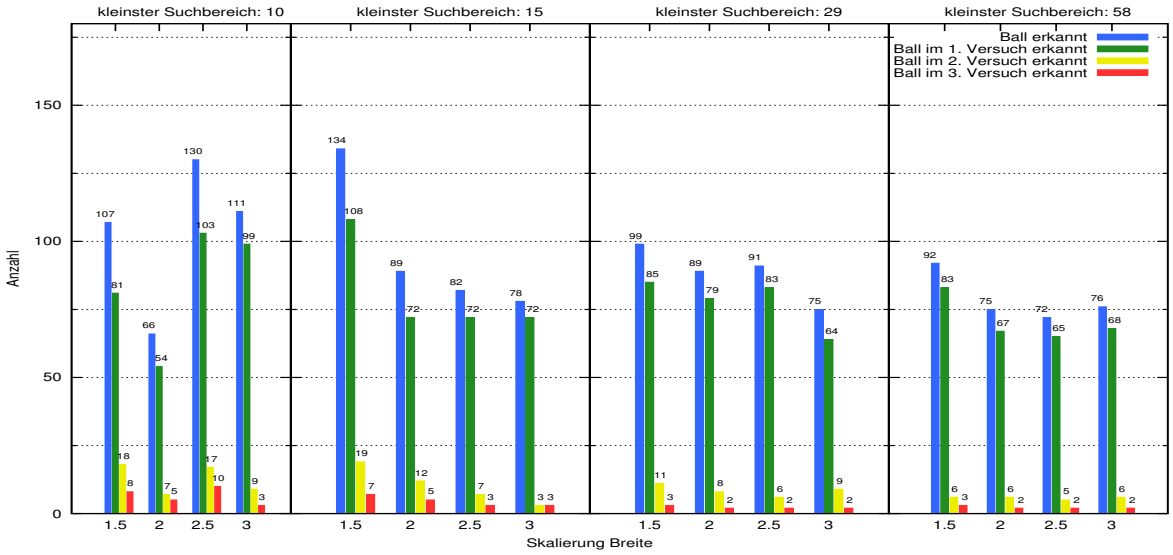
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.5)



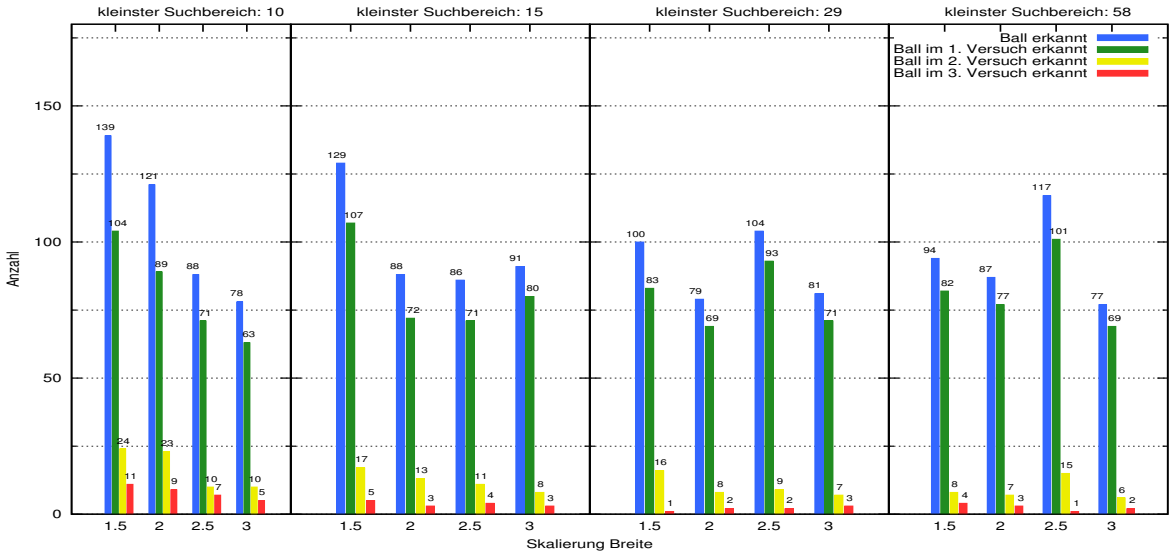
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.7)



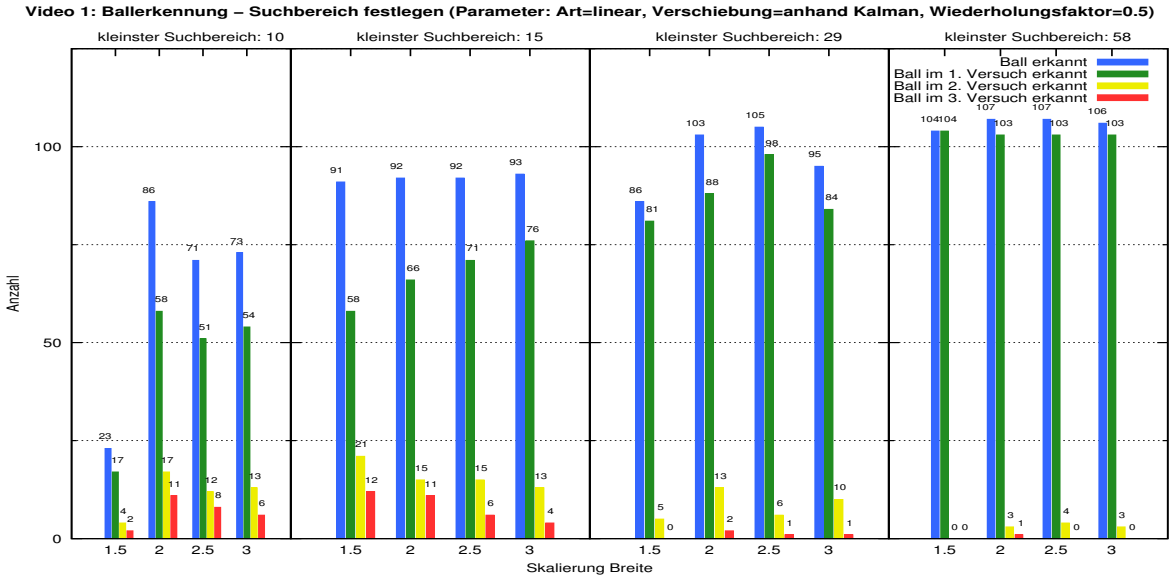
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9)

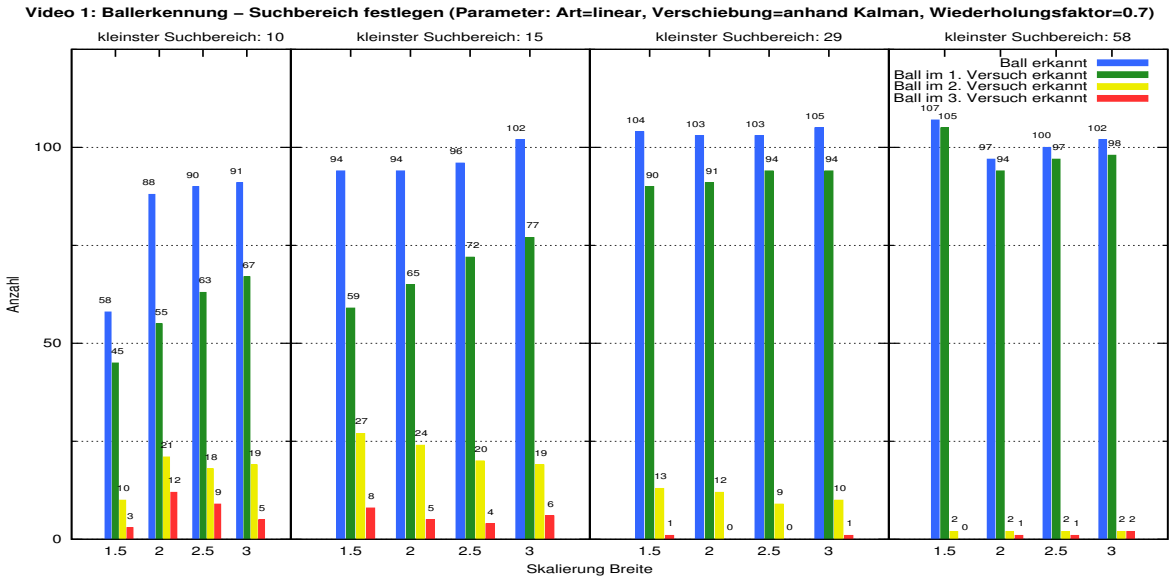


Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3)

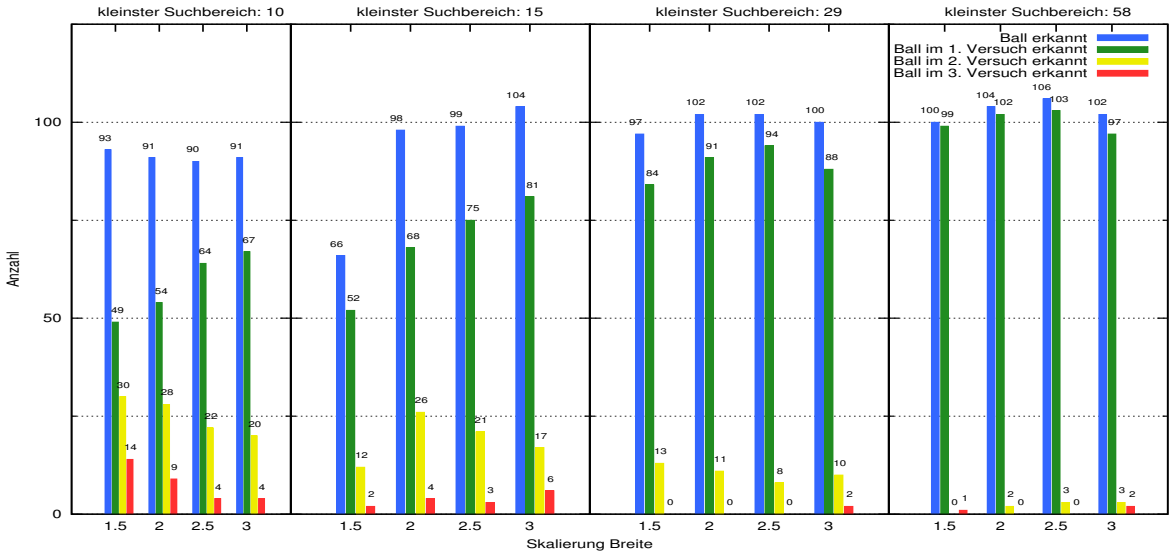


A.1.2.2. Parameter: Art = linear, Verschiebung = anhand Kalman Filter

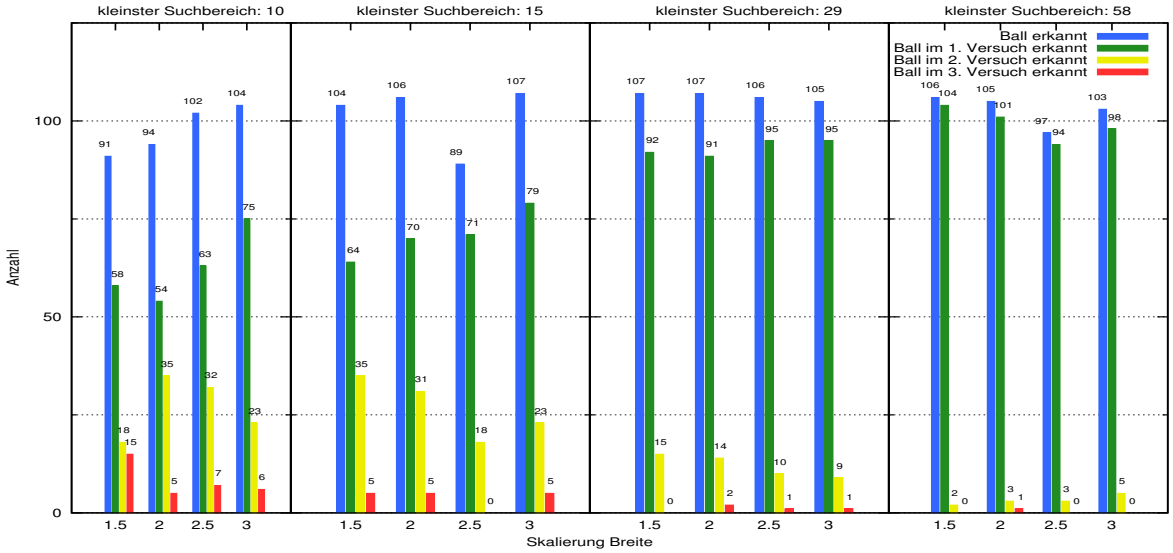




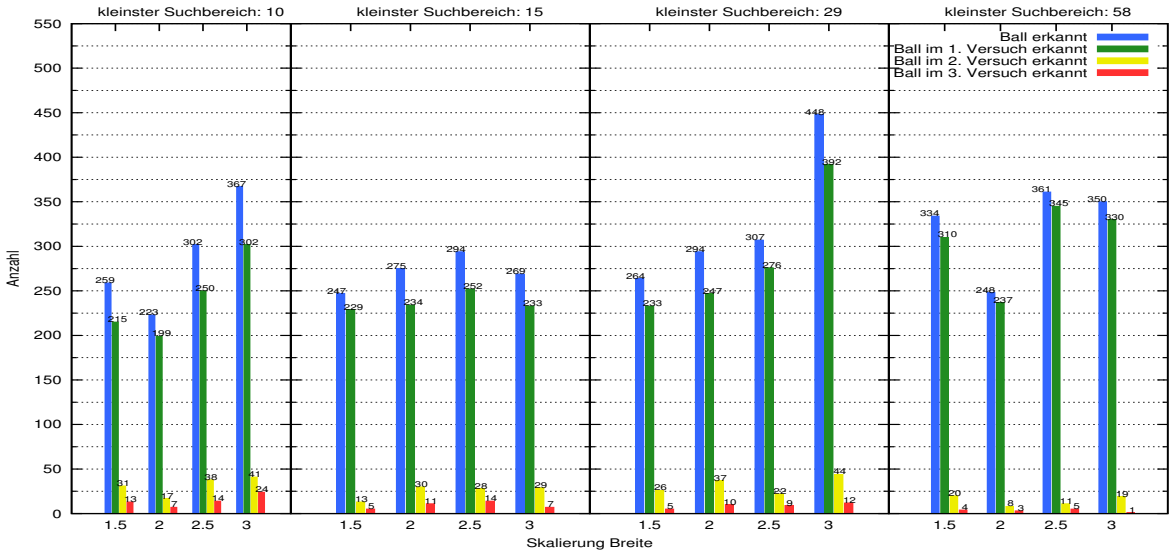
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9)



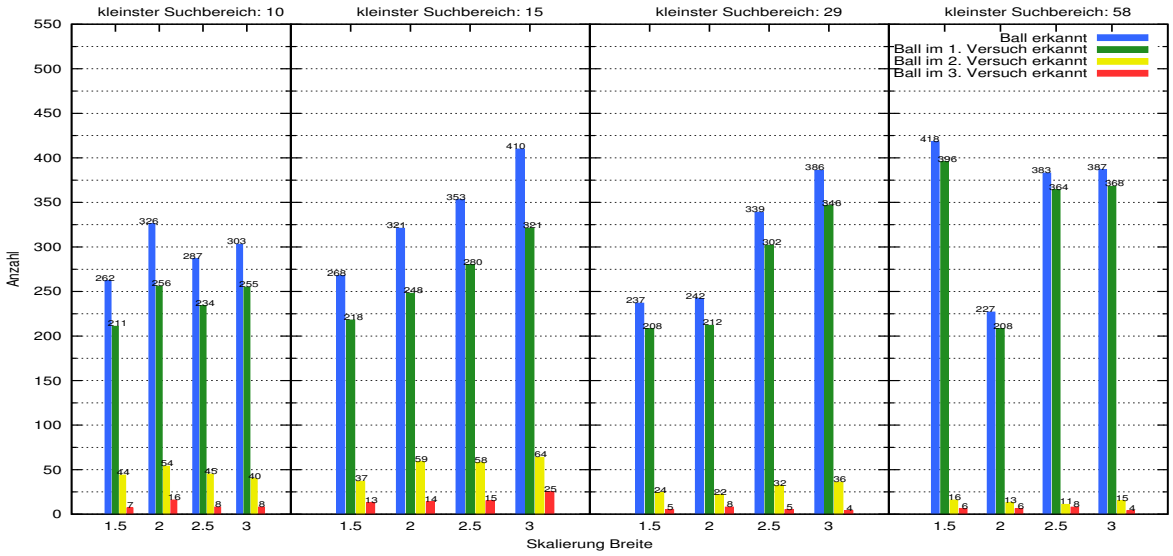
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3)



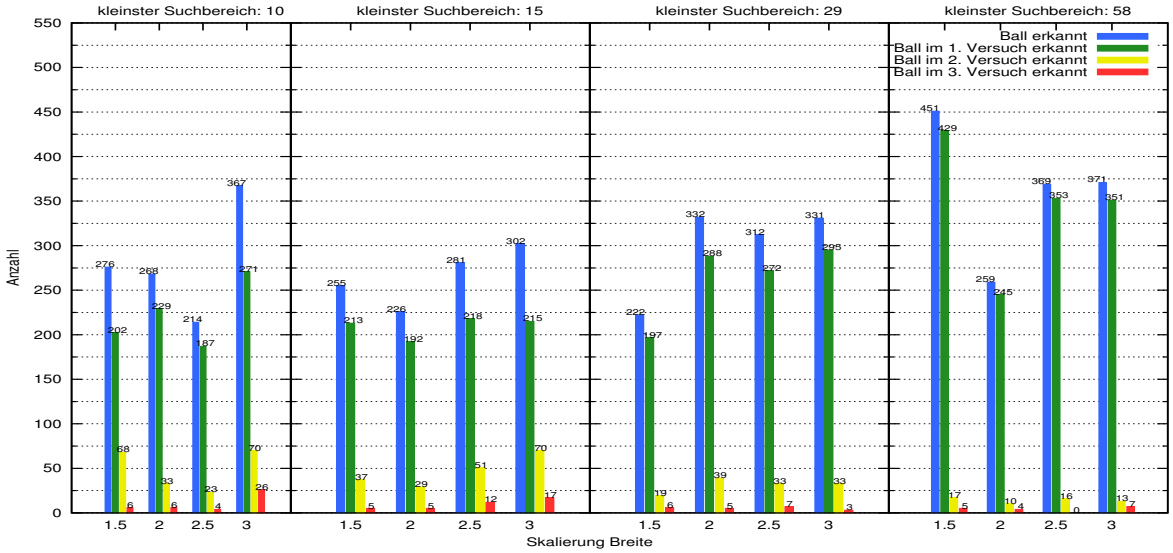
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.5)



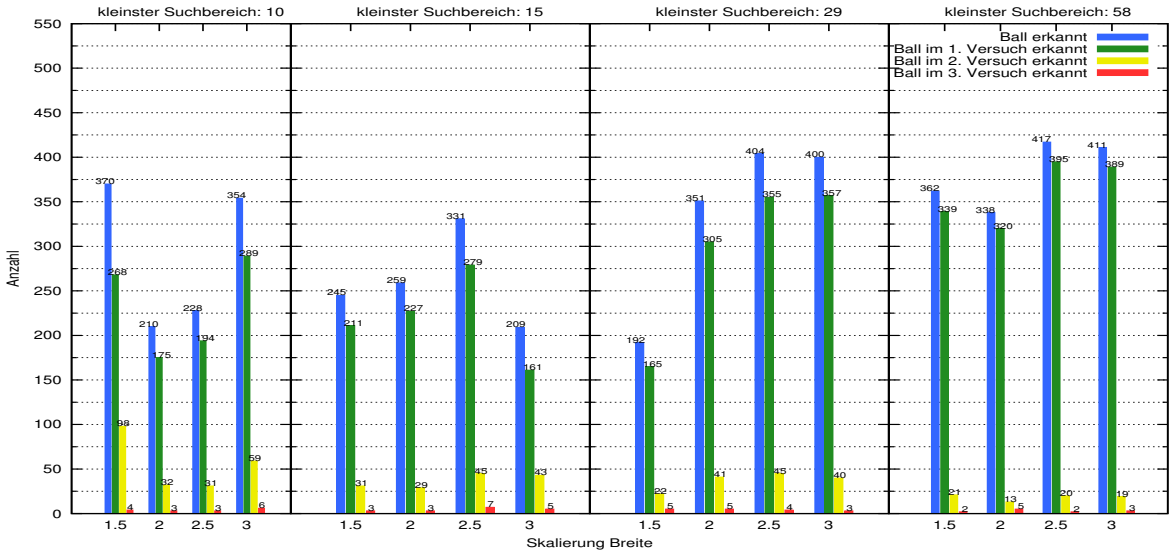
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.7)



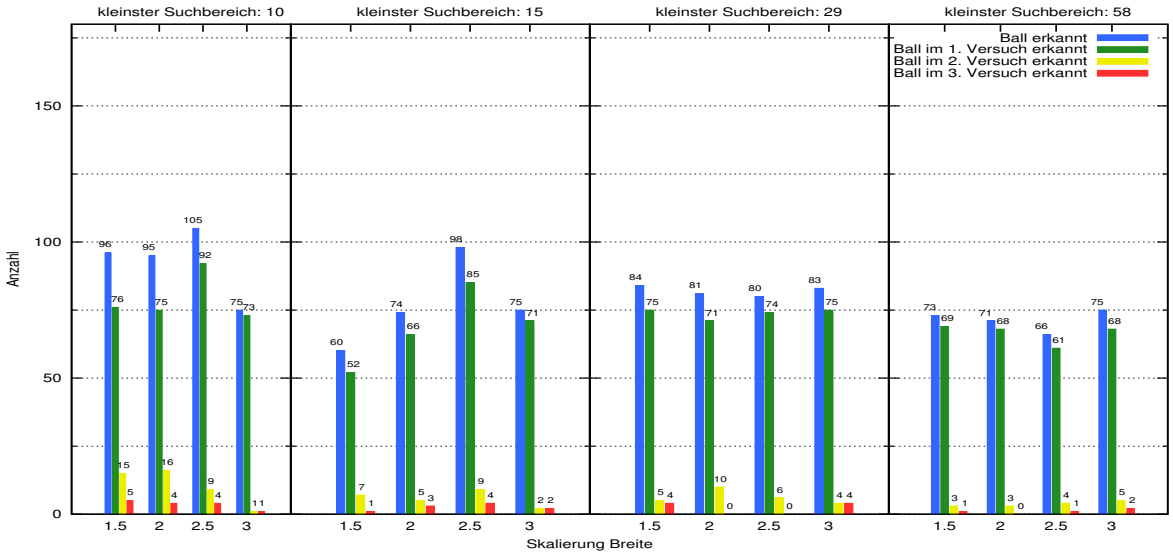
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9)



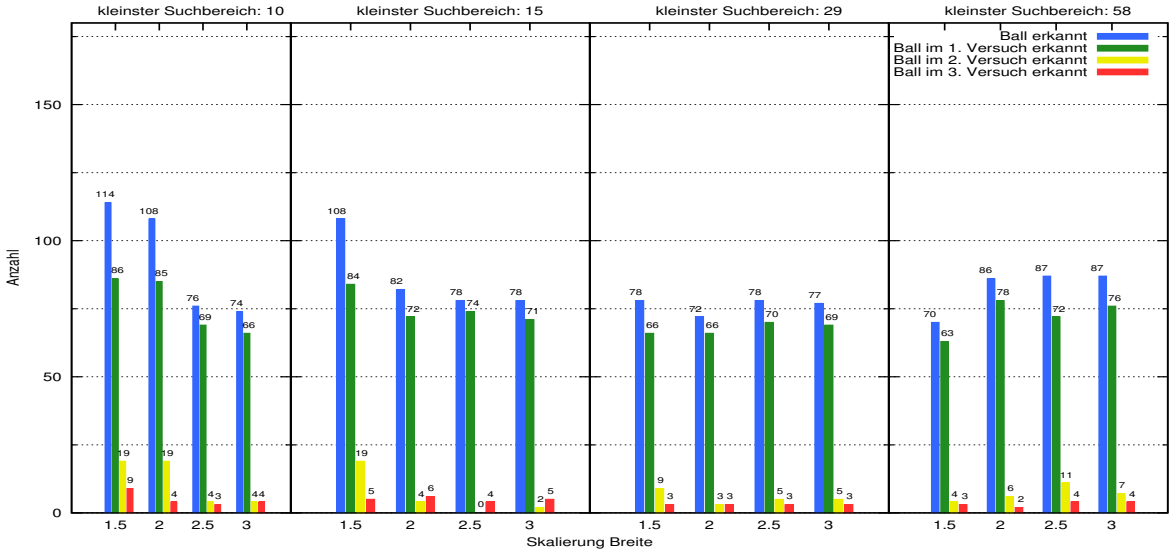
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3)



