Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Ines Hilbert

Entwurf und Implementierung einer Bildverarbeitung zur Objekterkennung und -markierung

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Ines Hilbert

Entwurf und Implementierung einer Bildverarbeitung zur Objekterkennung und -markierung

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung

im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Dipl.-Kfm. Jörg Dahlkemper
Zweitgutachter: Prof. Dr. Björn Ingo Lange

Eingereicht am: 20.09.2013

Ines Hilbert

Thema der Arbeit
Entwurf und Implementierung einer Bildverarbeitung zur Objekterkennung und -markierung

Stichworte
Bildverarbeitung, Billardtrainer, Kugelerkennung, Queueerkennung, Kamera-Projektor System, Tischerkennung, Billard, Stoßsimulation

Kurzzusammenfassung
Dieses Dokument beschreibt die Entwicklung einer Bildverarbeitunssoftware für die allgemeine Anwendung eines Billardtrainers. Das System bestimmt den Spielzustand des Billardtisches und simuliert einen Stoß mit dem aktuellen Queuewinkel. Die Ergebnisse der Objekterkennung und der Stoßberechnung werden mittels Projektor auf dem Billardtisch angezeigt. Es werden unter anderem Algorithmen zur Tischerkennung, Kugelerkennung und Queueerkennung vorgestellt.

Ines Hilbert

Title of the paper
Development and implementation of an image processing system for object detection and marking

Keywords
image processing, billiards assistant, ball recognition, cue recognition, camera-projector system, table detection, pool, shot simulation

Abstract
This document describes the development of an general image processing software for an billiards assisting system. The system determines the state of the billiard table and simulates a shot with the current cue angle. The results of the object recognition and the impact calculation are displayed by a projector onto the pool table. It will be presented amongst others algorithms for table detection, cue and ball recognition.

Inhaltsverzeichnis

1 Einleitung

Computergestütze Assistenzsysteme finden mehr und mehr ihren Weg in den Alltag der Menschen. Sie unterstützen den Menschen bei Entscheidungen durch die Aufarbeitung und Darstellung von Informationen. Unter diesen Systemen sind auch viele Anwendungen, die Bildinformationen auswerten und analysieren, wie z.B Barcode-Scanner für Mobiltelefone, Systeme zur Personenidentifizierung in Bildern, Abstands- und Spurhalteassistenten für Fahrzeuge oder Fitnessanwendungen, welche die richtige Körperhaltung bei einer Übung überwachen.

Für die Unterstütung eines menschlichen Spielers bei der Wahl eines Stoßes beim Billard, werden computergestützten Billardtrainer entwickelt. Diese bestimmen den Zustand der Spielsituation auf einen Billardtisch, analysieren diesen und geben das Ergebnis sichtbar für den Nutzer wieder aus. Auf diese Weise kann dem Spieler z.B. ein möglicher Stoß oder der Tischzustand nach einem speziellen Stoß angezeigt werden. Für den Nutzer bietet dies Möglichkeiten seinen geplanten Spielzug zu überpüfen oder einen möglichen Stoß zu entdecken, den er vorher nicht gesehen hat.

Motivation, Ausgangslage

An der HAW Hamburg wurde im Jahr 2011 mit der Entwicklung eines Billardtrainers begonnen. Es handelt sich dabei um ein fest installiertes System, welches die grundlegenden Aufgaben eines Billardtrainers zur Verfügung stellt. Es bestimmt die Kugel- und Queuepositionen und bietet die Möglichkeit einen Stoß zwischen zwei Kugeln zu simulieren. Für demonstationszwecke soll ein kleineres mobiles System realisiert werden. Die benötigte Hardware ist vorhanden.

Zielsetzung

In dieser Arbeit soll eine allgemeine Bildverarbeitungssoftware entwickelt werden, die für verschiedene Billardtrainer-Systeme einsetzbar ist. Dabei müssen die verschiedenen Eigenschaften der Systeme berücksichtigt und die draus resultierenden Probleme behoben werden.

Es soll der Zustand des Billardtisches anhand eines aufgenommenen Bildes rekonstruieren werden. Dafür müssen die Positionen der Kugeln und des Queue im Bezug auf den Billardtisch ermittelt werden. Mit diesen Eigenschaften kann eine Stoßsimulation ausgeführt werden. Das Ergebnis der Detektion und der Simulation soll direkt auf dem Billardtisch mittels eines Projektors dargestellt werden.