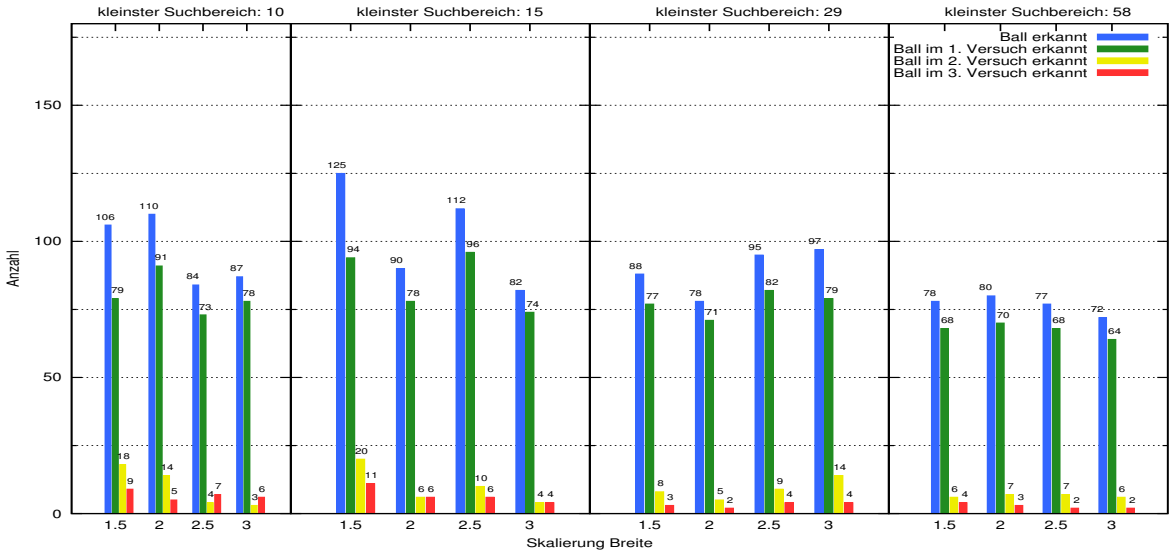
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.5)

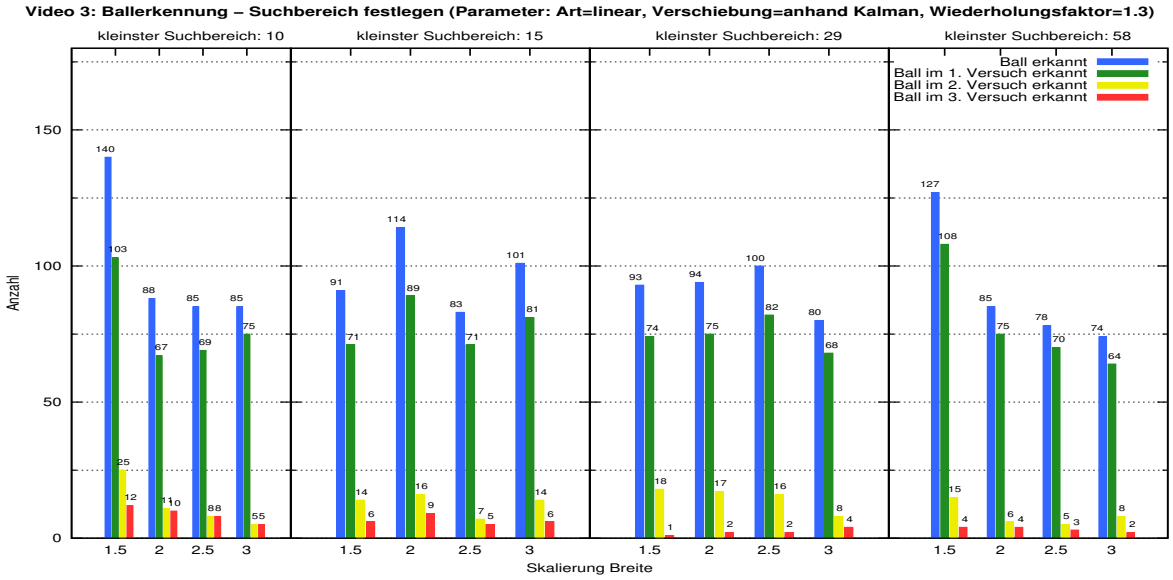


Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.7)

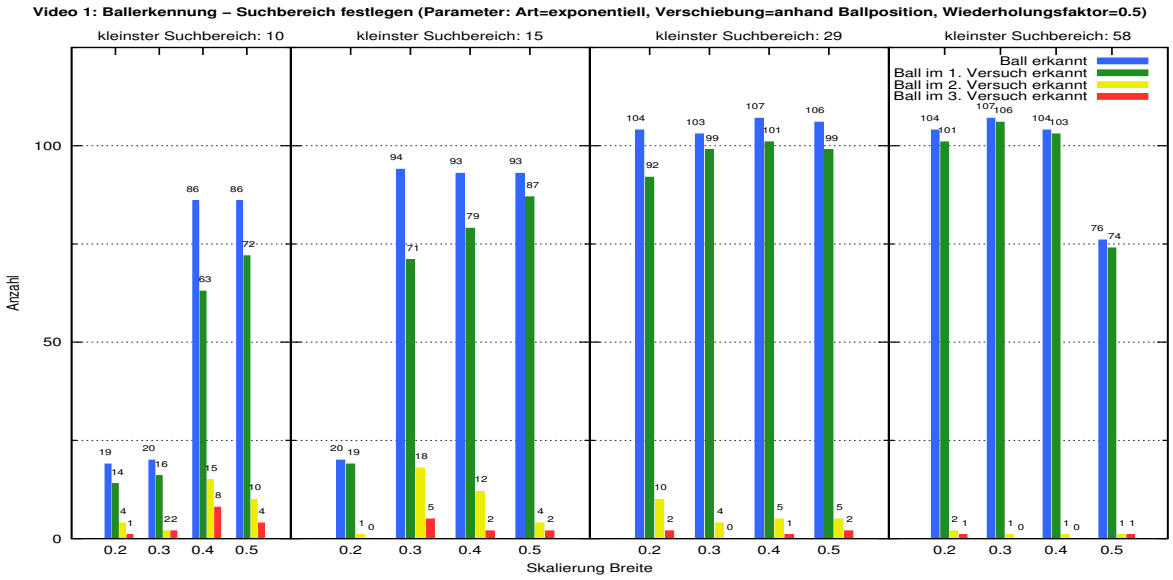


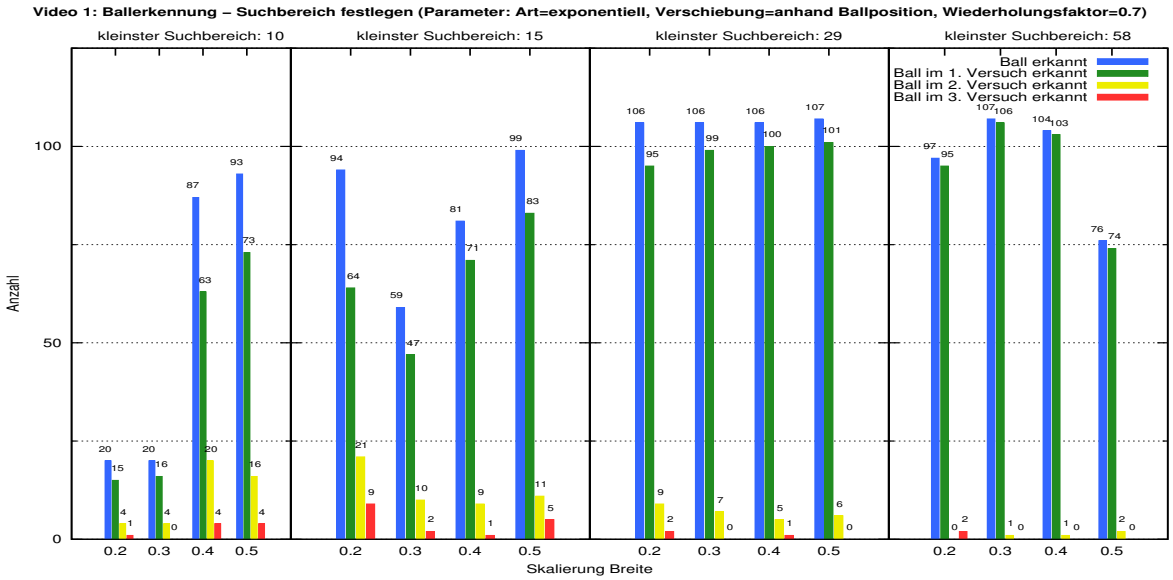
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9)



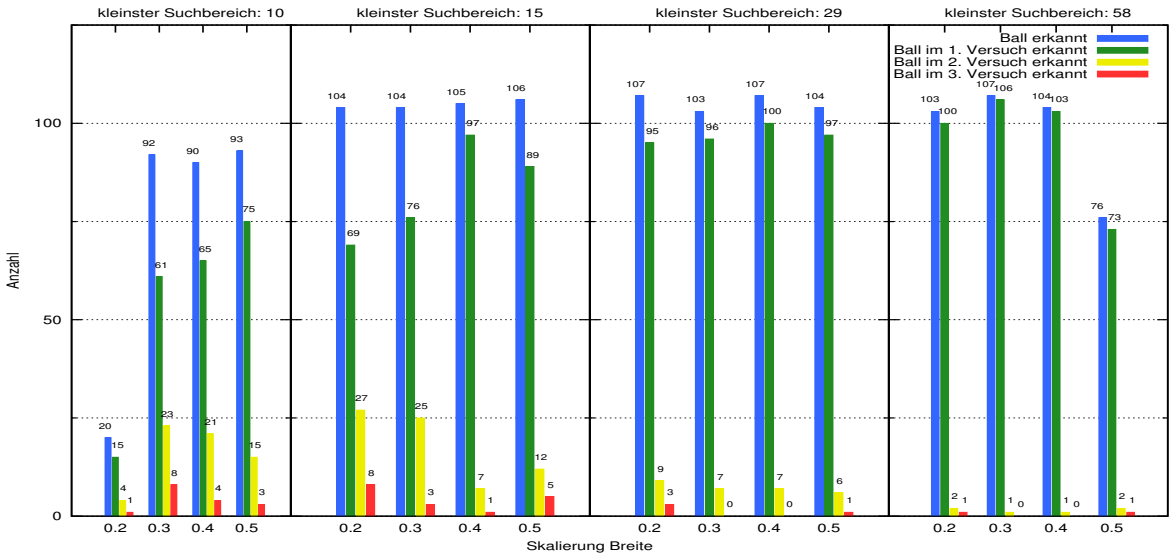

Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3)
kleinster Suchbereich: 10
kleinster Suchbereich: 15
kleinster Suchbereich: 29
kleinster Suchbereich: 58
Ball erkannt
Ball im 1. Versuch erkannt
Ball im 2. Versuch erkannt
Ball im 3. Versuch erkannt
Anzahl
0
50
100
150
140 103 25 12
88 67 11 10
85 69 8 8
85 75 5 5
91 71 14 6
114 89 16 9
83 71 7 5
101 81 14 6
93 74 18 1
94 75 17 2
100 82 16 2
80 68 8 4
127 108 15 4
85 75 6 4
78 70 5 3
74 64 8 2
1.5
2
2.5
3
Skalierung Breite

A.1.2.3. Parameter: Art = exponentiell, Verschiebung = anhand Ballposition

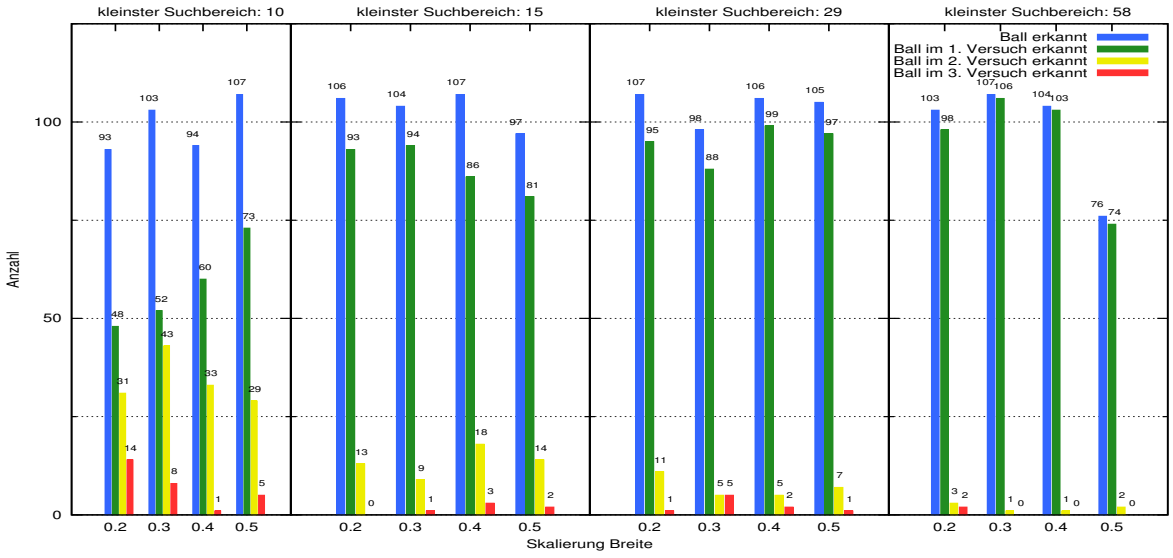




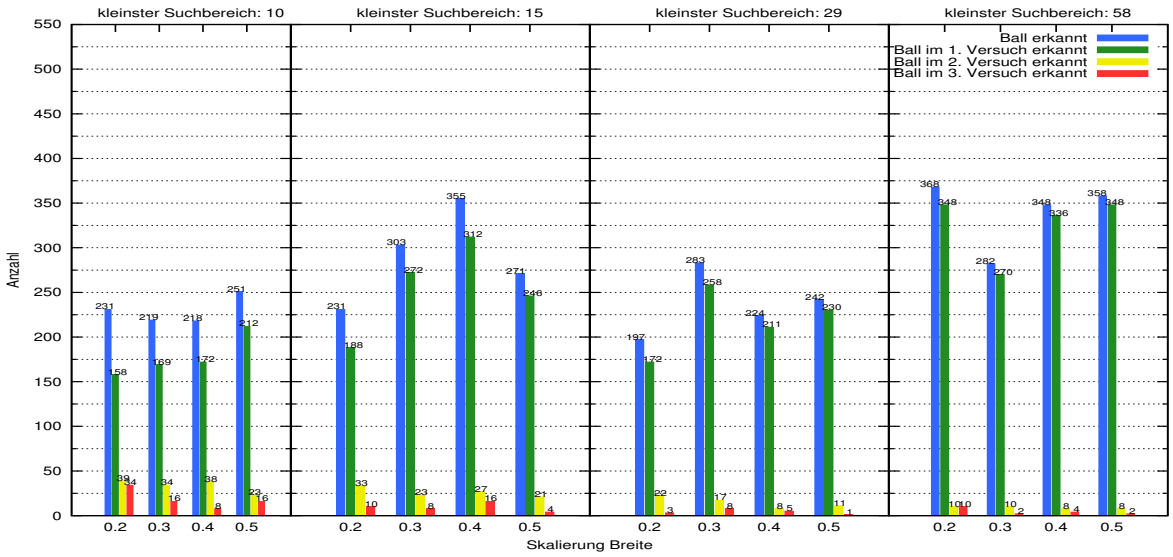
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9)



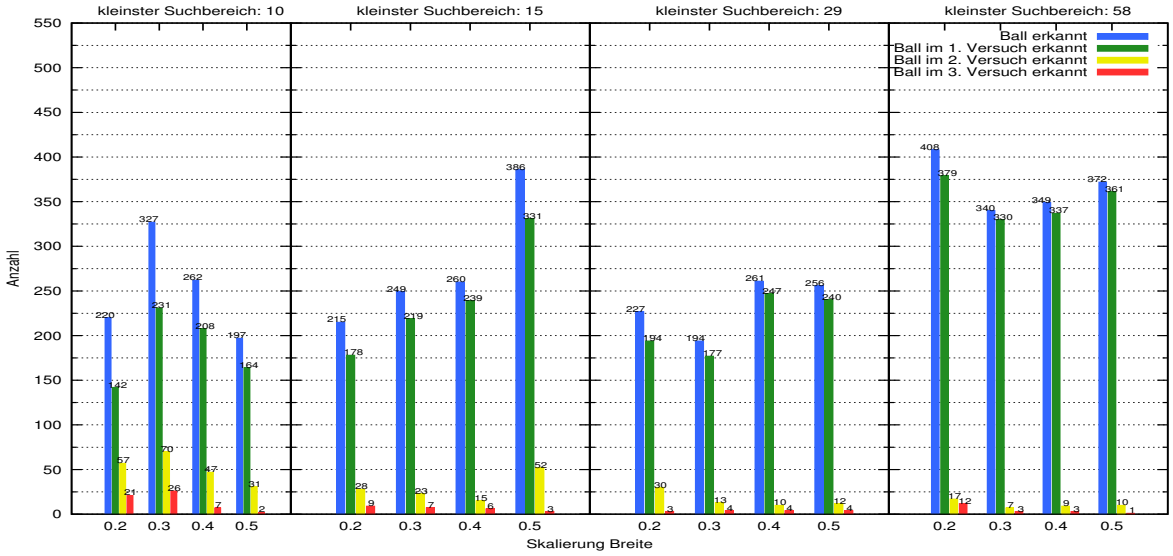
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3)



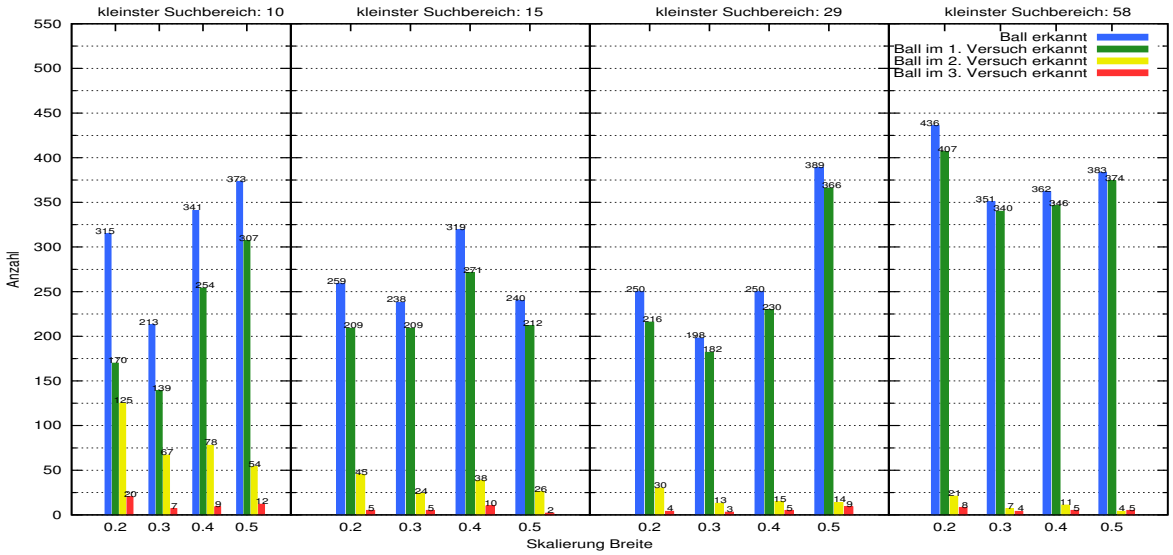
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.5)



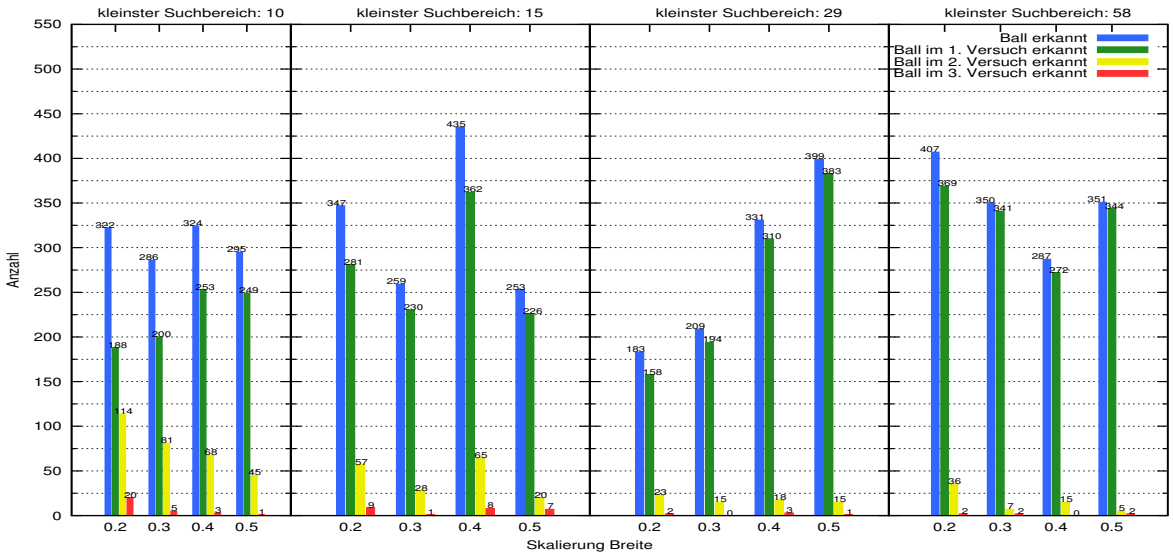
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.7)



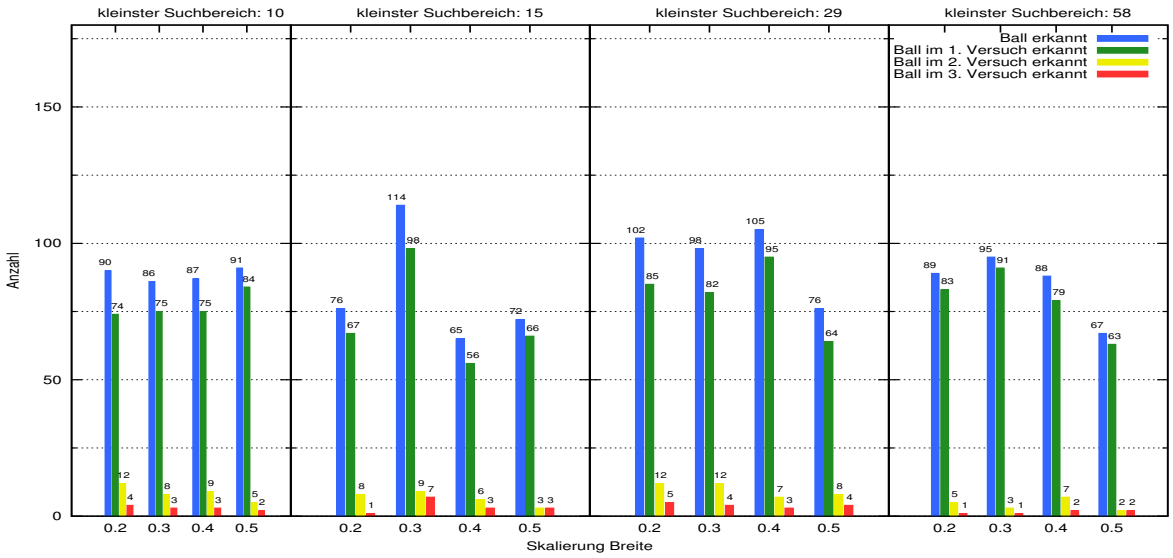
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9)



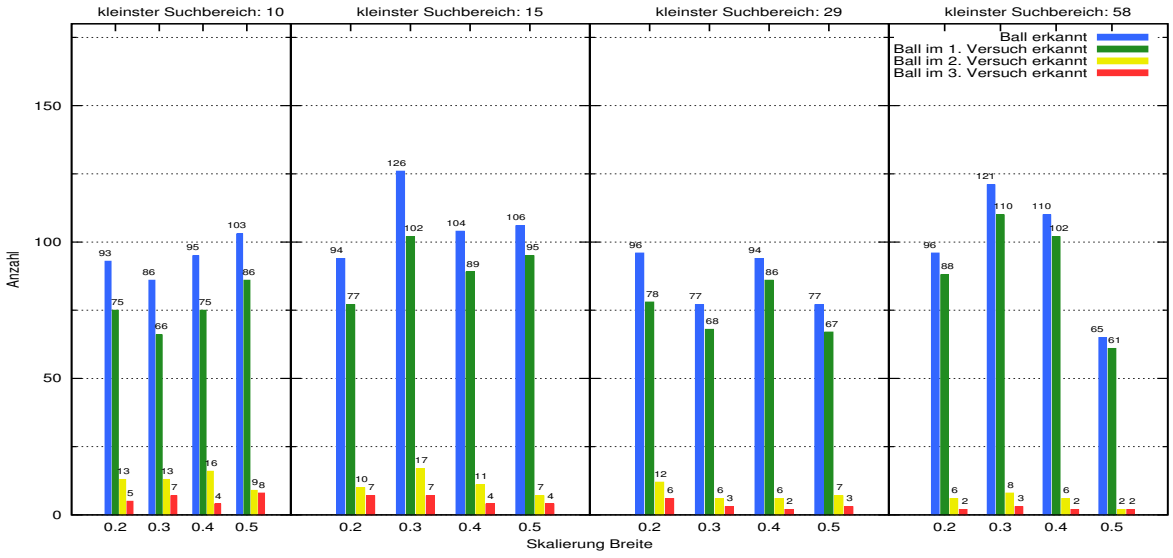
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3)



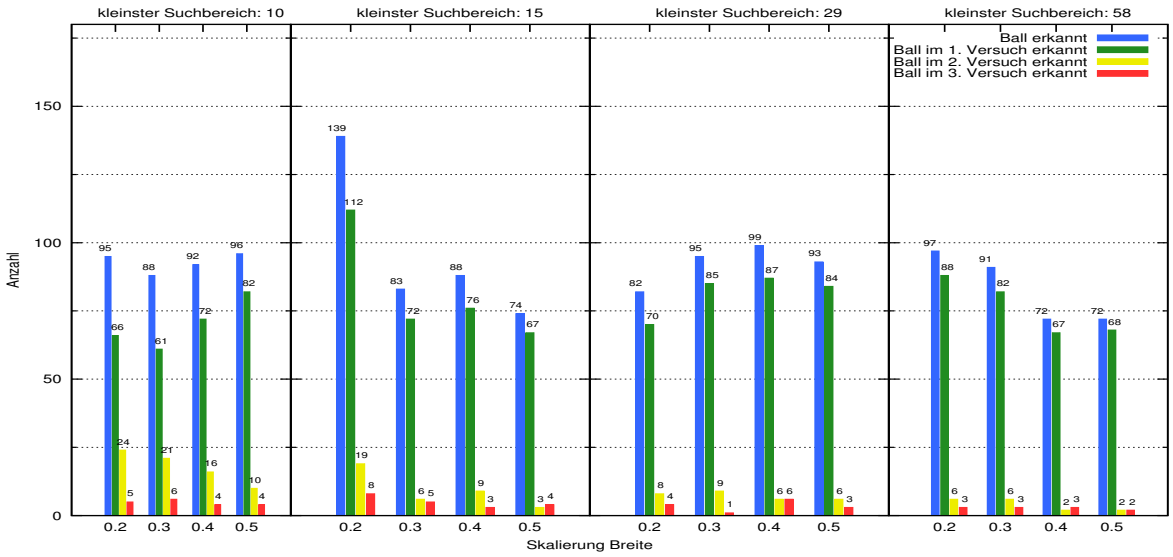
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.5)

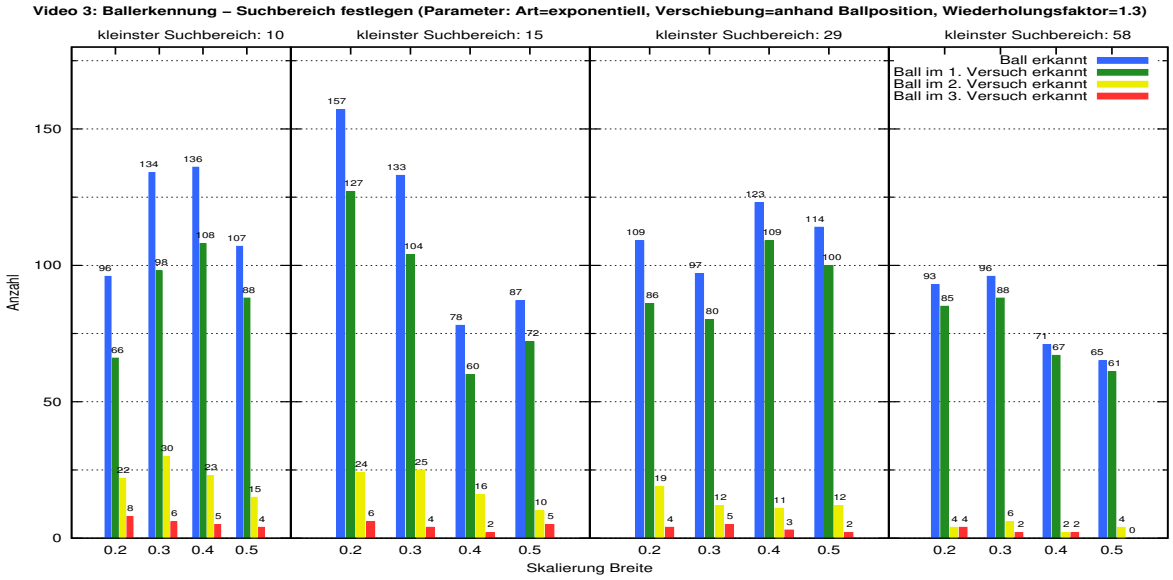


Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.7)



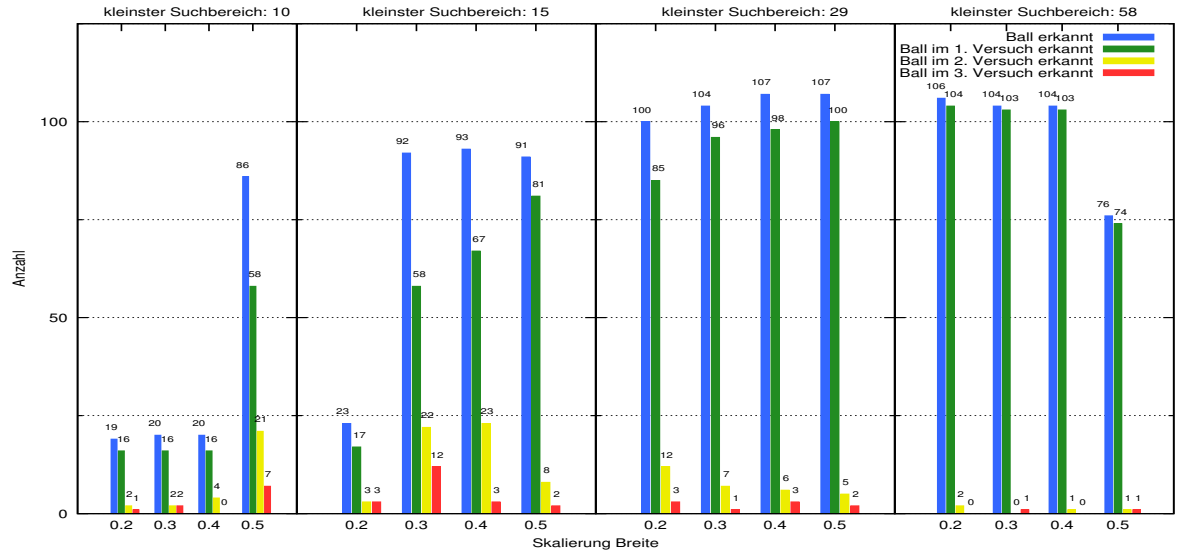
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9)



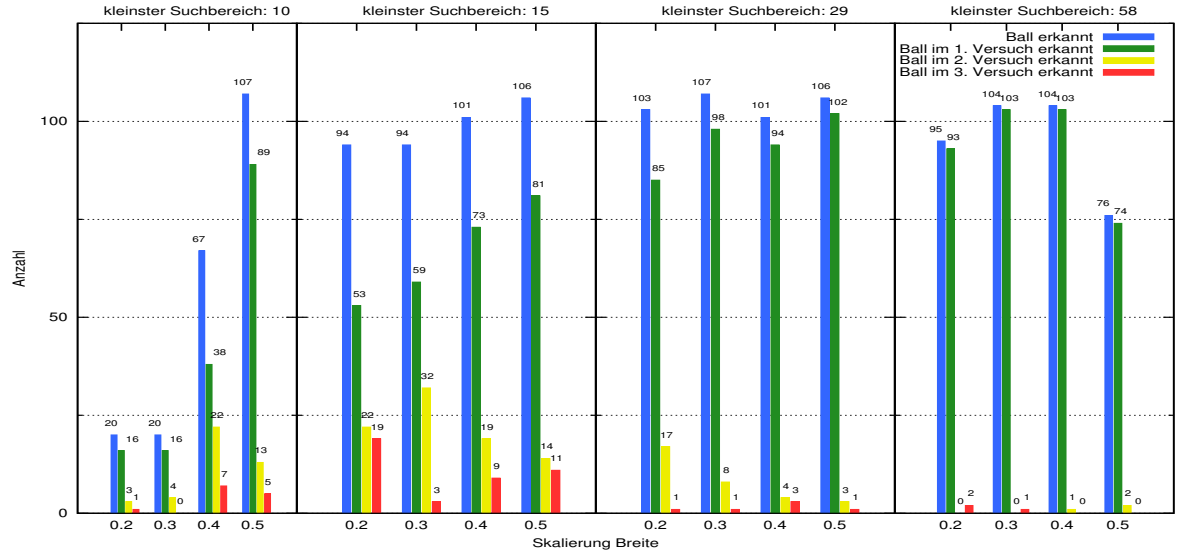

Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3)
kleinster Suchbereich: 10
kleinster Suchbereich: 15
kleinster Suchbereich: 29
kleinster Suchbereich: 58
Ball erkannt
Ball im 1. Versuch erkannt
Ball im 2. Versuch erkannt
Ball im 3. Versuch erkannt
Anzahl
Skalierung Breite
0.2
0.3
0.4
0.5
0
50
100
150
96 66 22 8
134 98 30 6
136 108 23 5
107 88 15 4
157 127 24 6
133 104 25 4
78 60 16 2
87 72 10 5
109 86 19 4
97 80 12 5
123 109 11 3
114 100 12 2
93 85 4 4
96 88 6 2
71 67 2 2
65 61 4 0

A.1.2.4. Parameter: Art = exponentiell, Verschiebung = anhand Kalman Filter

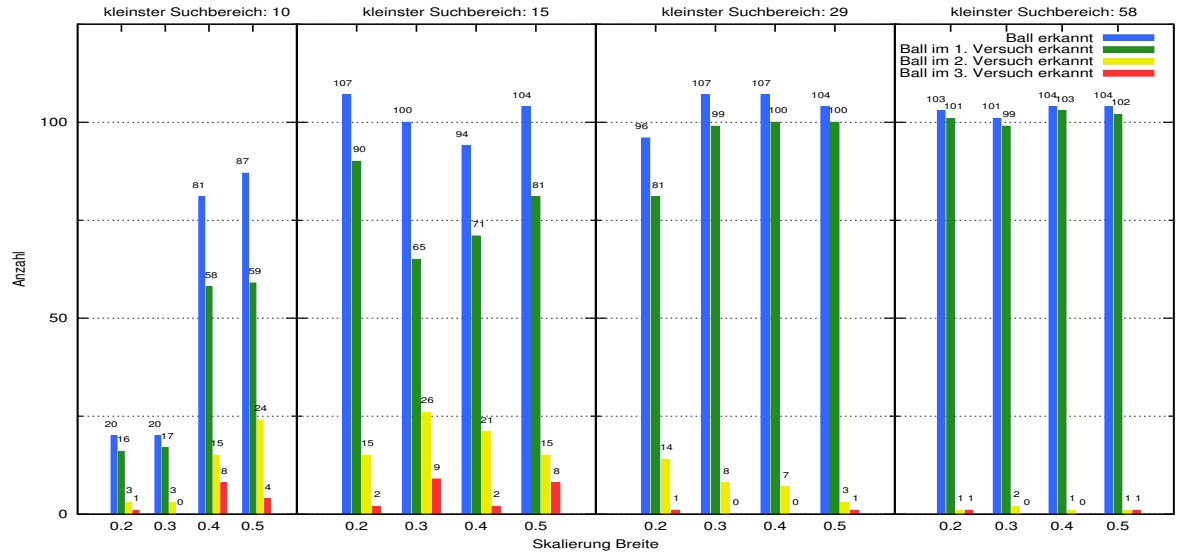
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.5)



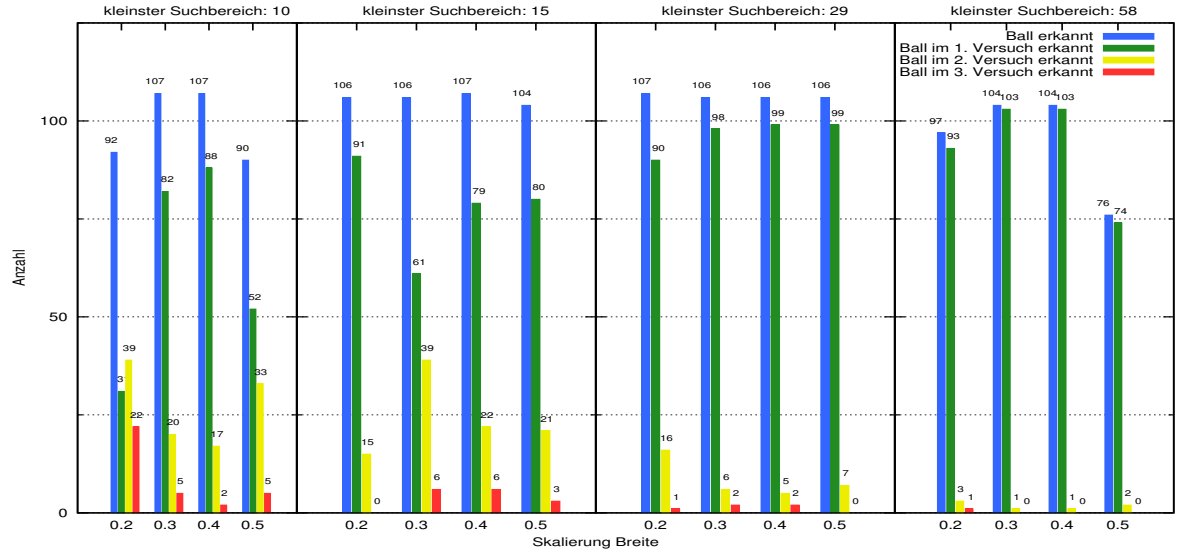
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.7)



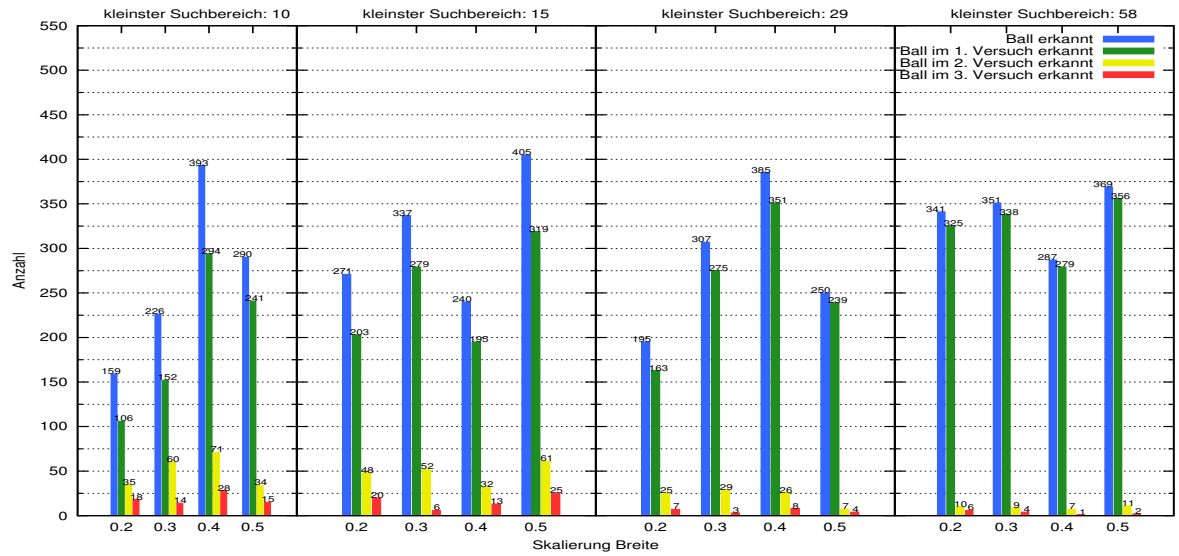
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9)



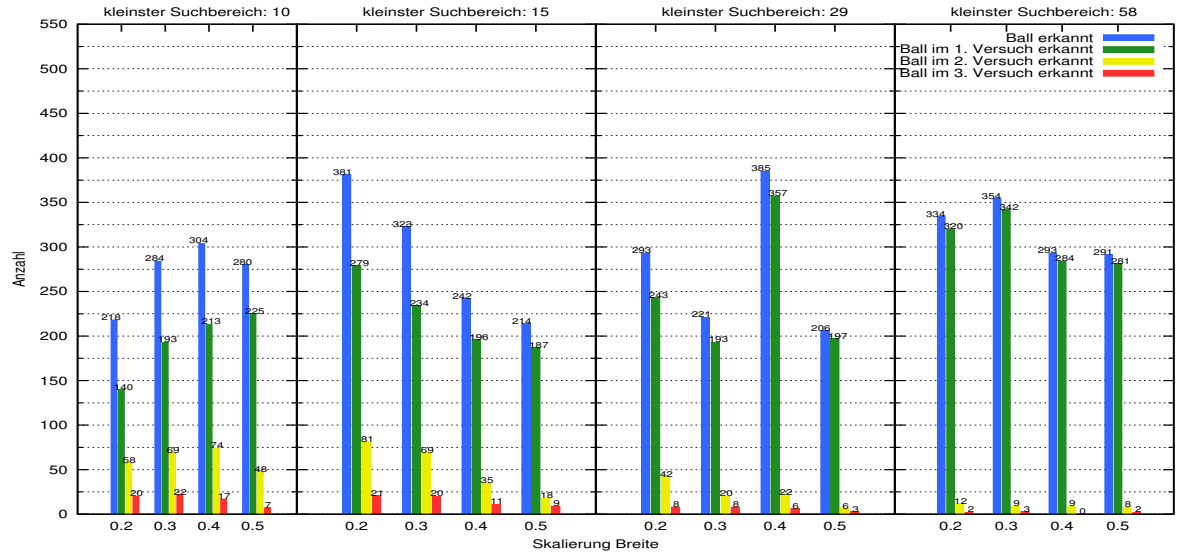
Video 1: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3)



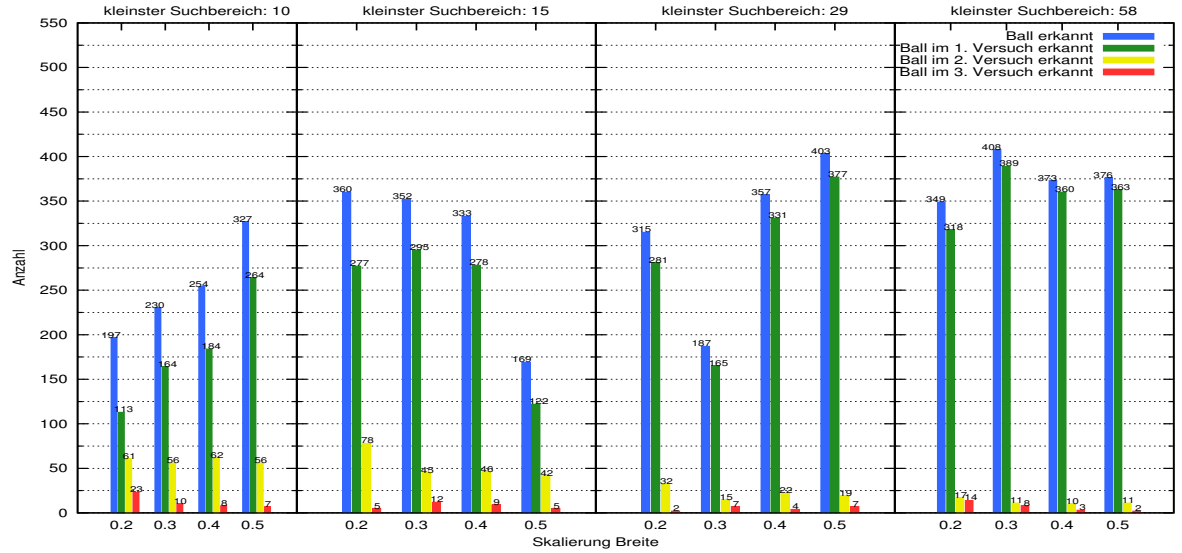
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.5)



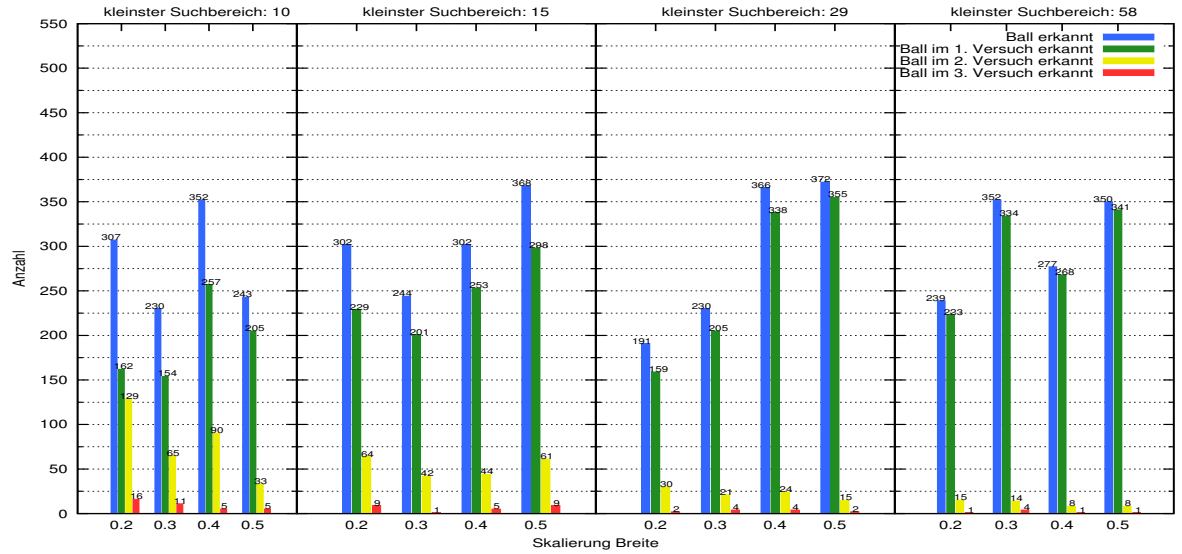
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.7)



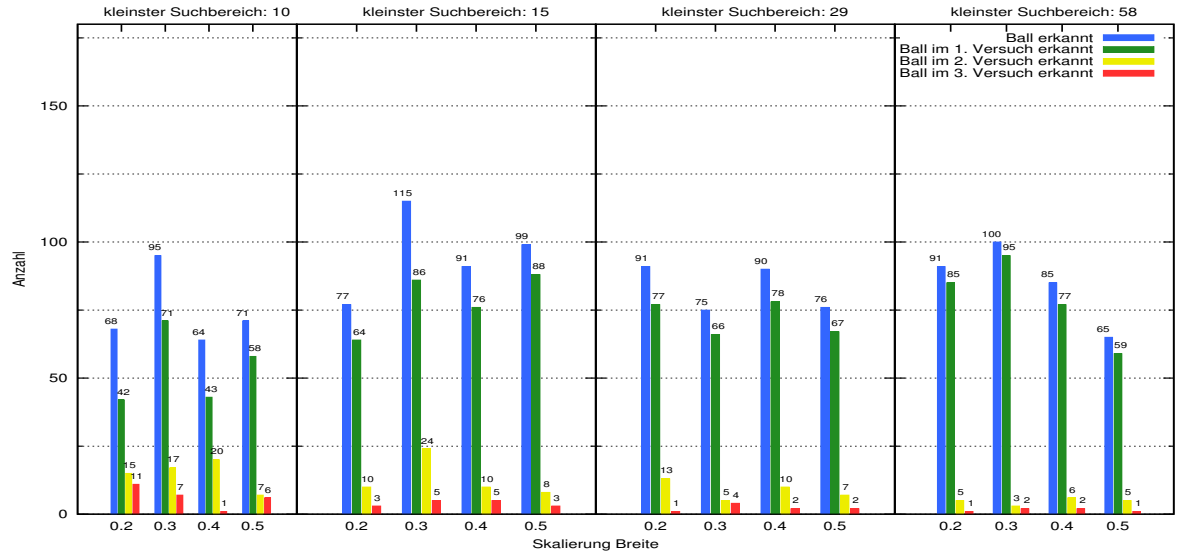
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9)