Themenabgrenzung

Im Gegensatz zu anderen Arbeiten und Projekten wird hier die Entwicklung einer Billardtrainersoftware gefordert, die auf verschiedene Hardwarekombinationen und Anordnung reagieren kann. Bei anderen Projekten sind diese Parameter fest definiert. Dies ist auch bei dem installierten System an der HAW bisher der Fall. Diese Systeme verlieren ihre Funktionsfähigkeit, sobald ein Parameter, wie z.B die Anordnungsgeometrie zwischen Tisch und Kamera verändert wird.

2 Stand der Technik

Im folgenden Kapitel wird zunächst ein Überblick über existierende Billardtrainer und Billardroboter gegeben. Im Anschluss folgt eine Beschreibung des computergestützten Billardtrainers an der HAW Hamburg, welcher durch die Ergebnisse dieser Arbeit weiterentwickelt wird.

2.1 Billardroboter und virtuelle Billardassistenten

Die *Snooker Maschine* war der erste Versuch das Billardspielen zu automatisieren. Sie wurde in den späten 1980er Jahren an der Universität Bristol (England) entwickelt ((vgl. Greenspan u. a., 2007), zit. n. Chang (1994)). Seid dieser Zeit gibt es eine stetige Entwicklung in den Bereichen des automatisierten und unterstützenden Spiels anhand von verschiedenen Billardspielarten an Hochschulen der ganzen Welt. Dabei haben sich zwei verschiedene Entwicklungsrichtungen gebildet. Zum einen wird das automatisierte Spiel eines Roboters angestrebt (vgl. Greenspan u. a., 2008; Müller, 1998), zum anderen soll ein menschlicher Spieler unterstützt oder bewertet werden (vgl. Chou u. a., 2009; Larsen u. a., 2002; Uchiyama und Saito, 2007; Shih, 2010). Die grundlegenden Verarbeitungsschritte der Roboter- und der Trainingssysteme unterschieden sich dabei nur wenig. Der Hauptunterschied liegt darin, wie mit der gewonnenen Information verfahren wird. Bei einem Robotersystem agiert das System, also der eingebaute Roboter, selbst mit dem Billardtisch. Bei den Trainingssystemen werden die gewonnenen Ergebnisse an den Nutzer weitergegeben, damit dieser darauf reagieren kann.

Um die Entwicklung der strategischen Fähigkeiten bei computergestützten Systemen zu fördern, werden Wettkämpfe für künstliche Intelligenzen im Bereich des Billardspielens von der International Computer Games Association (ICGA) ausgetragen.

In den nächsten Abschnitten werden drei Systeme aus unterschiedlichen Anwendungsbereichen vorgestellt.

Deep Green

An der Queen's Universität in Kingston (Kanada) wurde ein automatisches System zum Spielen der Billardvariante 8-Ball entwickelt. Ziel der Entwicklung ist es, dass das System einen professionellen menschlichen Spieler schlagen kann. Deep Green soll im Jahr 2008 besser gespielt haben, als ein Amateurspieler und Kombinationen mit Bandenschüssen gemeistert haben (vgl. Greenspan u. a., 2008).

Das System besteht aus einem an der Decke fixierten Portalroboter, an welchem sich eine Stoßvorrichtung mit sowohl pneumatischen als auch elektrischen Antrieb für einen stark gekürzten Billardqueue befindet. An der Stoßvorrichtung ist ein Kamerasystem befestigt, welches als „Local Vision System" (LVS) bezeichnet wird. Das LVS wird für das Anspielen des Spielballs genutzt und für die Platzierung des „Pick and Place Tools", mit welchen der Spielball plaziert werden kann. Als Spieltisch wird ein Standardbillardtisch mit den Maßen 4 Fuß x 8 Fuß benutzt, über welchem sich ein Kamerasystem befindet, welchens das ganze Spielfeld abdeckt. Die Daten werden von einem PC gesammelt und verarbeitet. Um den nächsten Stoß zu planen, werden die möglichen Stöße simuliert und dann der strategisch beste ausgeführt (vgl. Greenspan u. a., 2008). Der Aufbau des Systems ist in Abbildung 2.1 zu sehen.