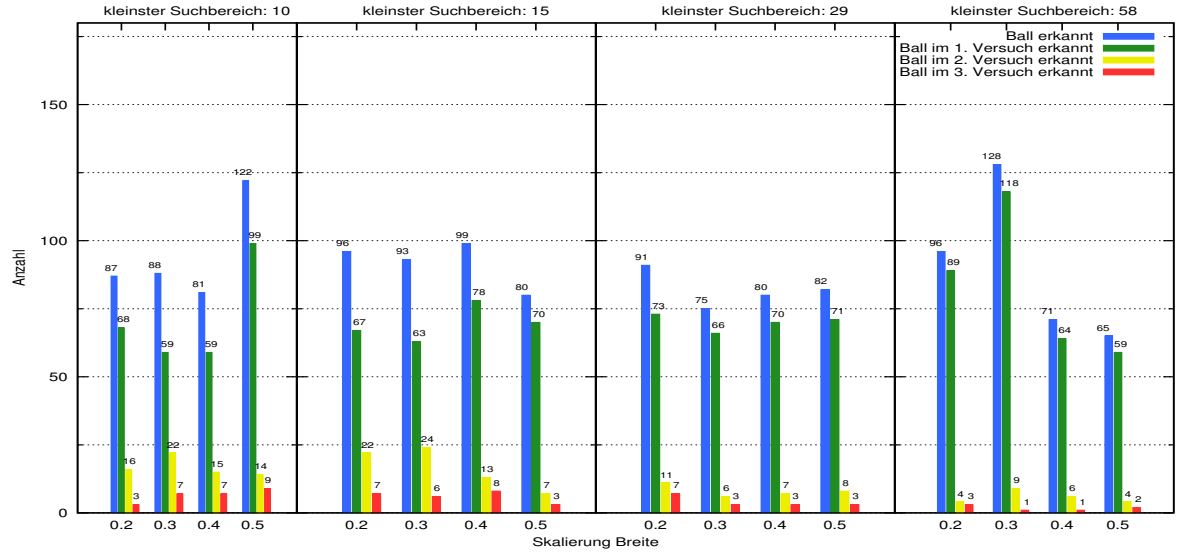
Video 2: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3)



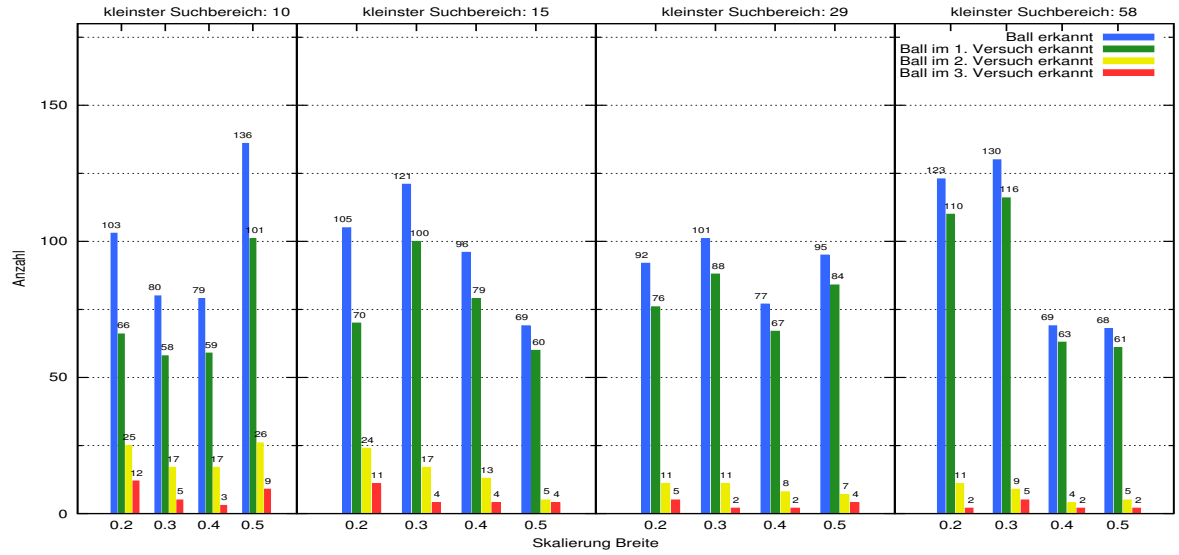
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.5)



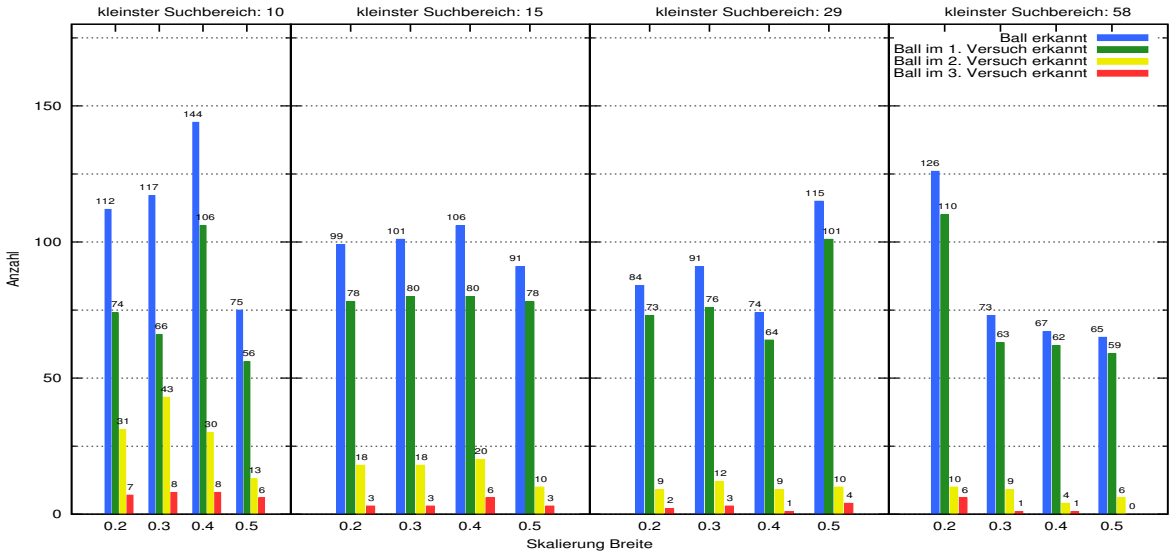
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.7)



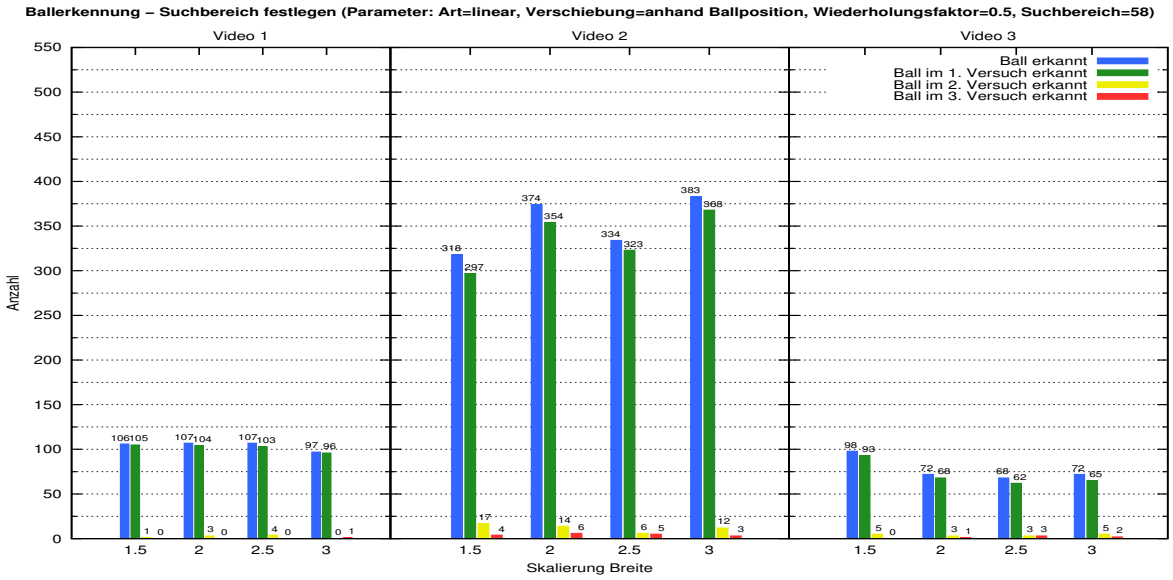
Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9)

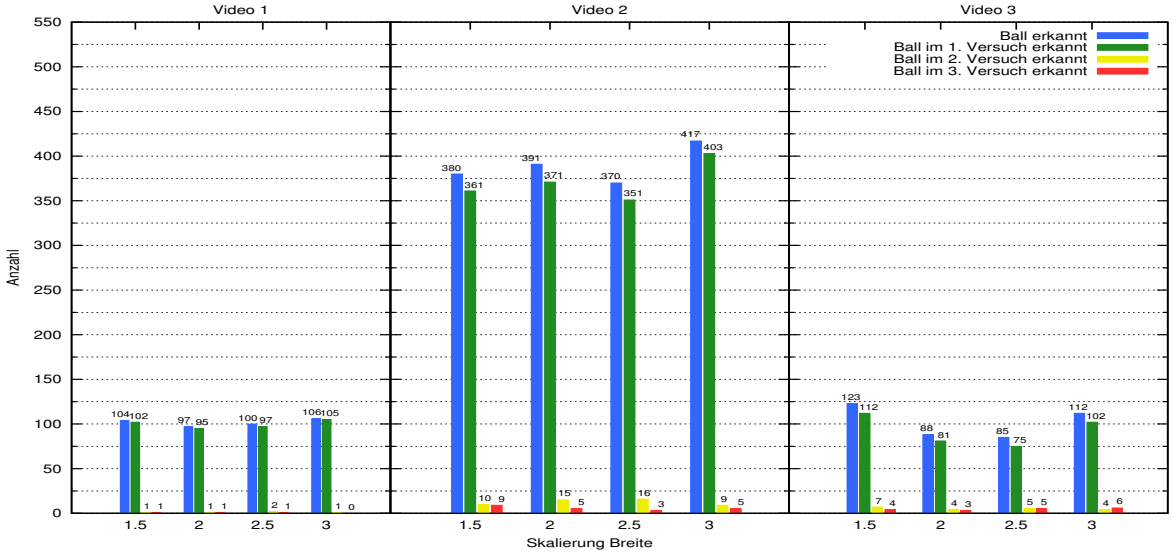


Video 3: Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3)

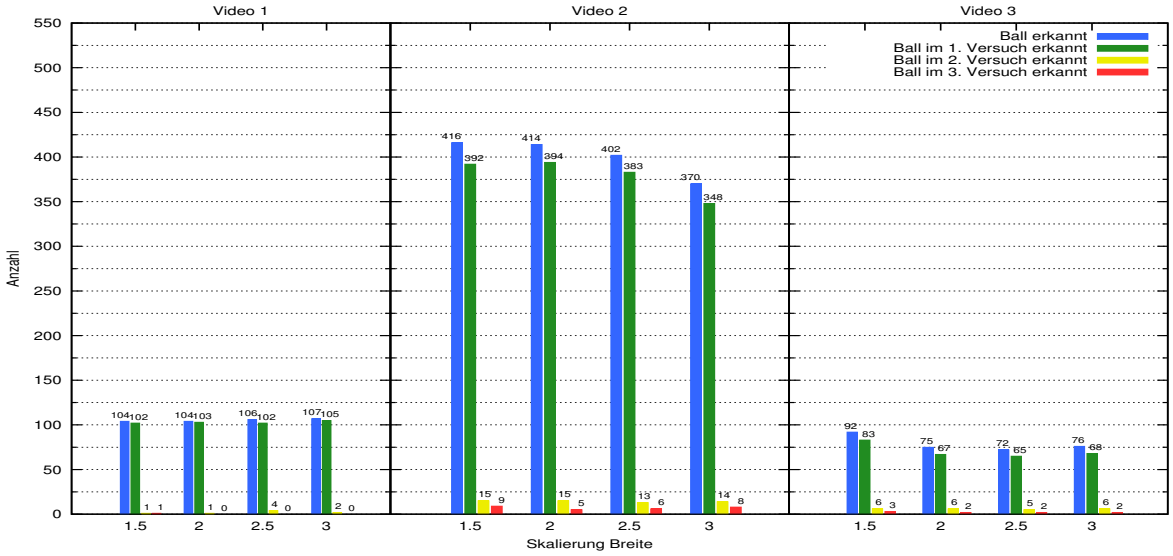


A.1.2.5. Analyse des Suchbereichs 58

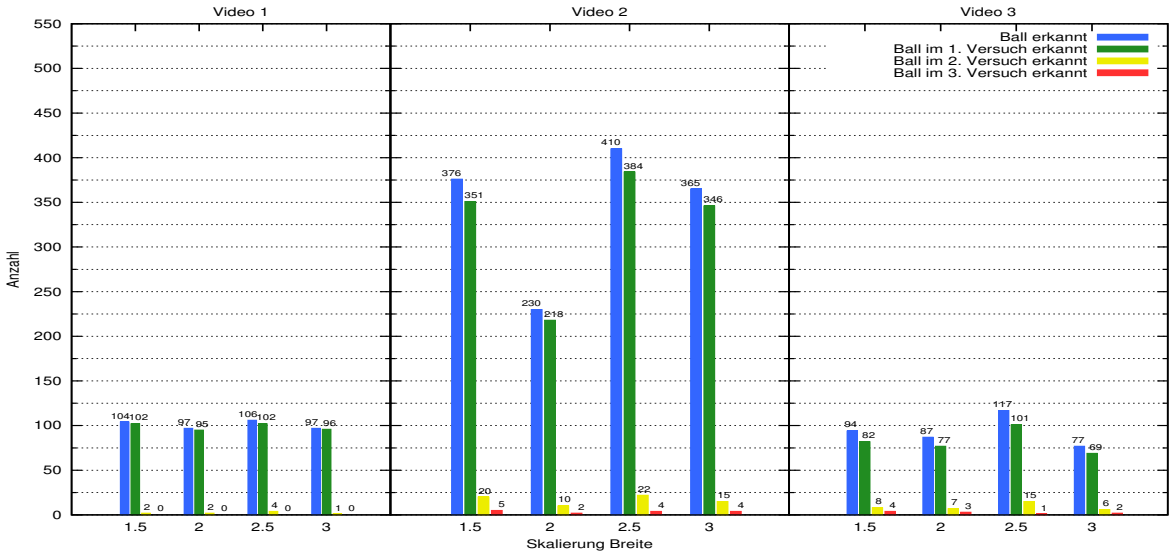




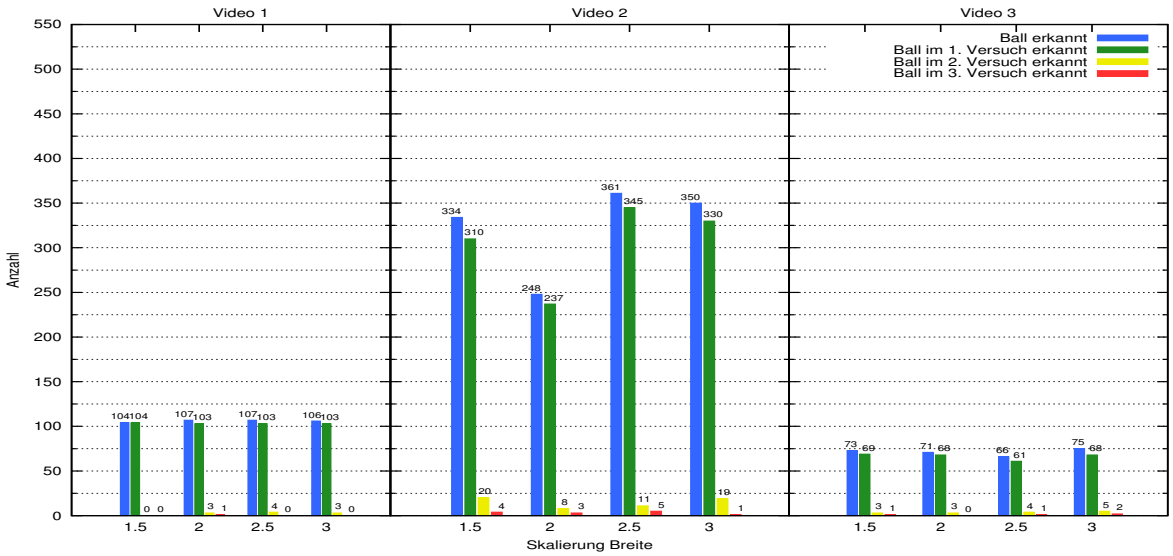
Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9, Suchbereich=58)

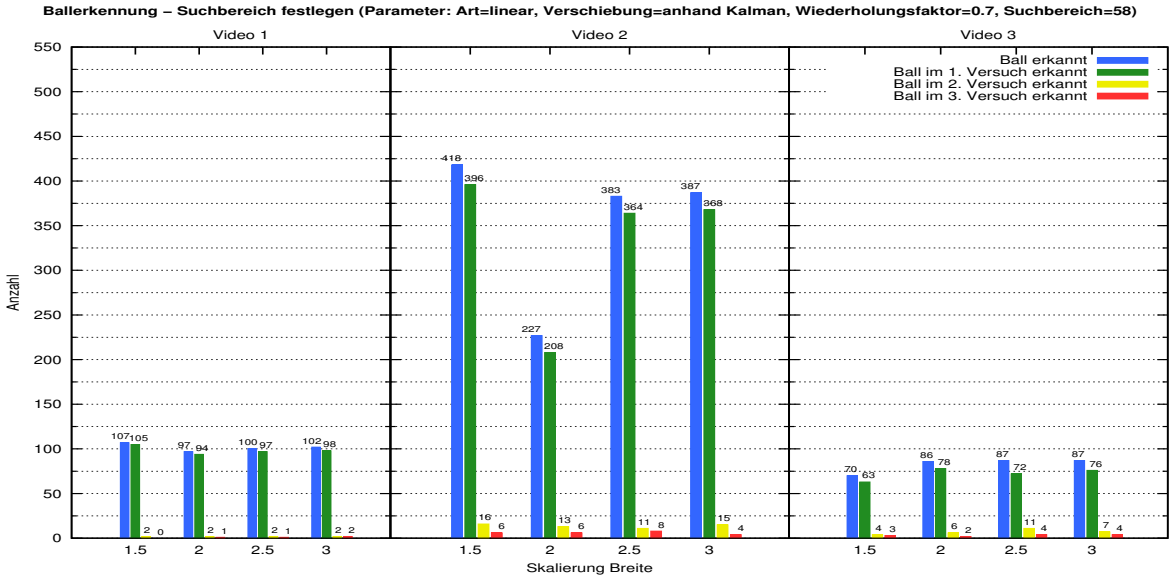


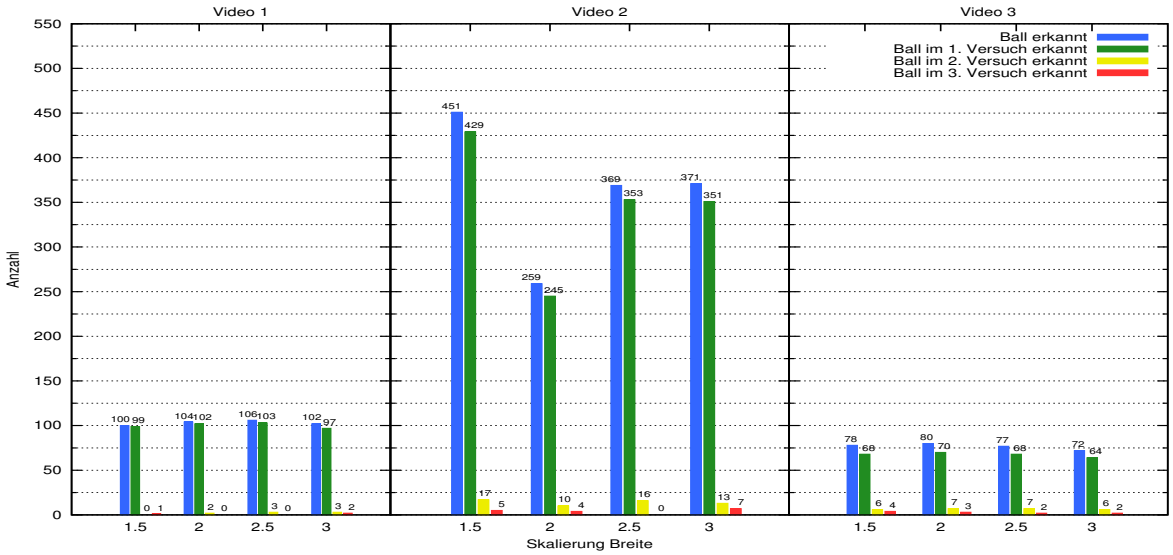
Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3, Suchbereich=58)

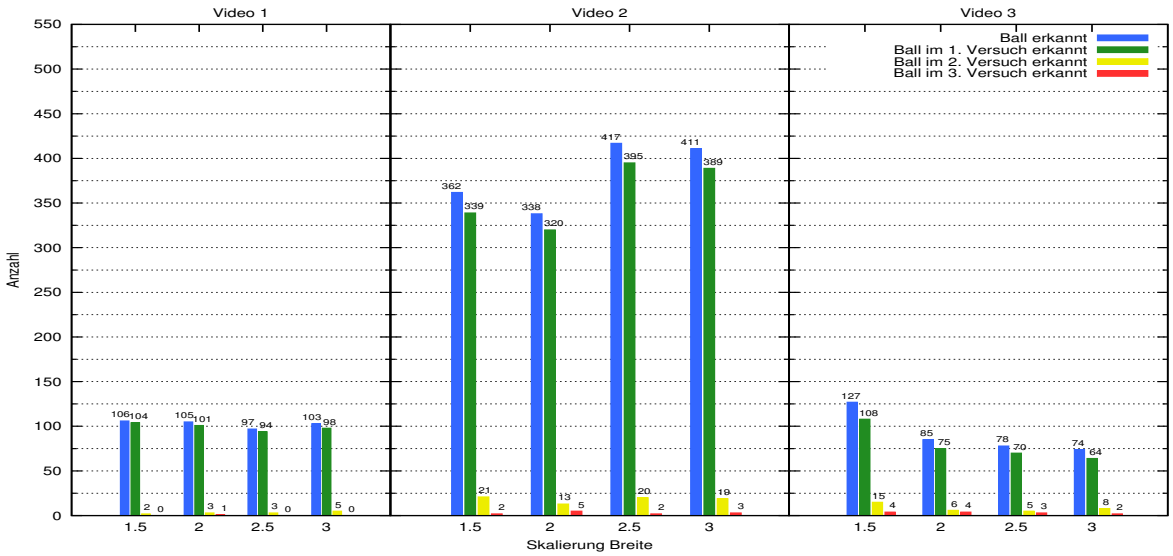


Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.5, Suchbereich=58)

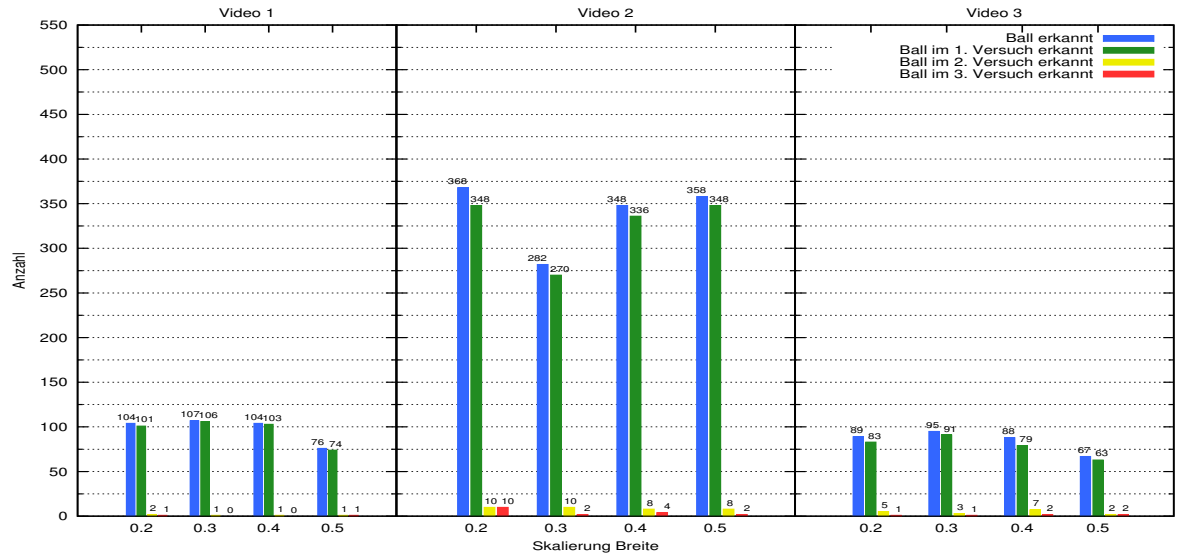



Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.7, Suchbereich=58)
Video 1
Video 2
Video 3
Ball erkannt
Ball im 1. Versuch erkannt
Ball im 2. Versuch erkannt
Ball im 3. Versuch erkannt
Anzahl
Skalierung Breite

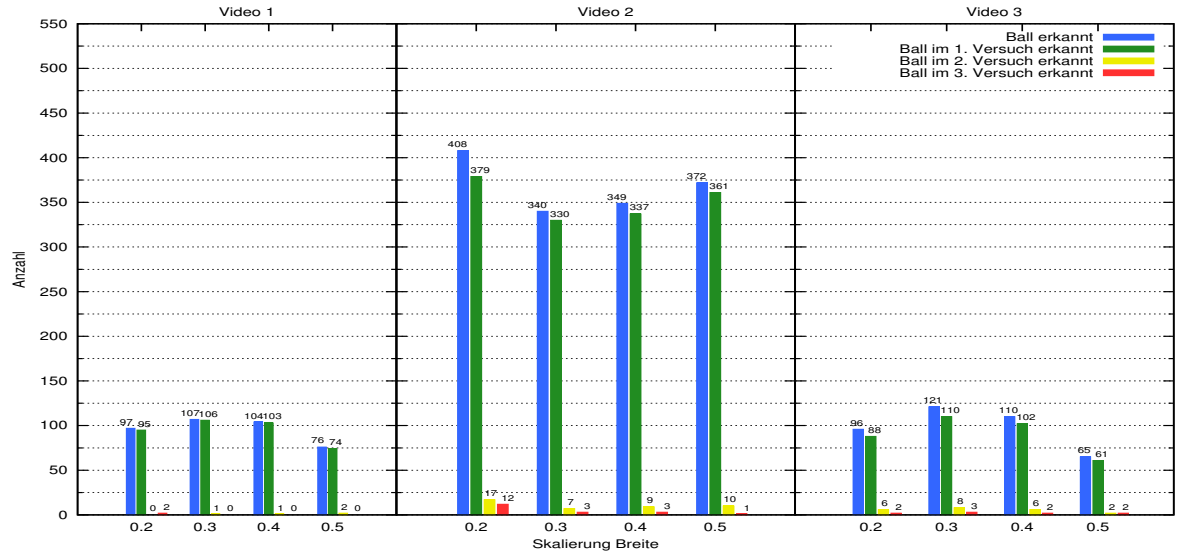

Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9, Suchbereich=58)
Video 1
Video 2
Video 3
Ball erkannt
Ball im 1. Versuch erkannt
Ball im 2. Versuch erkannt
Ball im 3. Versuch erkannt
Anzahl
Skalierung Breite


Ballerkennung – Suchbereich festlegen (Parameter: Art=linear, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3, Suchbereich=58)
Video 1
Video 2
Video 3
Ball erkannt
Ball im 1. Versuch erkannt
Ball im 2. Versuch erkannt
Ball im 3. Versuch erkannt
Anzahl
Skalierung Breite

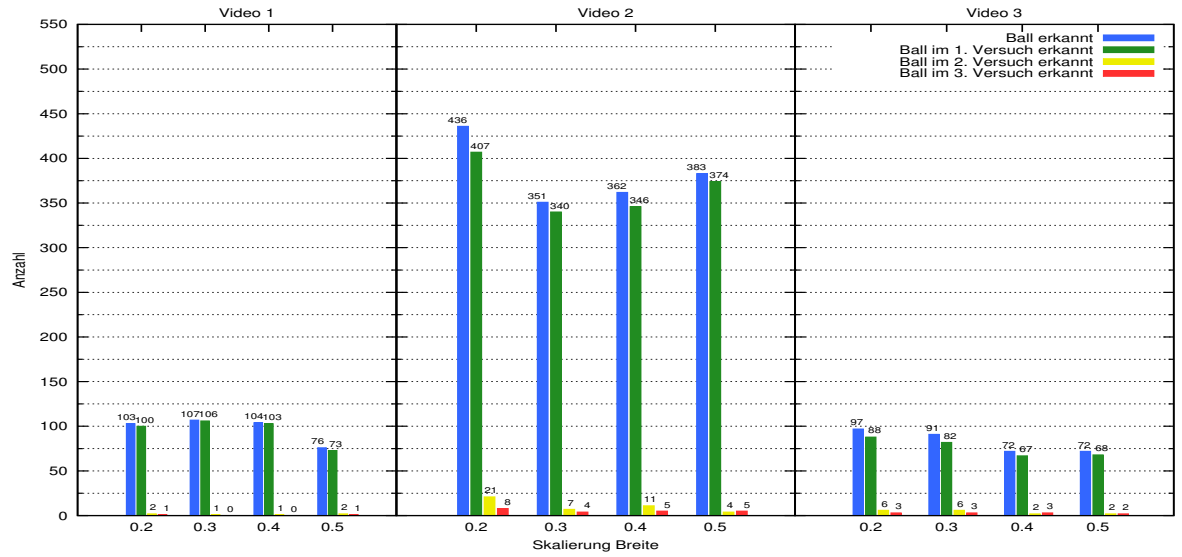
Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.5, Suchbereich=58)



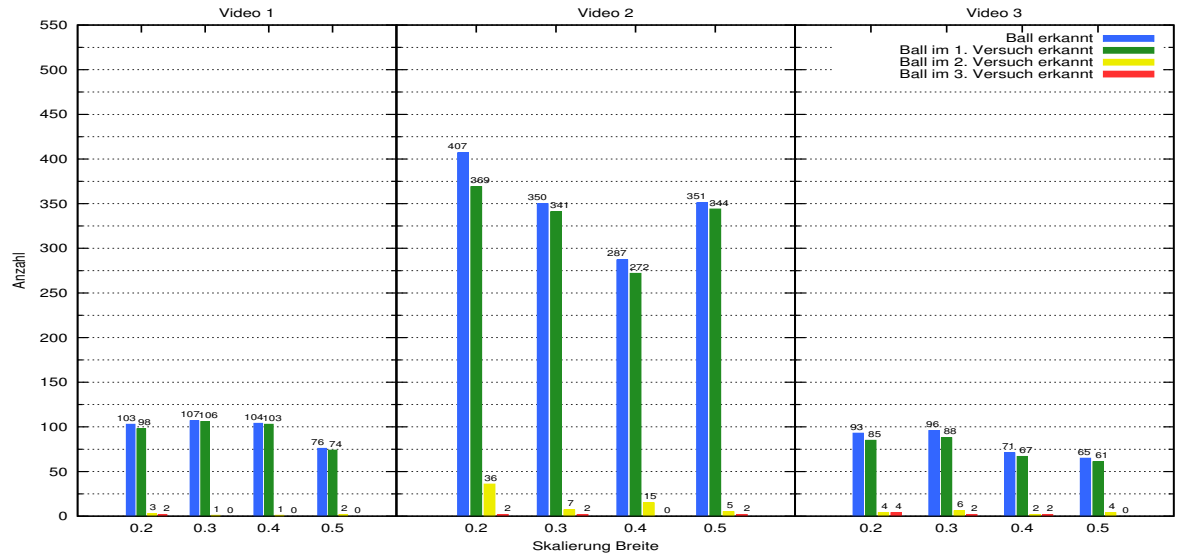
Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.7, Suchbereich=58)



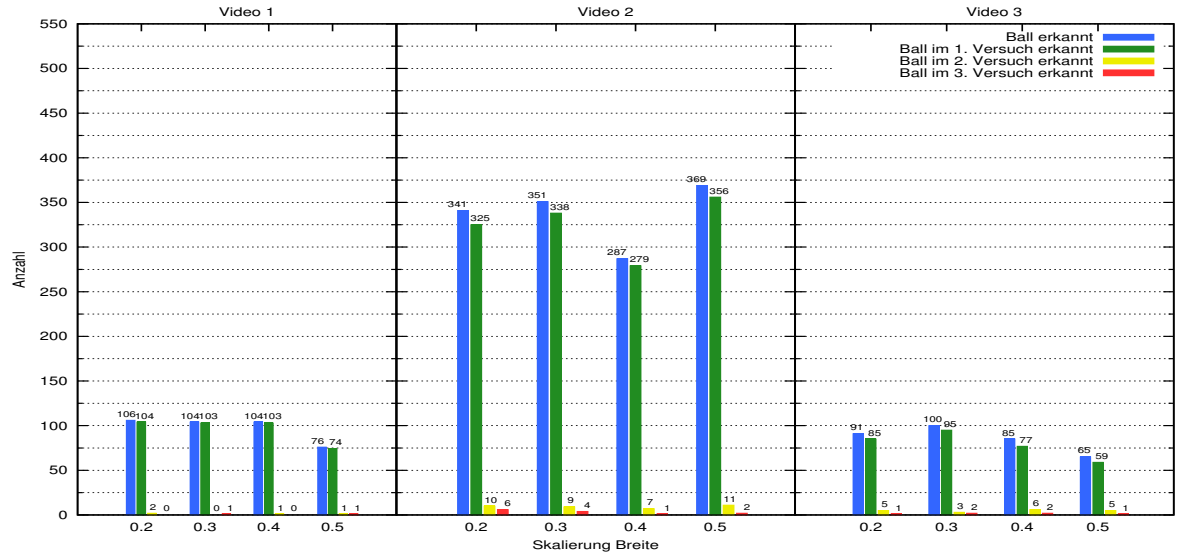
Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=0.9, Suchbereich=58)



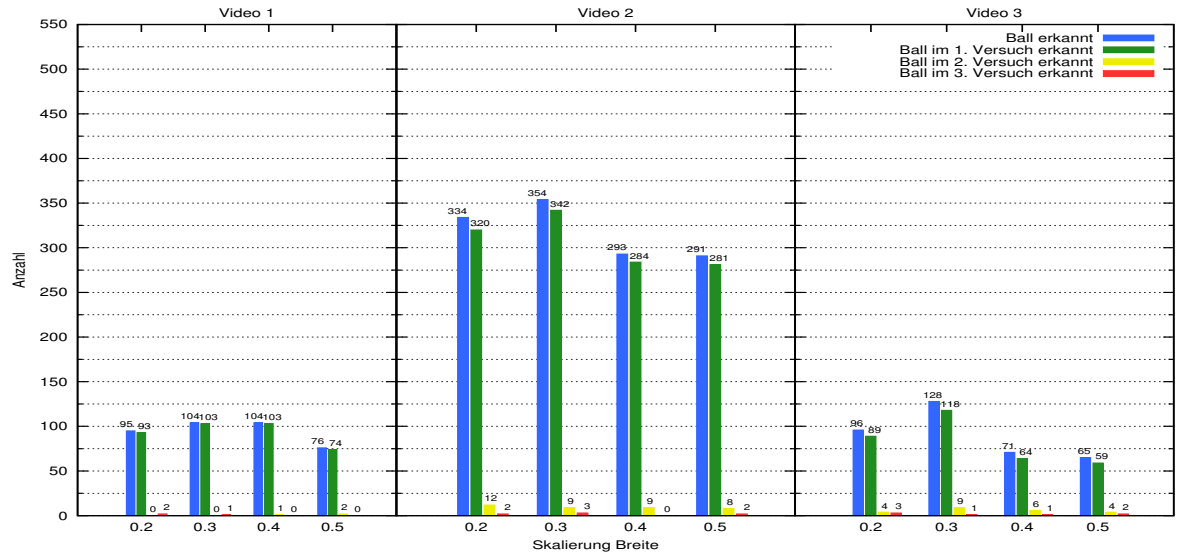
Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Ballposition, Wiederholungsfaktor=1.3, Suchbereich=58)



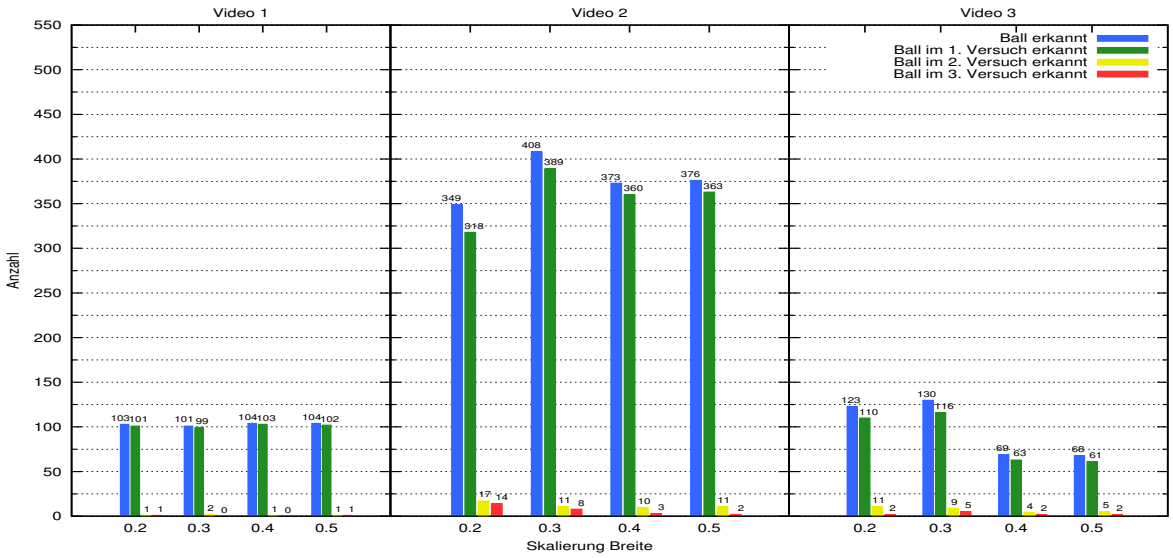
Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.5, Suchbereich=58)



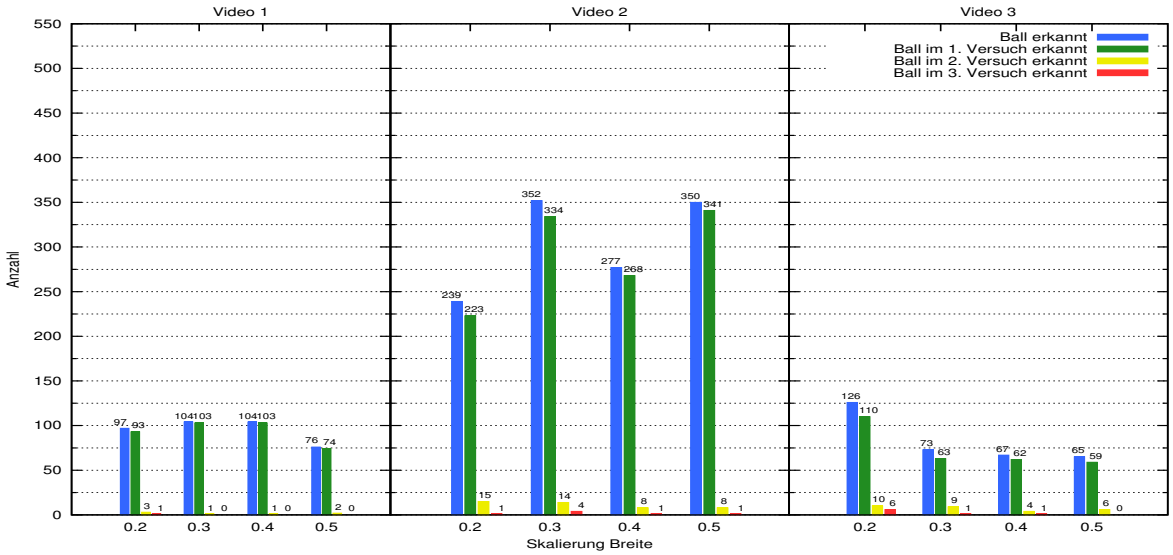
Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.7, Suchbereich=58)



Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=0.9, Suchbereich=58)

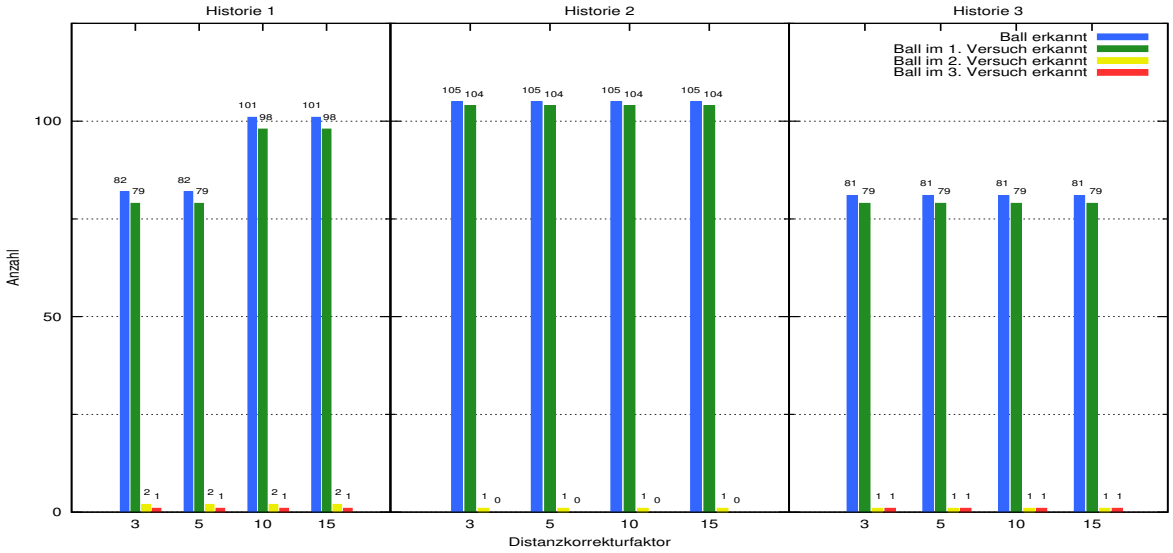


Ballerkennung – Suchbereich festlegen (Parameter: Art=exponentiell, Verschiebung=anhand Kalman, Wiederholungsfaktor=1.3, Suchbereich=58)

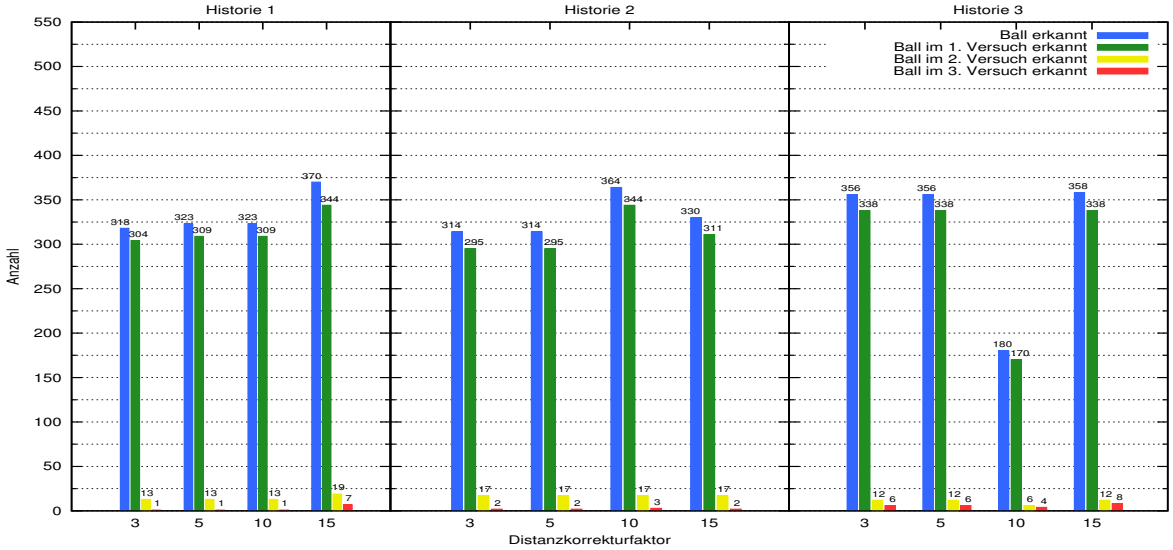


A.1.3. Analyse der Ballgröße und Distanzbestimmung

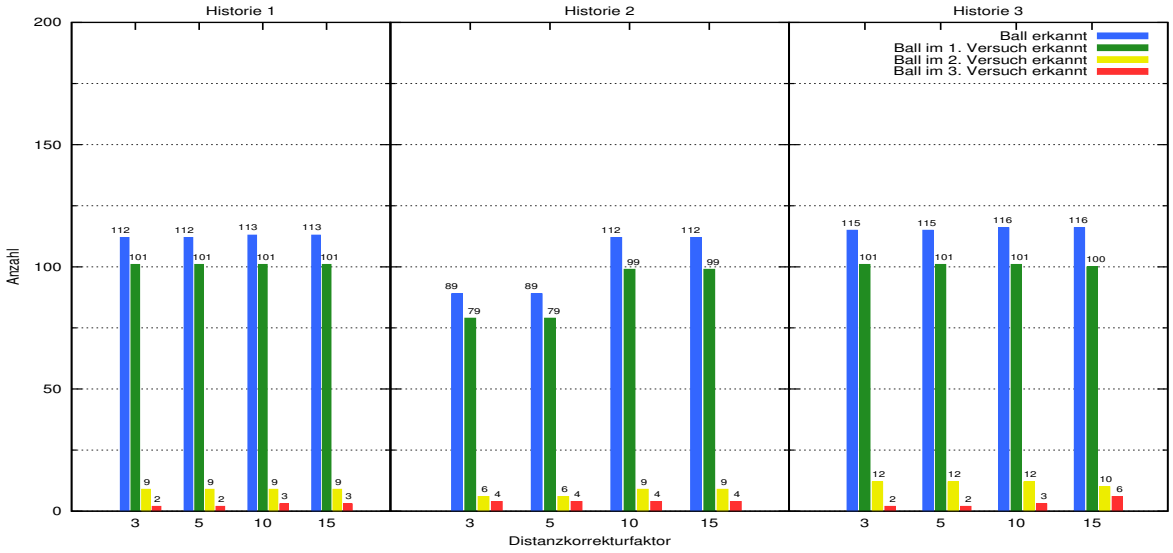
Video 1: Ballerkennung – Ermittlung der Ballgröße (Historie) und Distanz (Ermittlung durch Kalman Vorhersage)



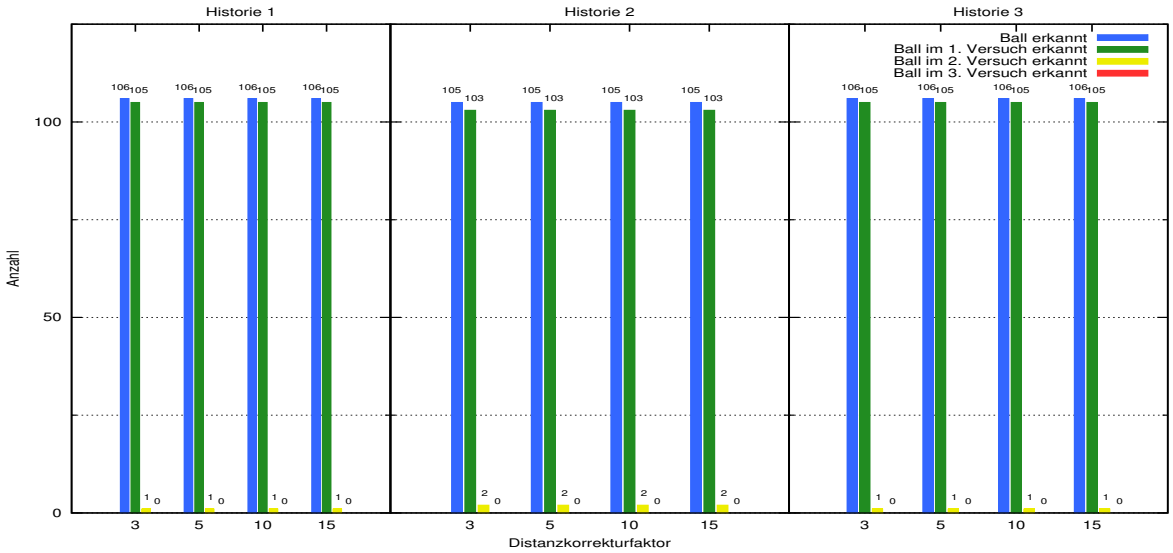
Video 2: Ballerkennung – Ermittlung der Ballgröße (Historie) und Distanz (Ermittlung durch Kalman Vorhersage)