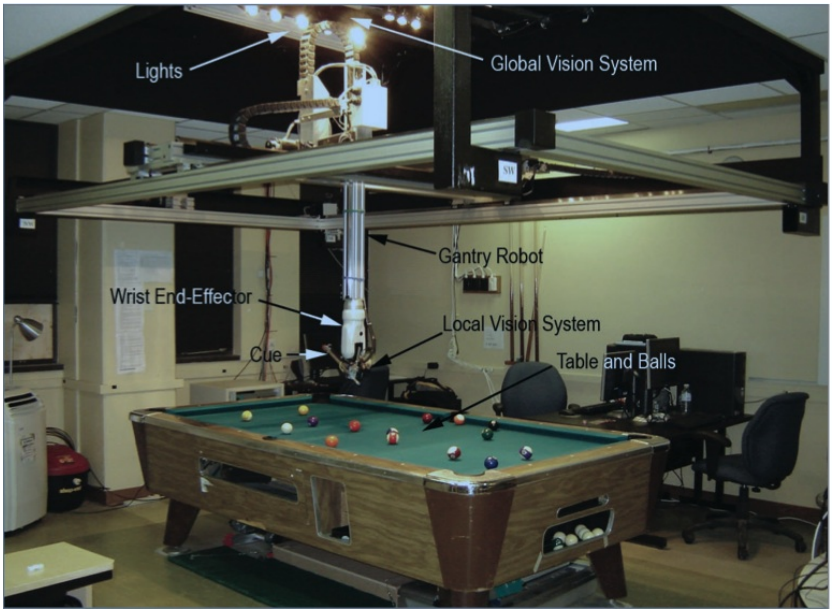


Abbildung 2.1: Hardwareanordnung *Deep Green* (vgl. Greenspan u. a., 2007)

Automatic Pool Trainer

Der *Automatic Pool Trainer* der Aalborg University in Dänemark bietet dem Nutzer verschiedene Aufgaben nach dem Target-Pool-Prinzip. Dabei werden meist die weiße und eine farbige Kugel benötigt. Das Ziel des Spielers ist es, die farbige Kugel zu versenken und die weiße in die angegebene Zielzone zu spielen. Es gibt eine Anzahl von Aufgaben, aus welchen der Spieler wählen kann. Für jede Aufgabe werden die Start- und Zielpositionen vorgegeben. Nach dem Ausführen seines Stoßes wird der Spieler über die erreichte Punktzahl informiert. Für die visuelle Ein- und Ausgabe nutzt das System eine Kamera und einen Projektor, welche über dem Tisch befestigt sind. Des Weiteren ist eine Interaktion mit dem System über Sprachsteuerung möglich. Es werden zusätzlich, zu der Entwicklung eines computergestützten Billardtrainers, auch die Möglichkeiten der verschiedenen Interaktionsformen mit dem Spieler erforscht (vgl. Larsen u. a., 2002, 2005).

Eine beispielhafte Aufgabe ist auf Abbildung 2.2 zu sehen. Die gelbe Kugel soll in die Tasche am unteren rechten Bildrand gespielt werden und die weiße soll in der Mitte der Zielscheibe, also auf der Position der gelben Kugel, liegen bleiben.

Abbildung 2.2: *Automatic Pool Trainer* der Universiät Aalborg (vgl. Larsen u. a., 2005)

Billiards Wizard

Bei dem *Billiards Wizard*, beschrieben in Chou u. a. (2009), handelt es sich um ein Programm zur Analyse von aufgezeichneten Billardspielen. Der Billiards Wizard filtert aus Videos von Billardspielen, die entscheidenden Spielsituationen heraus und empfiehlt für jede Spielsituation einen Stoß. Zusätzlich werden Statistiken zu dem betrachteten Spiel, z.B. über die Anzahl der Fehler, erstellt. Auf diese Weise kann der Nutzer aus Profispielen lernen oder sein eigenes Spiel analysieren lassen.

Die Problemstellung unterscheidet sich vor allem in der Veränderlichkeit der Hardware, der Aufnahmeposition und den Beleuchtungsbedingungen, dies stellt große Anforderungen an die Anpassungsfähigkeit des Systems. Das Hauptaugenmerk wurde auf eine robuste Tischerkennung und die Berechnung des optimalen Stoßes gelegt.