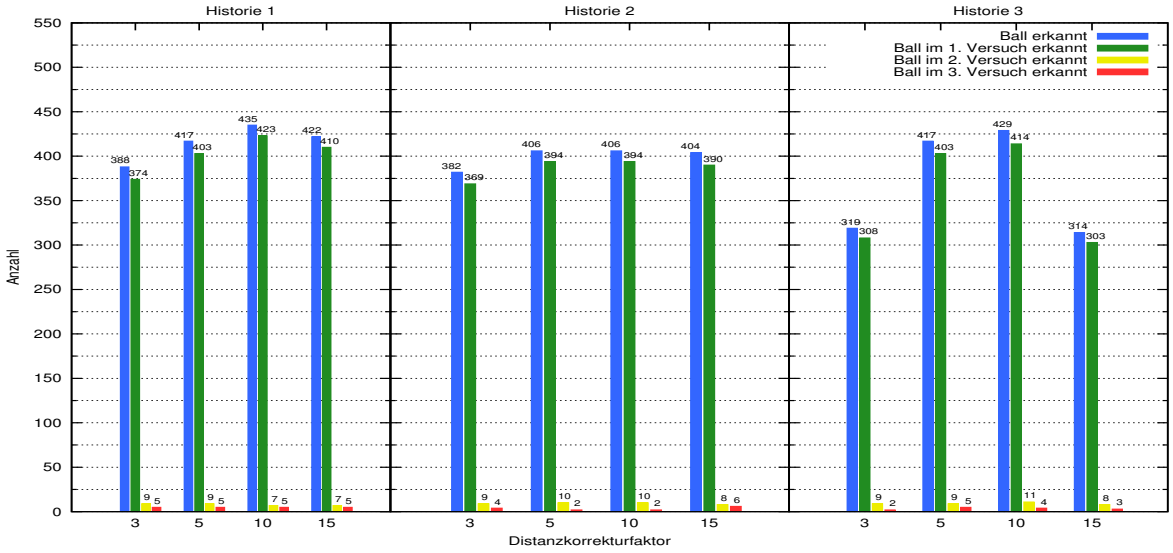
Video 3: Ballerkennung – Ermittlung der Ballgröße (Historie) und Distanz (Ermittlung durch Kalman Vorhersage)

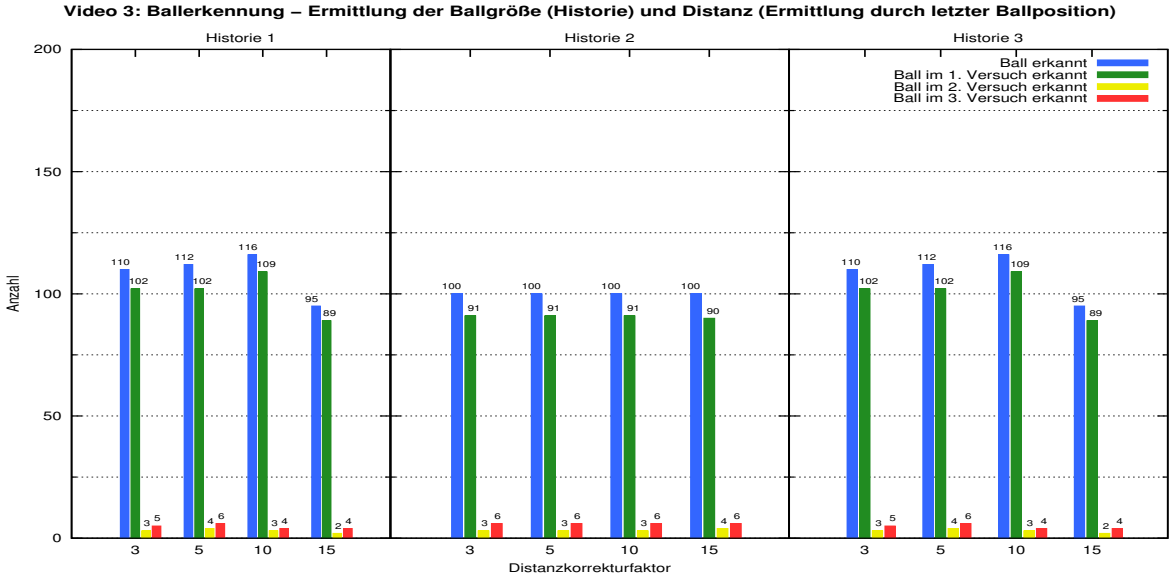


Video 1: Ballerkennung – Ermittlung der Ballgröße (Historie) und Distanz (Ermittlung durch letzter Ballposition)



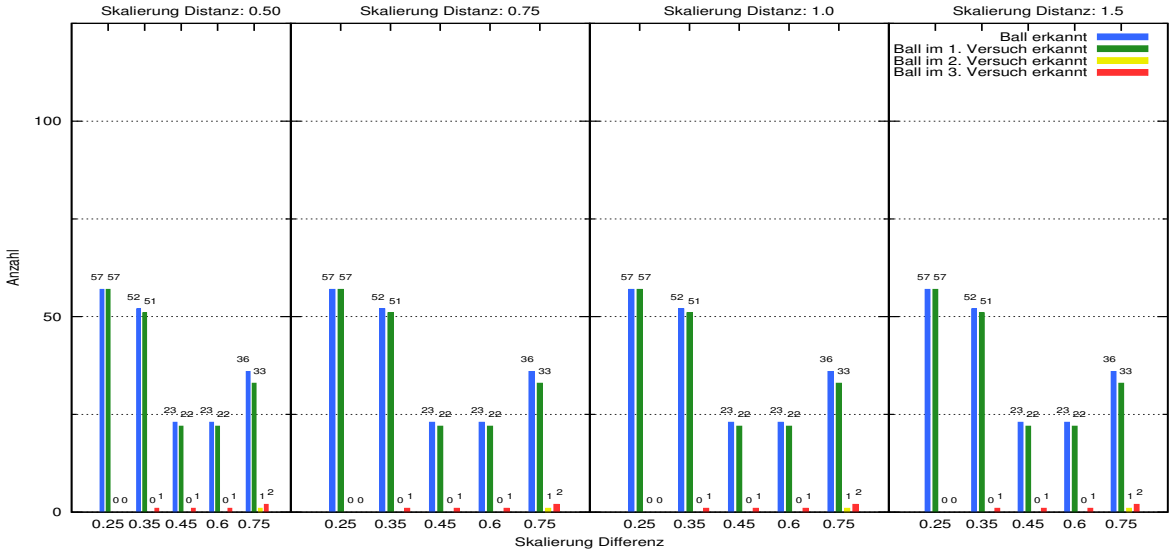
Video 2: Ballerkennung – Ermittlung der Ballgröße (Historie) und Distanz (Ermittlung durch letzter Ballposition)



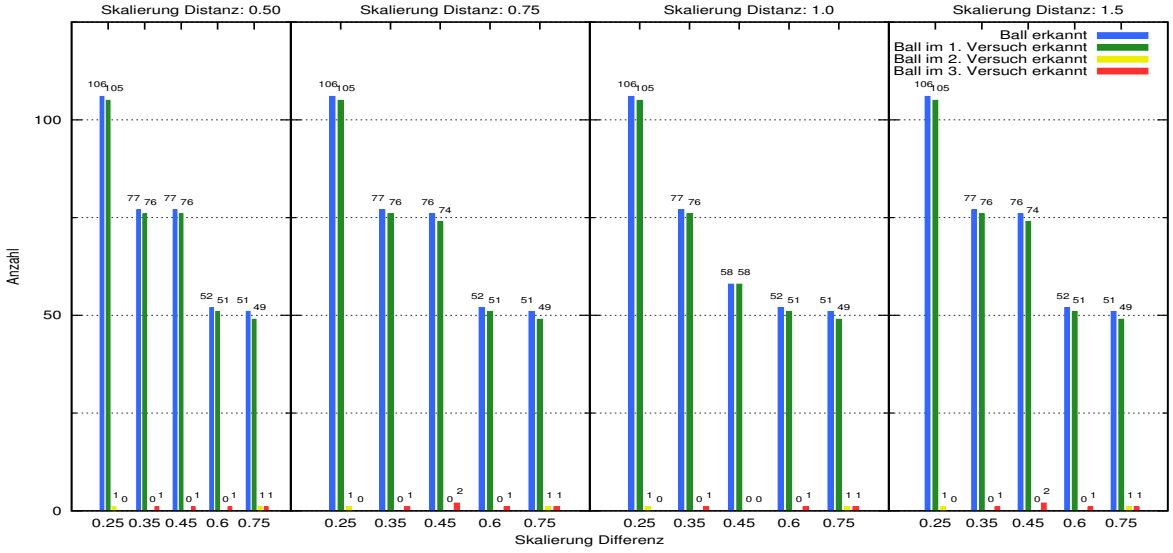

Video 3: Ballerkennung – Ermittlung der Ballgröße (Historie) und Distanz (Ermittlung durch letzter Ballposition)
Historie 1
Historie 2
Historie 3
Ball erkannt
Ball im 1. Versuch erkannt
Ball im 2. Versuch erkannt
Ball im 3. Versuch erkannt
Anzahl
Distanzkorrekturfaktor
200
150
100
50
0
3
5
10
15
110
102
3
5
112
102
4
6
116
109
3
4
95
89
2
4
100
91
3
6
100
91
3
6
100
91
3
6
100
90
4
6
110
102
3
5
112
102
4
6
116
109
3
4
95
89
2
4

A.1.4. Analyse der Parameter für die Ballidentifikation

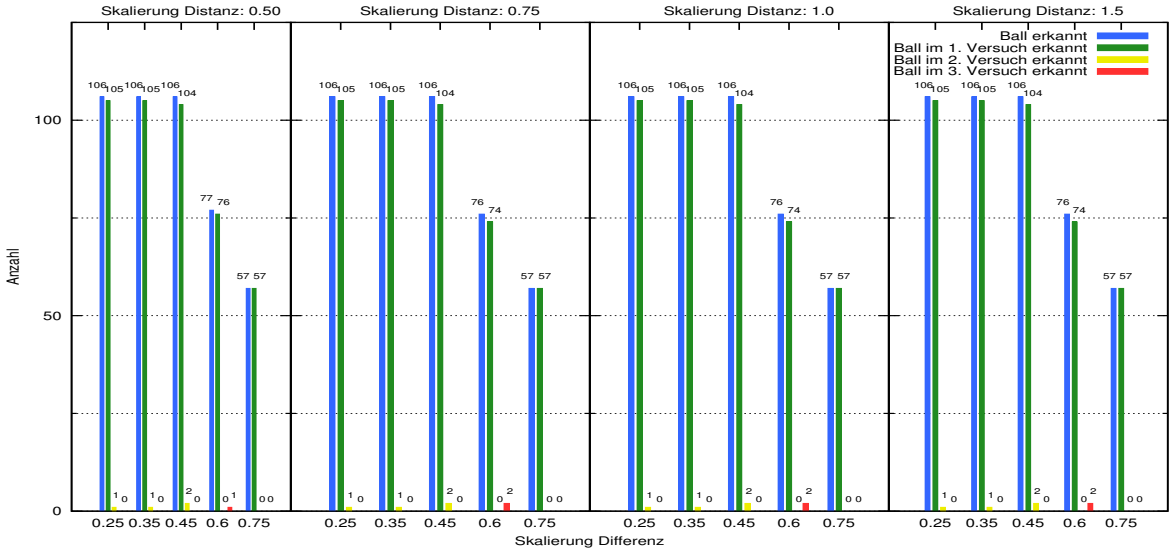
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für max. gültiger Abstand=7%)



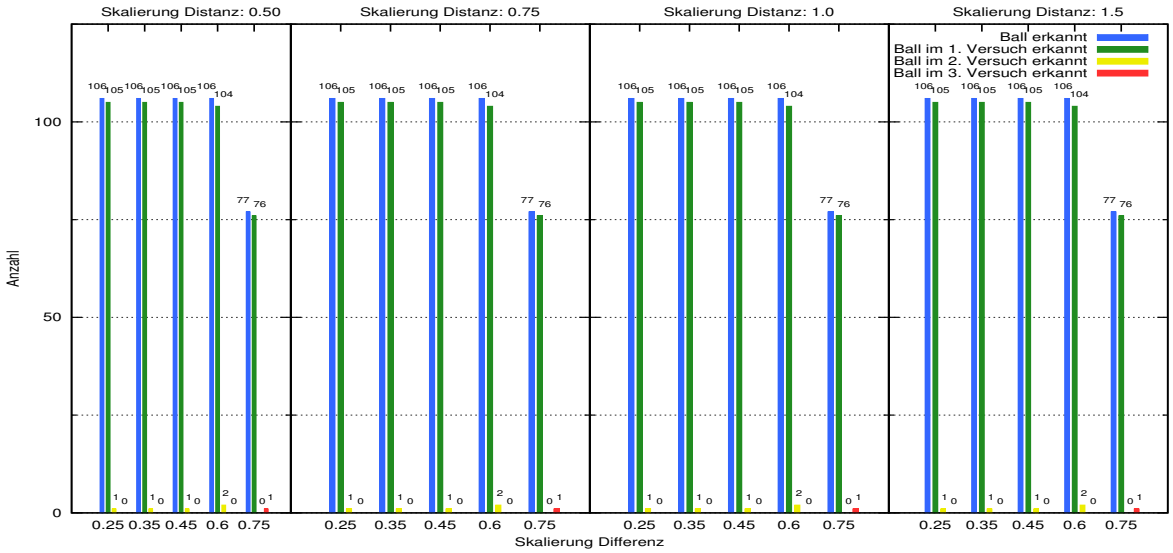
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=14%)



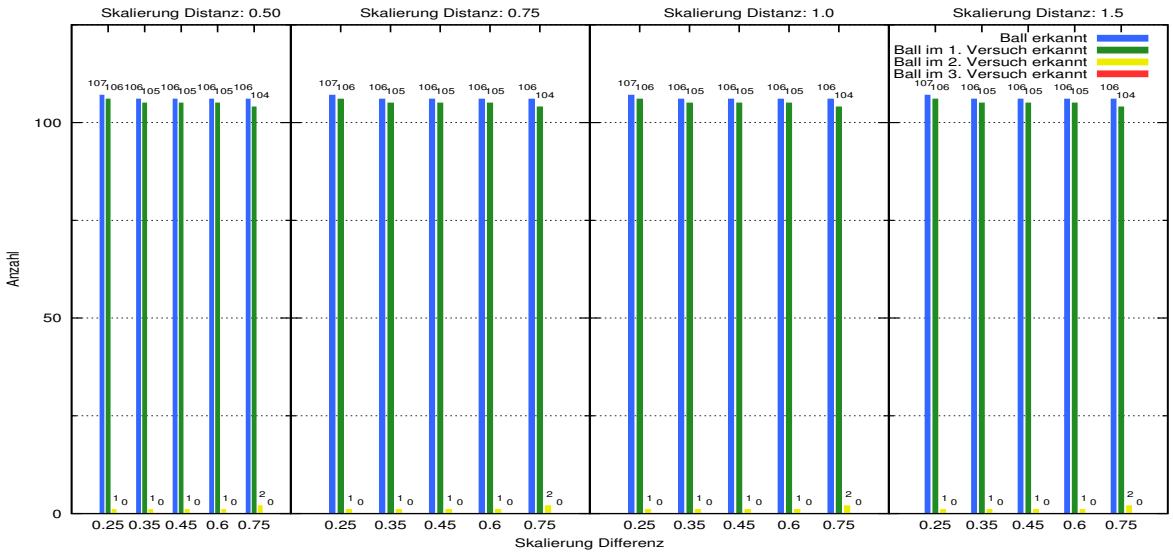
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=21%)



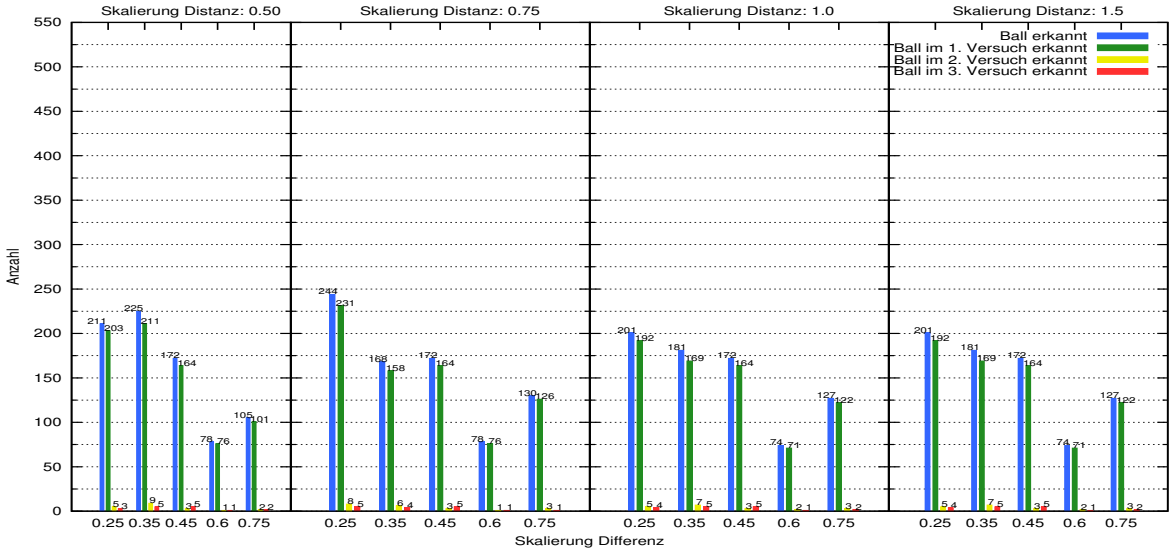
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=28%)



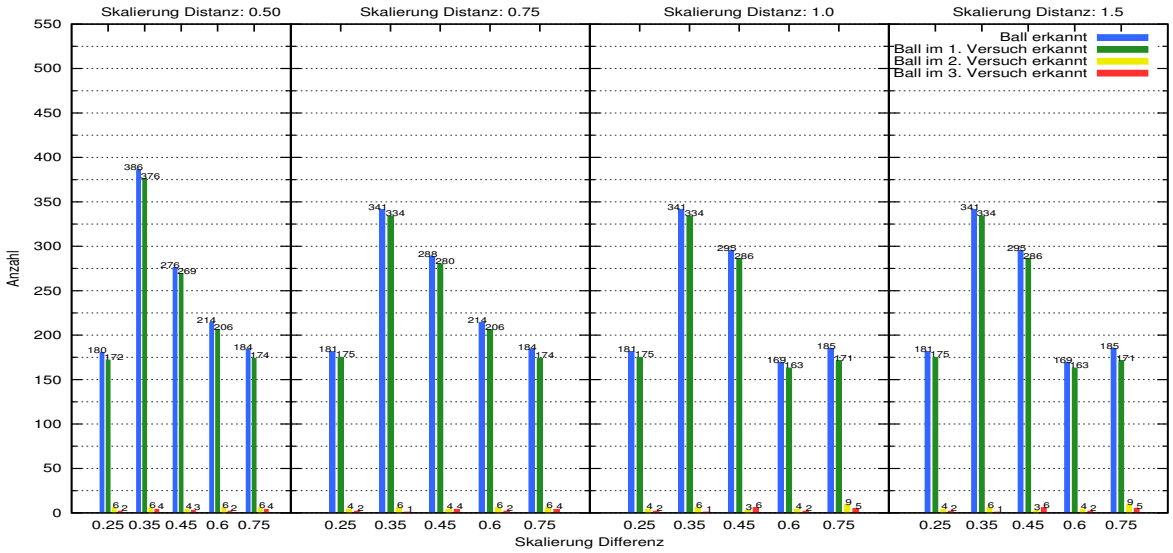
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=35%)



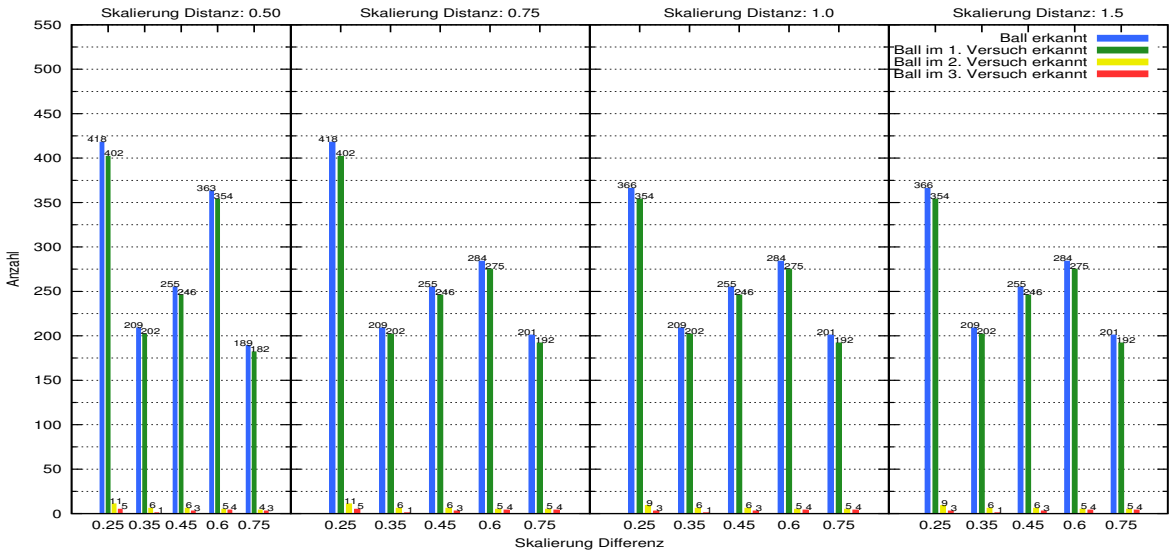
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=7%)



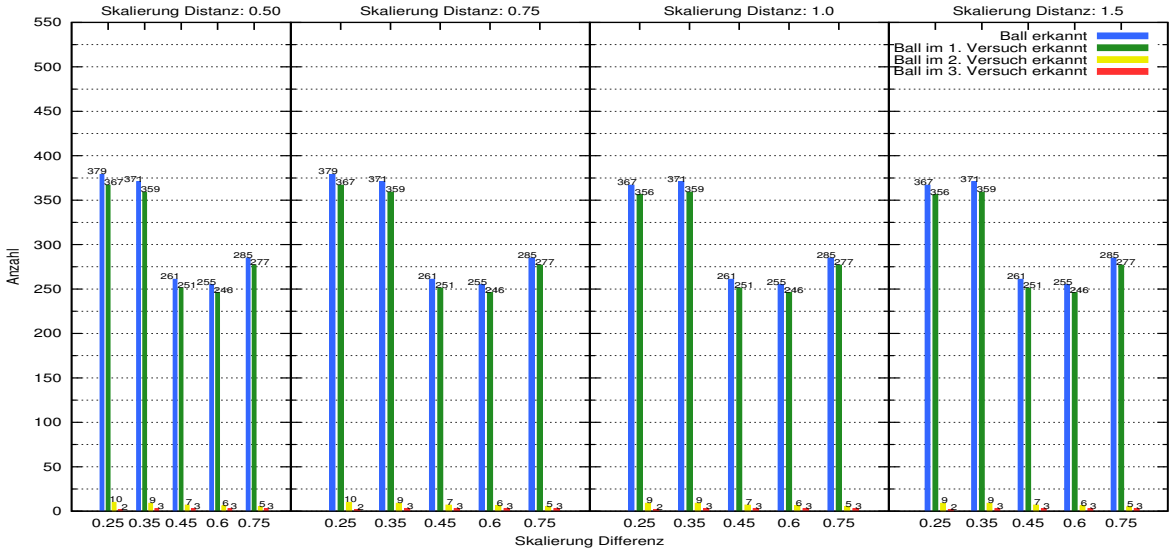
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=14%)