2.2 Projekt BillardTrainer an der HAW

Im Rahmen eines Wahlpflichtprojektes des Studiengangs Mechatronik wurde die erste Version eines virtuellen Billardassistenten *BillardTrainer* an der HAW Hamburg installiert. Auf die Einzelheiten der verwendeten Hard- und Software wird im Folgenden eingegangen.

2.2.1 Hardware

An der HAW Hamburg existieren zwei verschiedene Hardwaresysteme für die Realisierung eines Billardtrainers. Zum einen ein stationär aufgebautes System (stationäres Hardwaresystem), welches für die Realisierung der ersten Version des BillardTrainers genutzt wurde. Zum anderen ein kompakteres System, dass für eine Demonstration des BillardTrainers außerhalb des Projektraums angeschafft wurde (mobiles Hardwaresystem). Die Systeme können auf Abbildung 2.3 betrachtet werden. Es wird zunächst der allgemeine Hardwareaufbau erläutert, danach wird auf die verwendeten Hardwarekomponenten eingegangen.

Für die visuelle Ein- und Ausgabe sind über dem jeweiligen Billardtisch eine Kamera und ein Projektor befestigt. Die Bilddaten werden von einem zentralen Computer gesammelt und verarbeitet. Für das Anzeigen des Ergebnisses steht ein Monitor, wie auch der Projektor bereit.

Dabei werden alle spielbezogenen Informationen mit dem Projektor direkt auf der Spielfläche angezeigt. Diese Hardwarekomponenten werden im Folgenden zusammen als Hardwaresystem bezeichnet.

(a) Stationäres Hardwaresystem (b) Mobiles Hardwaresystem

Abbildung 2.3: Verschiedene Hardwareaufbauten für den BillardTrainer

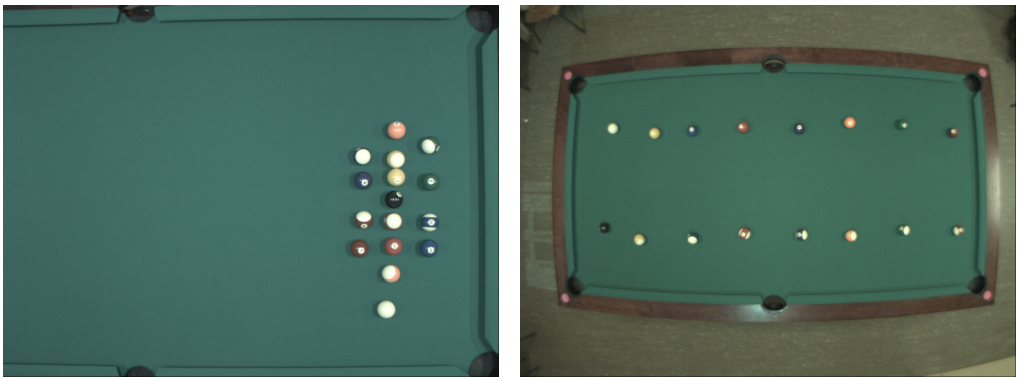
Kamerasystem

Bei der verwendeten Kamera handelt es sich bei beiden Hardwaresystemen um eine Kamera der Firma Basler vom Typ acA2500-14gc. Die Spezifikationen können aus Tabelle 2.1 entnommen werden. Da bei der Aufnahme mit maximaler Auflösung die resultierende Bildwiederholrate nur 14 Bilder pro Sekunde (fps) beträgt, wird die Auflösung auf 1280 Pixel x 960 Pixel reduziert. Bei geeigneter Einstellung der Belichtungszeit kann die Bildwiederholrate bei dieser Auflösung auf 24,5 fps maximiert werden.

Auflösung horizontal/vertikal	2590 Pixel x 1942 Pixel
Pixelgröße horizontal/vertikal	2,2 μm x 2,2 μm
Bildwiederholrate	14 fps

Tabelle 2.1: Basler acA2500-14gc Spezifikationen (vgl. Basler)

Es besteht die Möglichkeit, verschiedene Objektive zusammen mit dem Kamerakorpus zu benutzen. Es sind zwei Objektive vom Typ H614-MQ(KP) mit einer Brennweite von 6 mm und eines von Typ H416(KP) mit einer Brennweite von 4,2 mm der Firma Pentax für das Projekt verfügbar. Für die Realisierung des BillardTrainer-Prototypen wurde das Objektiv mit der Brennweite von 6 mm im stationären System eingesetzt, da es eine geringere radiale Verzeichnung aufweist. Aufgrund der Befestigungshöhe wird die Spielfläche des Billardtisches nicht vollständig abgebildet. Der Aufnahmebereich kann auf Abbildung 2.4a betrachtet werden. Das Weitwinkelobjektiv mit einer Brennweite von 4,2 mm deckt den gesamten Tischbereich ab, jedoch wird das Bild, wie auf Abbildung 2.4b zu sehen ist, stark verzerrt. Für das mobile Hardwaresystem wird eines der beiden Objektive mit 6 mm Brennweite benutzt.



(a) f=6 mm (b) f=4,2 mm

Abbildung 2.4: Aufnahme der Tisches mit verschiedenen Objektiven

Projektor

Bei dem Projektor handelt es sich bei beiden Systemen um einen Nahdistanzprojektor. Dieser ist beim stationären Hardwaresystem an der Decke über dem Billardtisch befestigt. Damit das mobile Hardwaresystem an jedem Ort aufgebaut werden kann, sind die Kamera und der Projektor dort an einem Schirmständer installiert. Das Projektionsfeld ist jeweils größer als das

Spielfeld, sodass es möglich ist, alle Objekte auf dem Tisch mit dem Projektor zu markieren. Um auf der farbigen Fläche des Billardtisches ein zufriedenstellendes Ergebnis zu erzielen, muss bei der Farbwahl auf einen hohen Kontrast im Verhältnis zu der entsprechenden Spielfeldfarbe geachtet werden.

Billardtisch und Zubehör

Im stationären Hardwaresystem wird ein 8-Fuß Poolbillardtisch mit Maßen und Kugeln nach Turnierrichtlinien benutzt (vgl. e.V, 2007). Dieser ist mit einem dunkelgrünen Spieltuch bespannt und besitzt keine Bandenmarkierungen. Für das mobile Billardsystem wird ein Kinderbillardset mit den Tischmaßen 86 cm x 43 cm mit kleineren Kugeln und Queues verwendet. Dabei sind die Kugeln nicht proportional zum Spielfeld verkleinert. Das Spielfeld ist mit einem blauen Stoff bezogen.

2.2.2 Software

Die Software für den BillardTrainer wurde speziell für das stationäre Hardwaresystem programmiert. Sie bietet einige Funktionen und Eigenschaften, die im Folgenden erläutert werden.

Funktionen

Ziel der Softwareerstellung für den BillardTrainer während des durchgeführten Projektes war es, einen Prototypen zu implementieren, um die Möglichkeiten des Systems zu demonstrieren.

Um mit dem Nutzer zu interagieren, werden zunächst die Kugeln auf dem Billardtisch erkannt. Welche Kugel als Spielball angenommen wird, hängt von den eingestellten Parametern ab. Eine beliebige andere Kugel wird für den Spieler sichtbar markiert. Wird der Queue erkannt, wird eine Linie durch den Mittelpunkt der weißen Kugel in Queuerichtung auf dem Tisch dargestellt. Falls die markierte Kugel auf der Bahn der Queuelinie liegt, wird die resultierende Bewegungsrichtung der Kugeln berechnet und angezeigt (siehe Abbildung 2.5). Dabei wird immer nur genau eine farbige Kugel betrachtet, die anderen werden bei der Laufbahnberechnung ignoriert, selbst wenn diese zwischen der weißen und der markierten Kugel liegen.