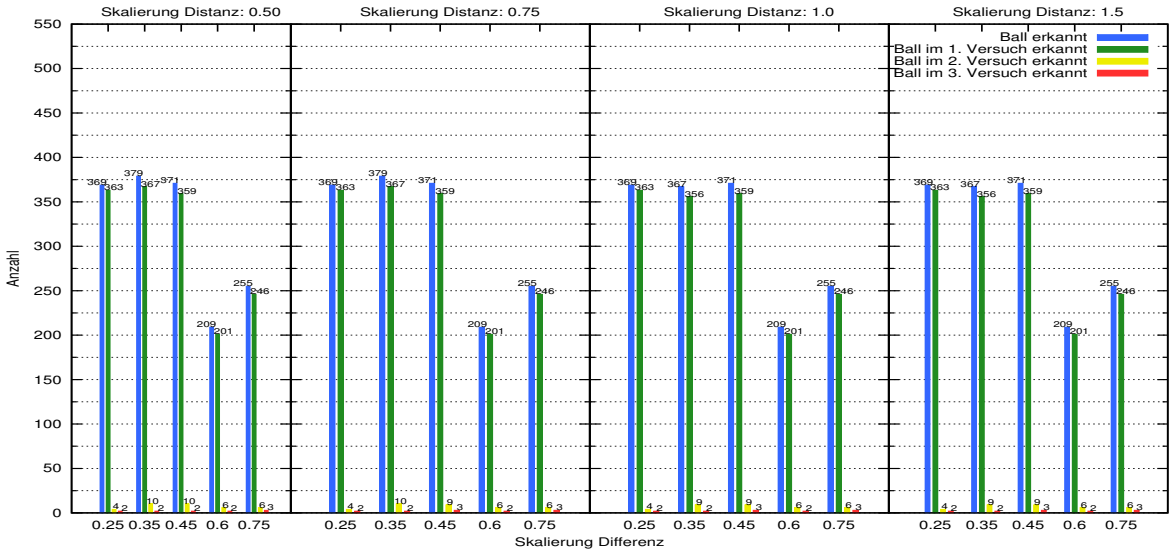
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=21%)



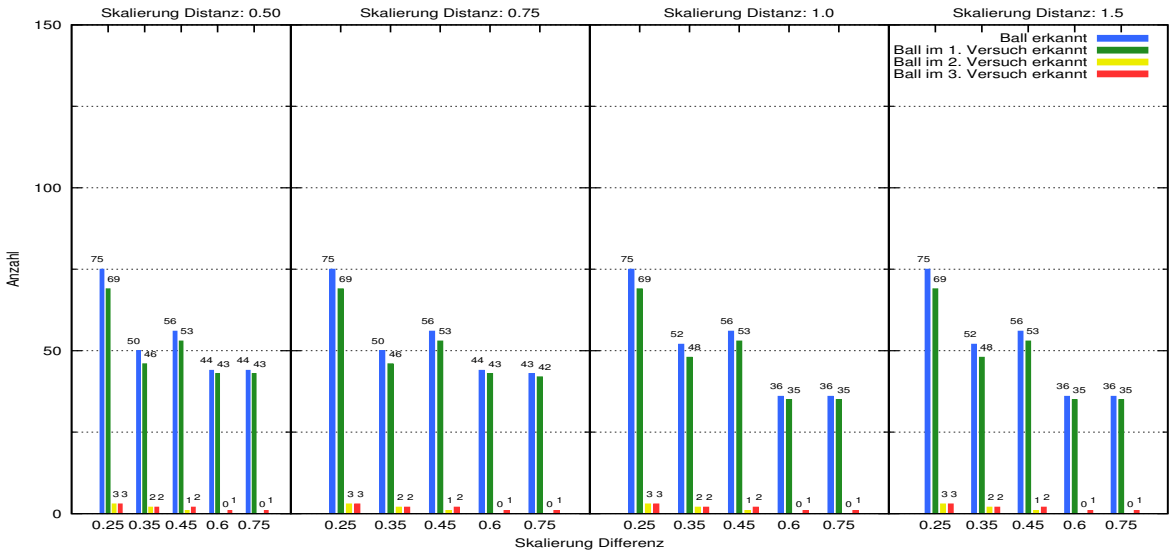
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=28%)



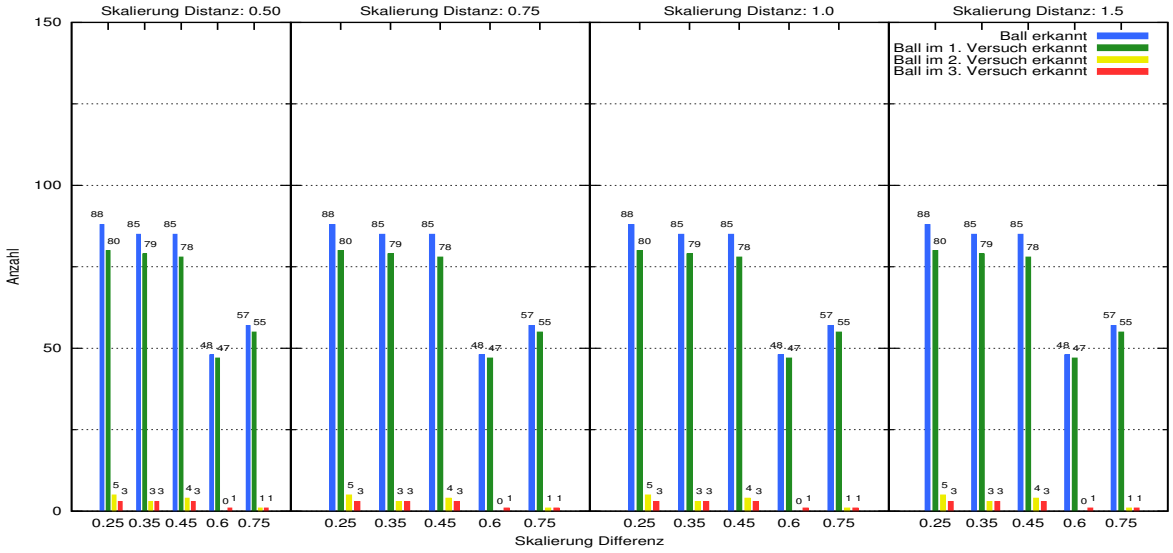
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=35%)



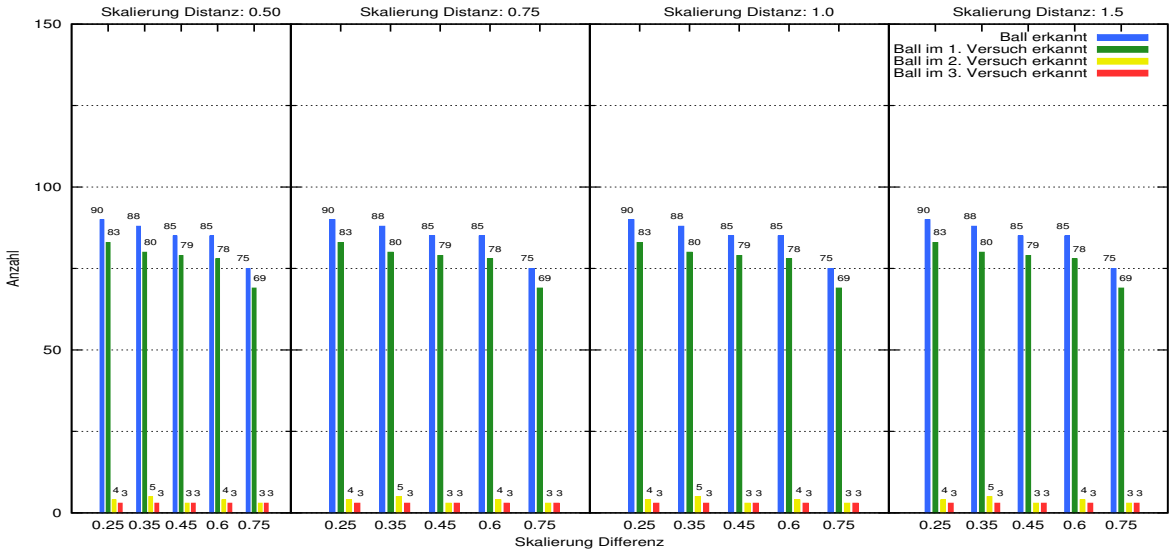
Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=7%)



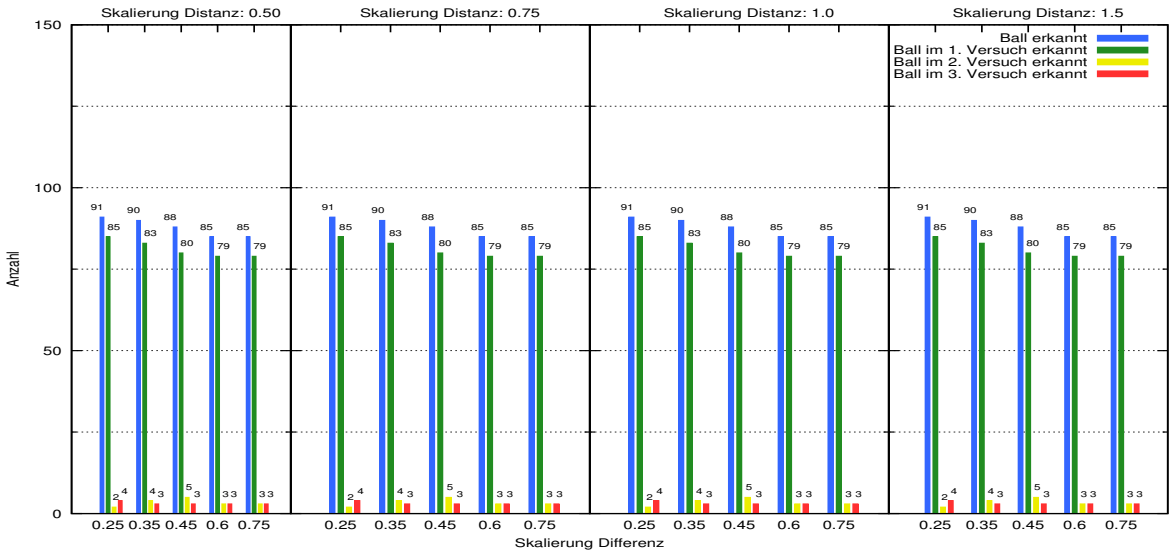
Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=14%)



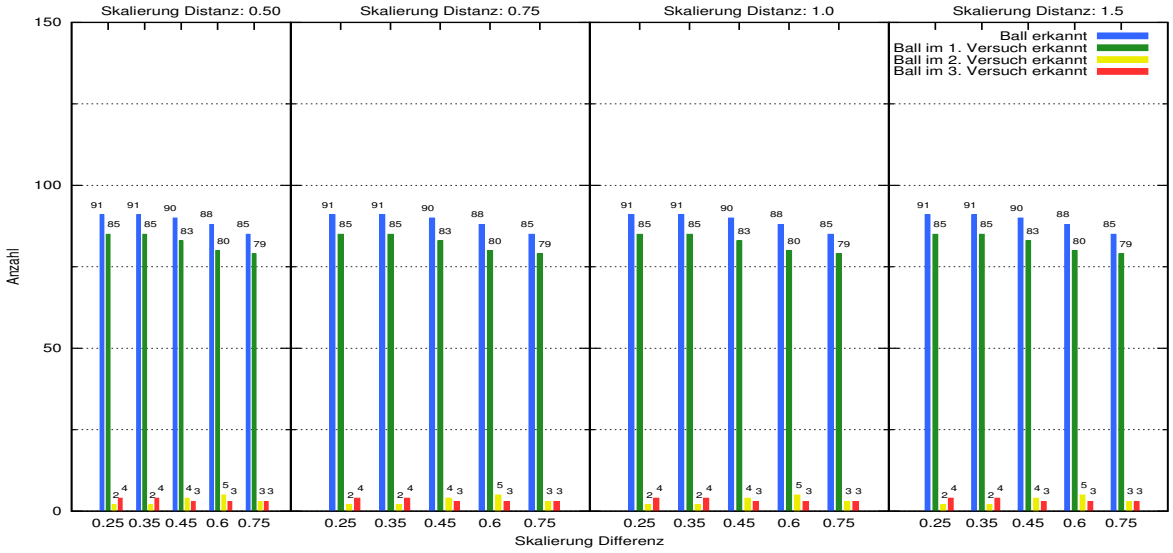
Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=21%)



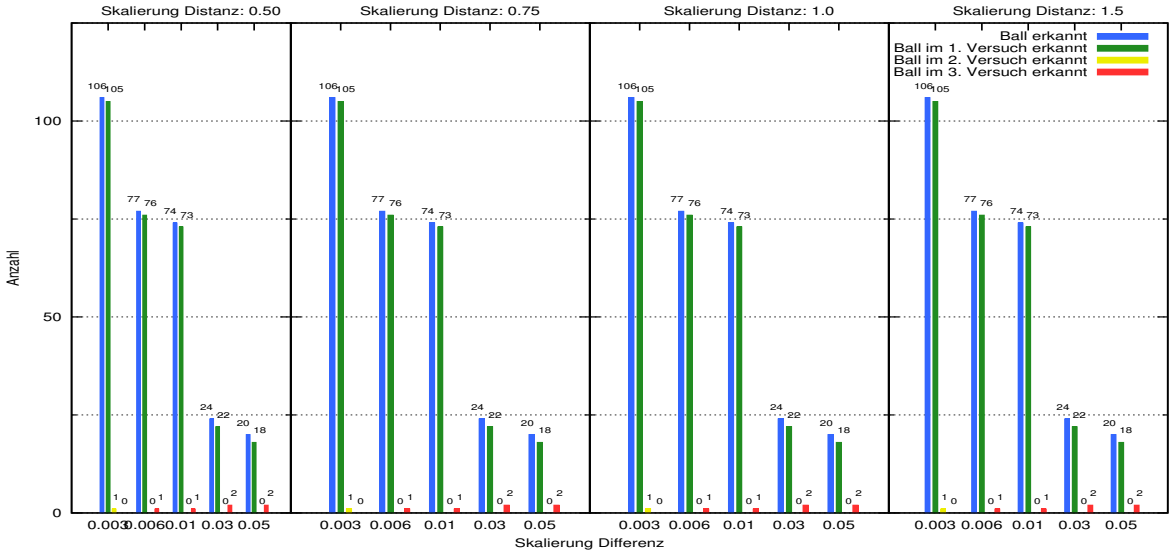
Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=28%)



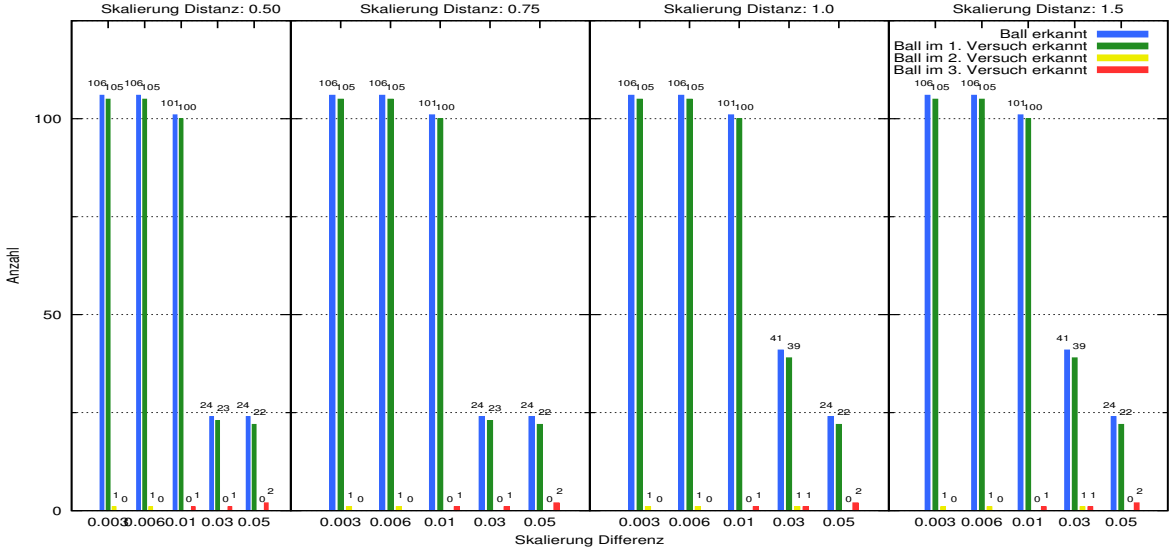
Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=linear, Faktor für gültigen Bereich=35%)



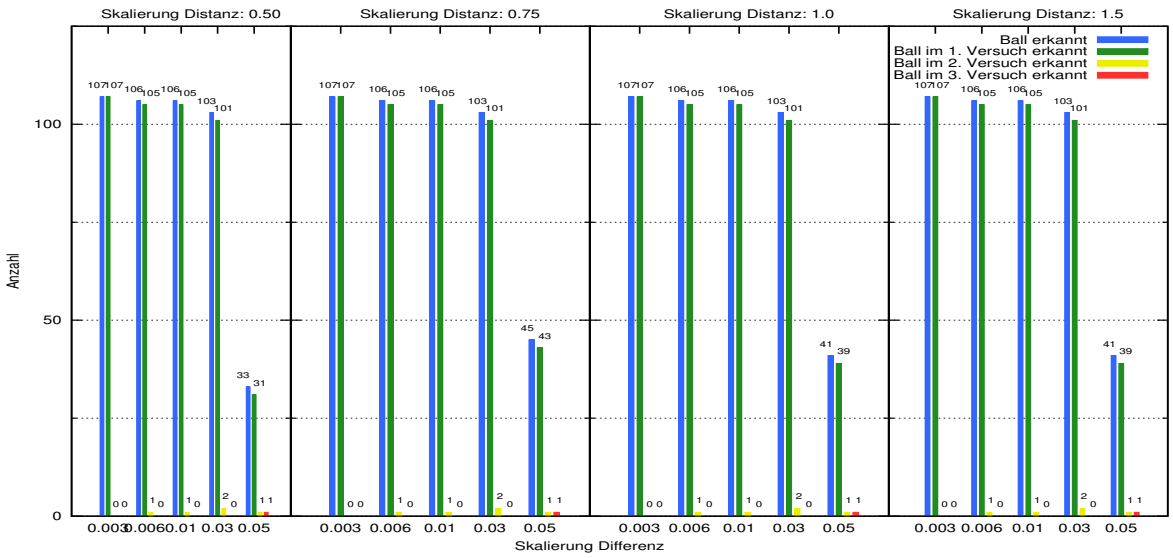
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=7%)



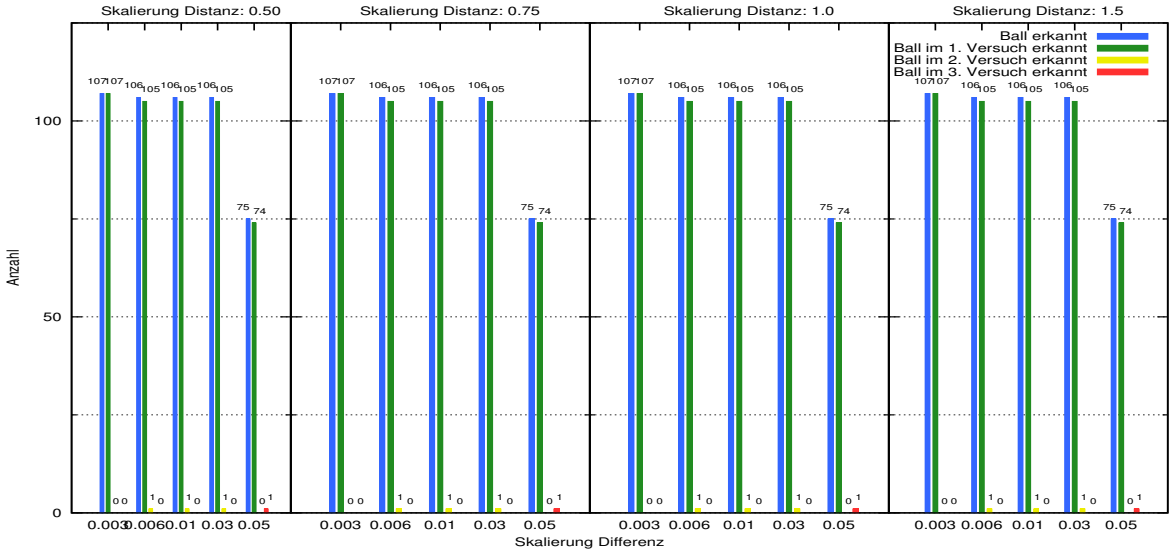
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=14%)



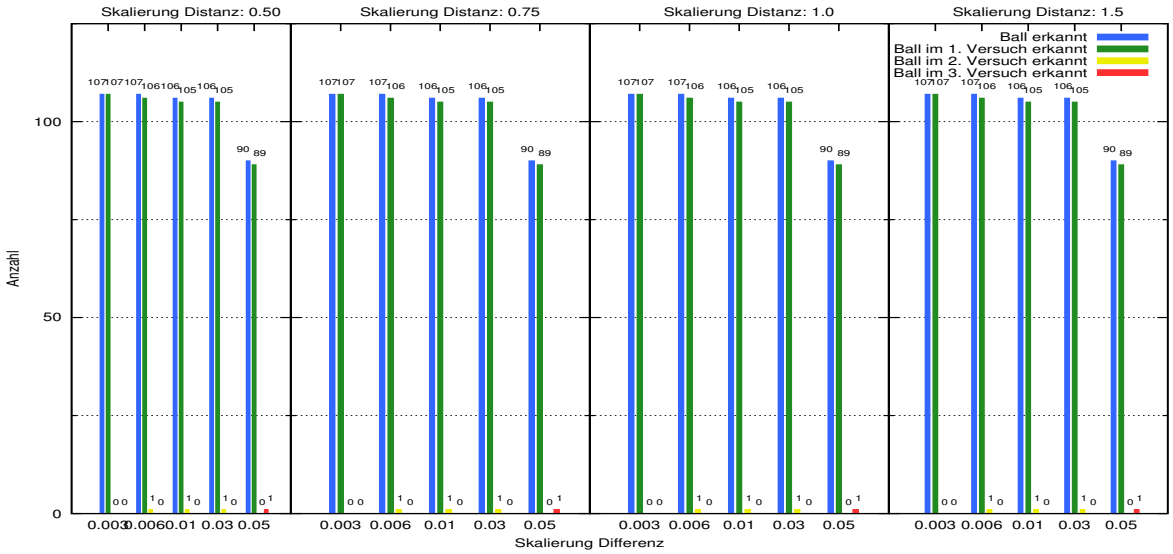
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=21%)



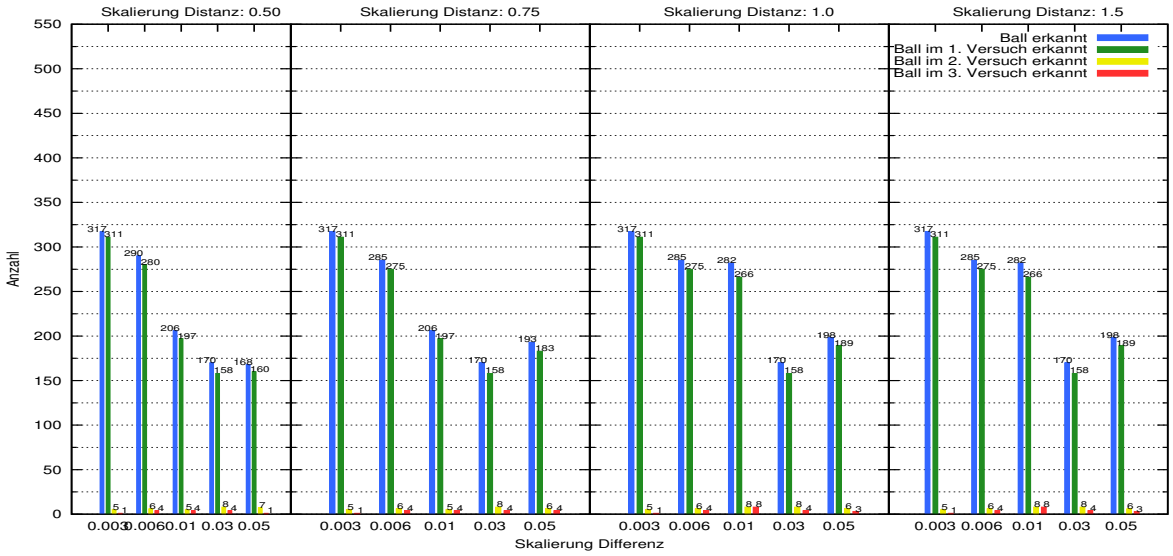
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=28%)



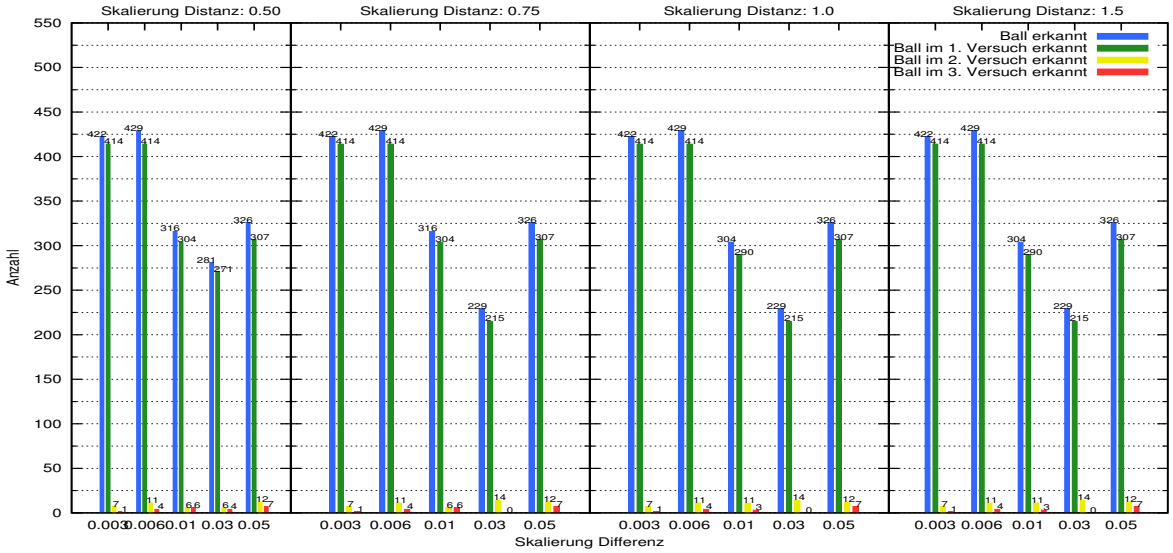
Video 1: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=35%)



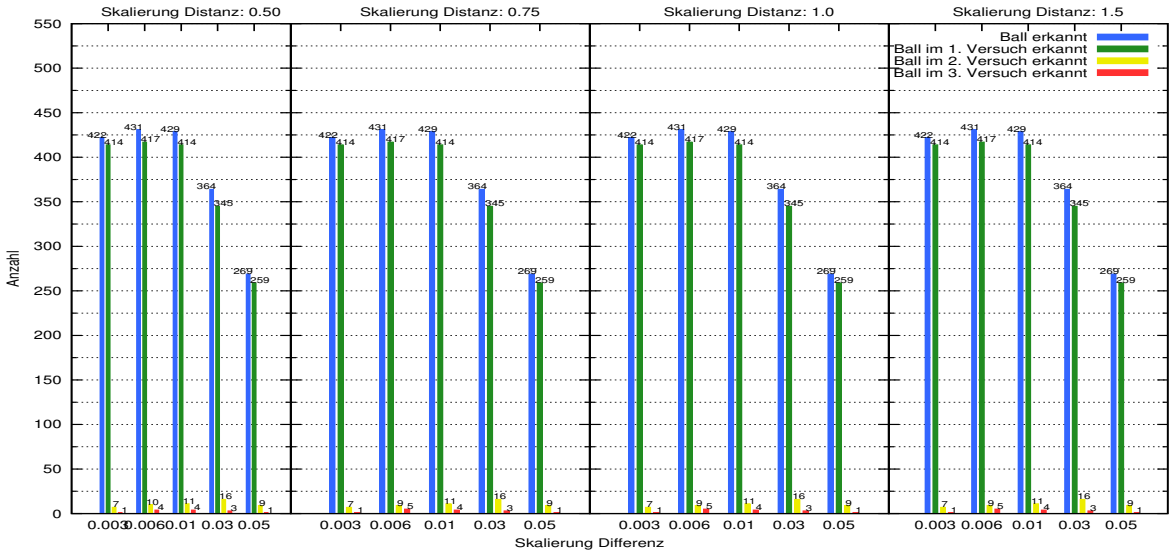
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=7%)



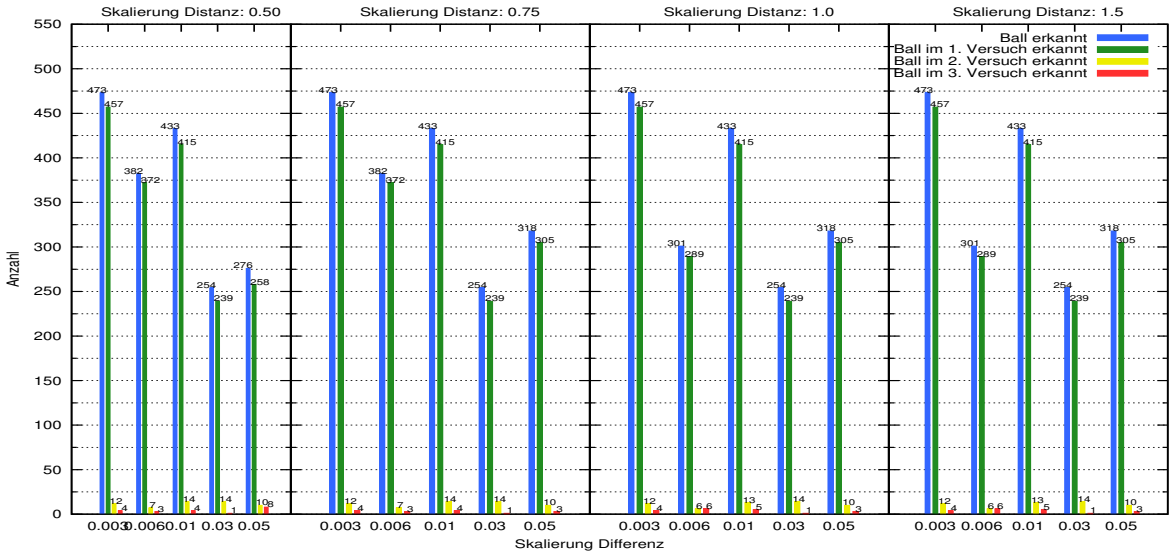
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=14%)



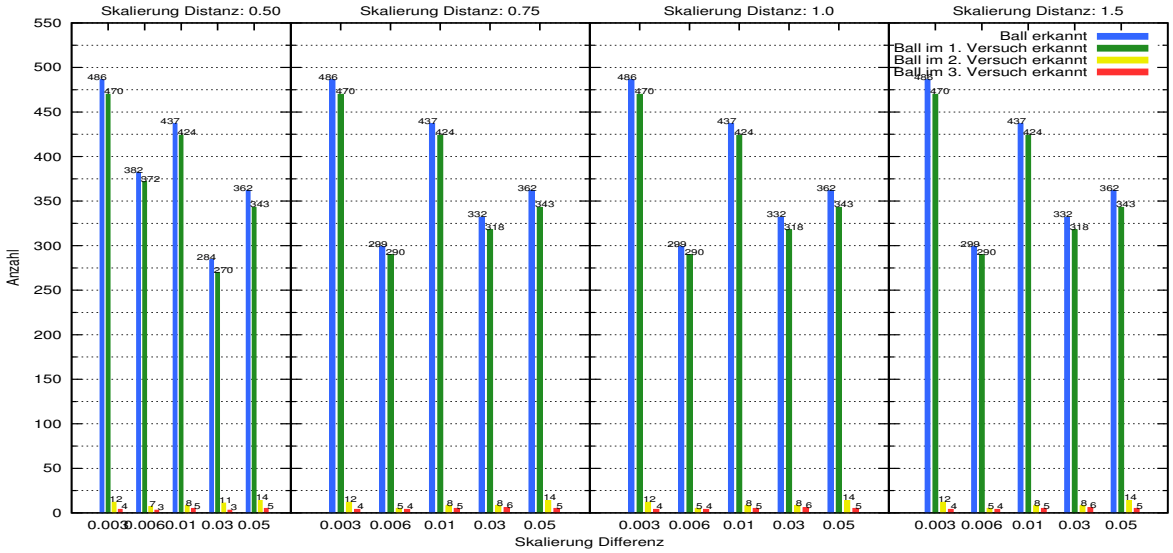
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=21%)



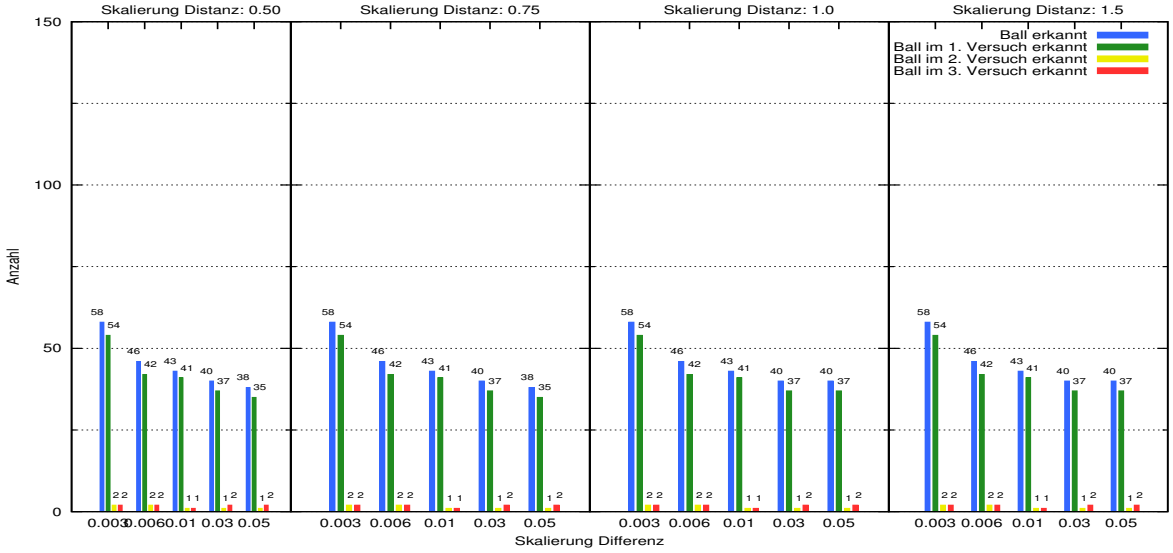
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=28%)



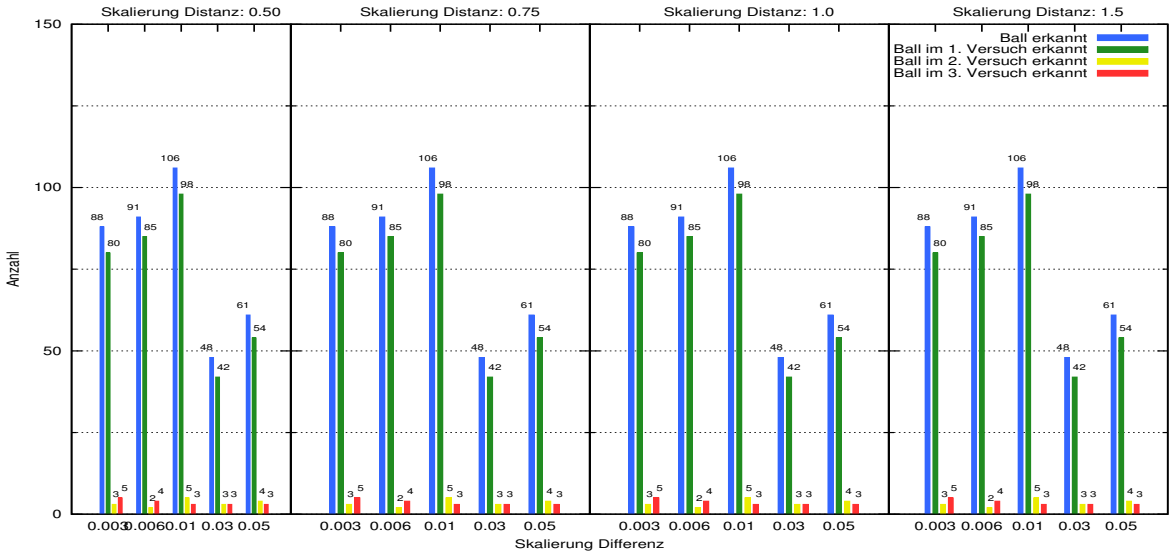
Video 2: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=35%)



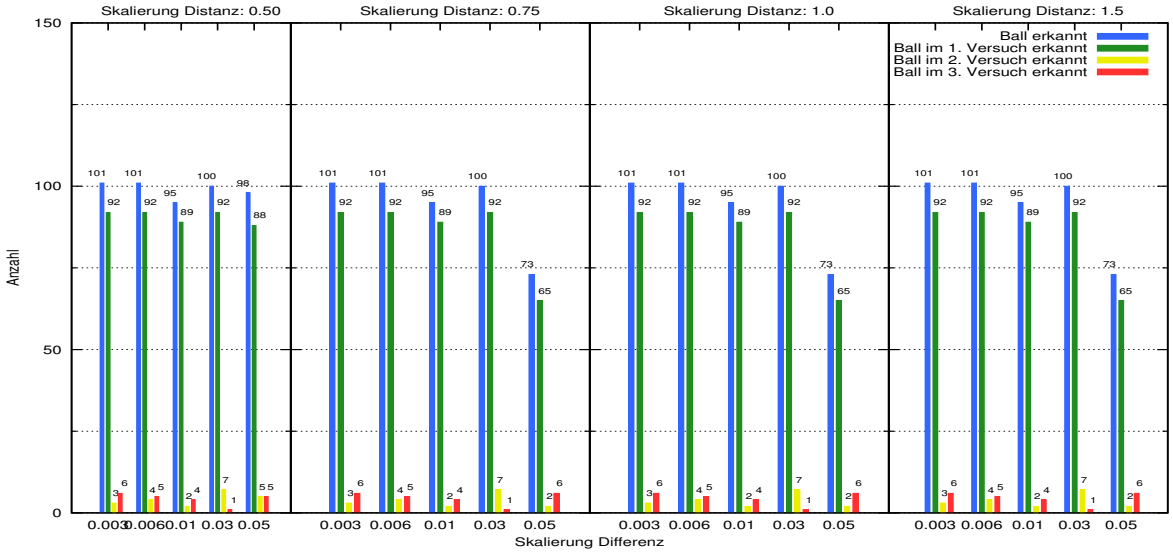
Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=7%)



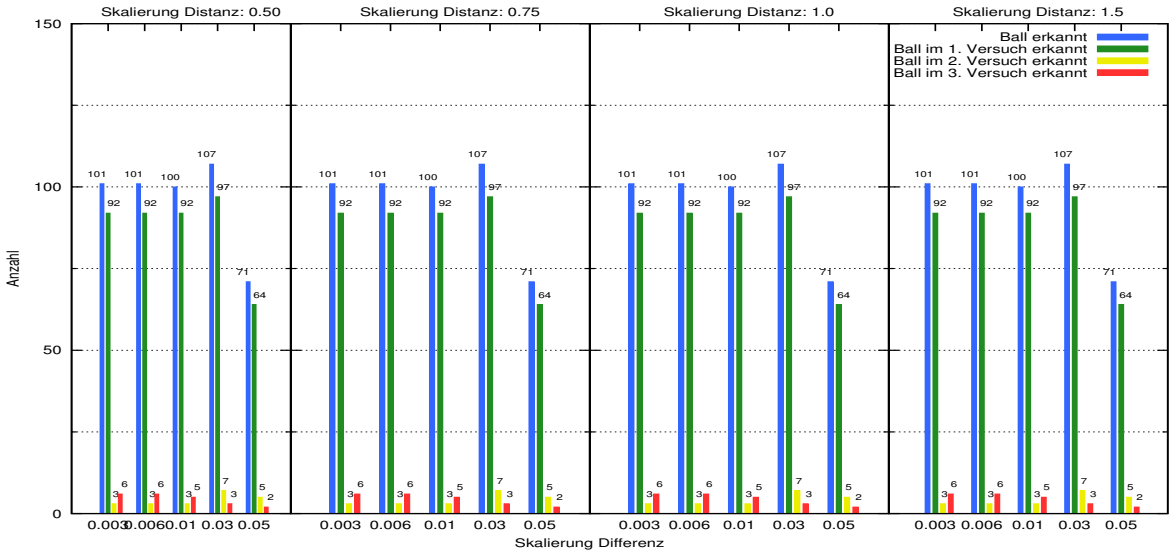
Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=14%)

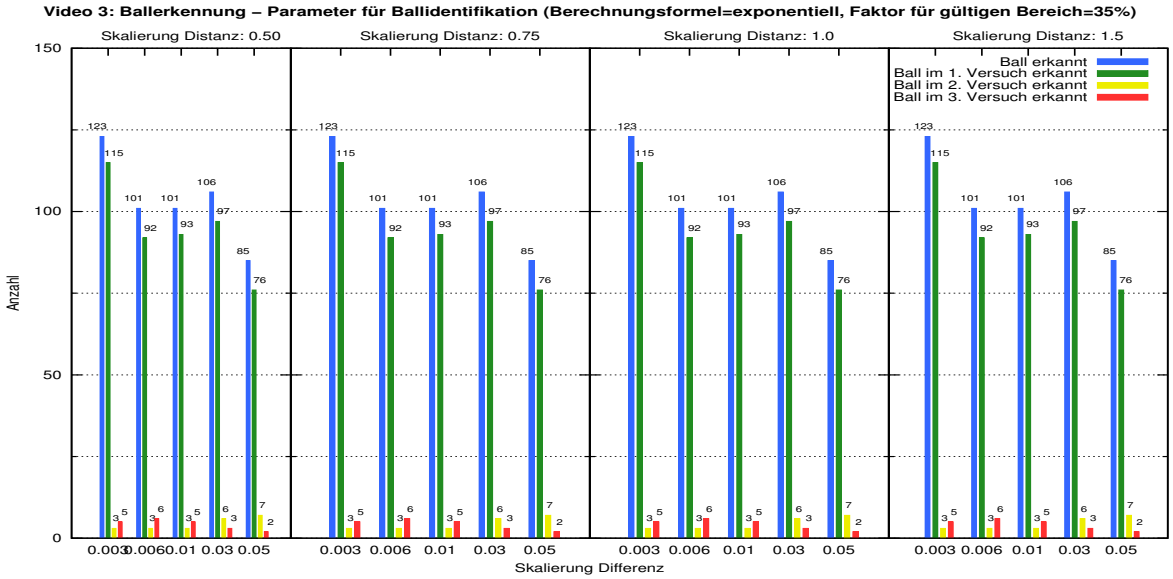


Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=21%)



Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=28%)




Video 3: Ballerkennung – Parameter für Ballidentifikation (Berechnungsformel=exponentiell, Faktor für gültigen Bereich=35%)
Skalierung Distanz: 0.50
Skalierung Distanz: 0.75
Skalierung Distanz: 1.0
Skalierung Distanz: 1.5
Ball erkannt
Ball im 1. Versuch erkannt
Ball im 2. Versuch erkannt
Ball im 3. Versuch erkannt
Anzahl
Skalierung Differenz
0.003 0.006 0.01 0.03 0.05
123 115 3 5
101 92 3 6
101 93 3 5
106 97 6 3
85 76 7 2

Literaturverzeichnis

[1] *The OpenCV Reference Manual: Release 3.0.0-dev.* http://docs.opencv.org/3.0-beta/opencv2refman.pdf. Version: 25.06.2014

[2] Angel, Edward; Shreiner, Dave: *Interactive computer graphics: A top-down approach with shader-based OpenGL.* 6th ed. Boston : Addison-Wesley, 2012. – ISBN 978–0–13–254523–5

[3] Chaumont, F. de; Dufour, A.; Olivo-Marin, J.-C.: Tracking articulated objects with physics engines. In: *2009 16th IEEE International Conference on Image Processing ICIP 2009*

[4] Gopal Sarma Pingali, Yves Jean and Ingrid Carlbom: *Real Time Tracking for Enhanced Tennis Broadcasts*

[5] Greg Welch and Gary Bishop: *An Introduction to the Kalman Filter.* Chapel Hill, NC 27599-3175, University of North Carolina, Paper, 2006

[6] Han, Jiawei; Kamber, Micheline: *Data mining: Concepts and techniques.* 2. ed., [Nachdr.]. Amsterdam : Elsevier/Morgan Kaufmann, 2010 (The Morgan Kaufmann series in data management systems). – ISBN 978–1–55860–901–3

[7] Kovacs, M. S.: Applied physiology of tennis performance. In: *British journal of sports medicine* 40 (2006), Nr. 5, S. 381–5; discussion 386. http://dx.doi.org/10.1136/bjsm.2005.023309. – DOI 10.1136/bjsm.2005.023309. – ISSN 0306–3674

[8] Kovacs, Mark S.: Applied physiology of tennis performance / University of Alabama, Department of Kinesiology, Tuscaloosa, Alabama 35487, US. 2006. – Forschungsbericht

[9] Markus Kohler: *Using the Kalman Filter to track Human Interactive Motion: Modelling and Initialization of the Kalman Filter for Translational Motion.* Dortmund, Universität Dortmund, Paper

[10] Ph. Egert: *FM-DBSCAN: Ein effizienter, dichte-basierter Clustering-Algorithmus.* Cottbus, Brandenburgische Technische Universität Cottbus-Senftenberg, Paper, 2016. http://dx.doi.org/10.14361/9783839433461-002. – DOI 10.14361/9783839433461–002

[11] Prof. Dr. Puppe: *Data Mining: Clustering.* Würzburg, Universität Würzburg, Skript, 2015

[12] Qazi, Tayeba; Mukherjee, Prerana; Srivastava, Siddharth; Lall, Brejesh; Chauhan, Nathi R.: Automated ball tracking in tennis videos. In: *2015 Third International Conference on Image Information Processing (ICIIP)*, 2015. – ISBN –2005

[13] Richardson, Iain E. G.: *The H.264 advanced video compression standard.* 2nd ed. Chichester, West Sussex, UK : Wiley, 2010. http://dx.doi.org/10.1002/9780470989418. http://dx.doi.org/10.1002/9780470989418. – ISBN 978–0–470–51692–8

[14] Salmen; Jan: Eine Systemarchitektur für effiziente videobasierte Fahrerassistenzsysteme.

[15] Sonka, Milan; Hlavac, Vaclav; Boyle, Roger: *Image processing, analysis, and machine vision.* 3. ed., internat. student ed. Stamford, Conn. : Cengage Learning, 2008. – ISBN 0–495–24438–4

[16] Suzuki, Satoshi; Abe, Keiichi: Topological structural analysis of digitized binary images by border following. In: *Computer Vision, Graphics, and Image Processing* 29 (1985), Nr. 3, S. 396. http://dx.doi.org/10.1016/0734-189X(85)90136-7. – DOI 10.1016/0734–189X(85)90136–7. – ISSN 0734189X

[17] Teachabarikiti, Kosit; Chalidabhongse, Thanarat H.; Thammano, Arit: Players tracking and ball detection for an automatic tennis video annotation. In: *2010 International Workshop on Content Based Multimedia Indexing (CBMI)*, 2010. – ISBN –2005

[18] Thormählen: *Vorlesung Grafikprogrammierung I: Kameras: perspektivische Projektion.* Marburg, Universität Marburg, Skript, 2014. http://www.mathematik.uni-marburg.de/~thormae/lectures/graphics1/graphics_6_1_ger_web.html#1

[19] Tuytelaars, Tinne; Mikolajczyk, Krystian: Local Invariant Feature Detectors: A Survey. In: *Foundations and Trends® in Computer Graphics and Vision* 3 (2007), Nr. 3, S. 177–280. http://dx.doi.org/10.1561/0600000017. – DOI 10.1561/0600000017. – ISSN 1572–2740

[20] Waggoner, Ben: *Compression for great video and audio: Master tips and common sense.* 2nd ed. Burlington MA : Focal Press, 2010 http://www.sciencedirect.com/science/book/9780240812137. – ISBN 0240812131

[21] Wiebusch, Latoschik: *Interactive Computer Graphics: Viewing and Projection.* Würzburg, Universität Würzburg, Skript, 2016

[22] Xiao, Jianxiong: *Advances in Computer Vision: Homographies and RANSAC*. Cambridge, Massachusetts Institute of Technology, Skript, 2012. http://6.869.csail.mit.edu/fa12/lectures/lecture13ransac/lecture13ransac.pdf

[23] Yu, X.; Sim, C. H.; Wang, J. R.; Cheong, L. F.: A trajectory-based ball detection and tracking algorithm in broadcast tennis video. In: *Image Processing, 2004. ICIP '04. 2004 International Conference on* Bd. 2, 2004. – ISBN –2005

[24] Zhang, Zhengyou: *The proceedings of the Seventh IEEE International Conference on Computer Vision: September 20 - 27, 1999, Kerkyra, Greece*. Los Alamitos, Calif. : IEEE Computer Society, 1999. – ISBN 0769501664

[25] Zinner T.: *Vorlesung Neue Internet-Anwendungen: 2.2 Digital Image*. Würzburg, Universität Würzburg, Skript, 2015

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Würzburg, März 2017