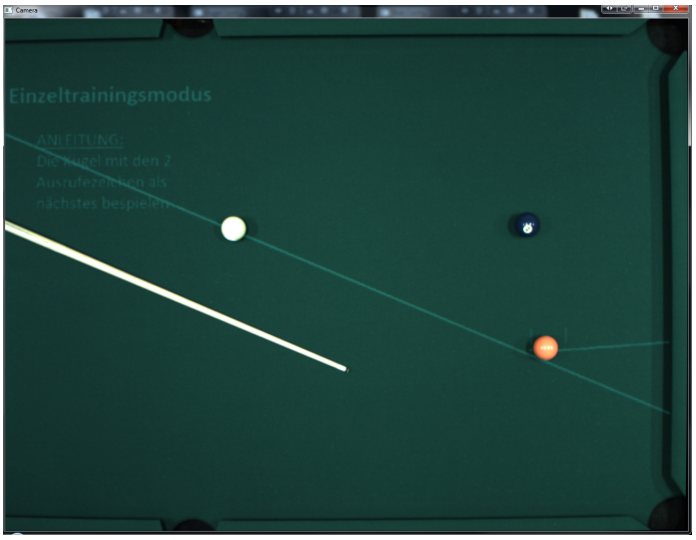


Abbildung 2.5: Ausgabe BillardTrainer-Prototyp

Sonstige Eigenschaften

Die Reaktionszeit des Systems auf die Bewegung des Queues beträgt etwa zwei Sekunden, dies ist für ein flüssiges Spielerlebnis unzureichend.

Das Programm ist speziell an das verwendete Hardwaresystem angepasst und damit nicht ohne weitere Anpassungen für eine andere Hardwarekombination einsetzbar. Zum Beispiel führt das Verwenden eines anderen Tisches oder auch das Verschieben des Tisches im Raum dazu, dass das Programm nicht mehr korrekt funktioniert. Die Unterscheidung zwischen Spielball und Objektball erfolgt über die Größe der erkannten Kugelflächen. Es nutzt somit die Eigenschaft aus, dass die weiße Kugel meist aufgrund ihres höheren Kontrastes als größer erkannt wird. Dies ist eine Eigenschaft, die nicht allgemein auf alle Billardsysteme übertragbar ist. Die Software ist prozedural gestaltet.

3 Analyse der Anforderungen

Um eine zielgerichtete Bearbeitung der Aufgabenstellung ausführen zu können, werden in diesem Kapitel zunächst die Anforderungen, die an das zu entwickelnde Programm gestellt werden, beschrieben. Sie werden in funktionale und nichtfunktionale Anforderungen unterteilt.

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen ergeben sich aus den Funktionen, welche die Billardtrainersoftware bieten soll. Zum einen sollen die an den Billardtrainer gestellten Hauptanforderungen erfüllt werden. Zum anderen ergeben sich zusätzliche Anforderungen, die sich aus der Nutzung des mobilen Hardwaresystems ergeben. In den folgenden Abschnitten wird auf diese Anforderungen näher eingegangen.

3.1.1 Anforderungen an einen computergestützen Billardtrainer

Ein Billardtrainer soll dem Nutzer helfen, sein Spiel zu verbessern. Dies soll zunächst über eine visuelle Hilfe erfolgen, die abhängig von den Kugelpositionen und dem Queuewinkel im aktuell aufgenommenen Bild die resultierenden Kugellaufbahnen auf dem Tisch anzeigt. Auf diese Weise kann der Spieler selbst einen Stoß finden, der die gewünschte Kugel versenkt. Wie ein Ergebnis aussehen könnte ist in Abbildung 3.1 dargestellt.

Aufnahme des Billardtisches und Entzerrung des Bildes

Die Kamera soll ein Bild aufnehmen, auf dem sich das ganze Spielfeld des zu beobachtenden Billardtisches befindet. Dieses Bild muss von dem zu erstellenden Programm eingelesen

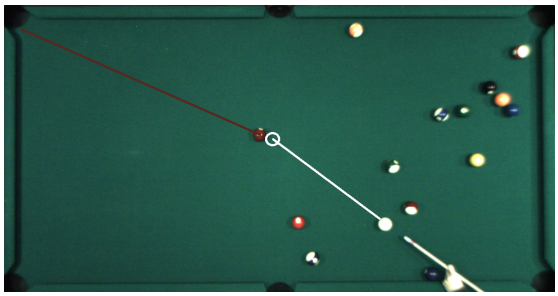


Abbildung 3.1: Gewünschte Anzeige des Billardtrainers

werden. Die radiale Linsenverzeichnung soll mit bekannten Kameraparametern korrigiert werden. Die Korrektur ist erforderlich, da sonst Objektpositionen aufgrund der Verzerrung fehlerhaft erkannt werden. Das Bild muss im ganzen entzerrt werden, da die Eigenschaften der Objekte durch die radiale Verzerrung beeinflusst werden und dies die Erkennung von selbigen erschwert.

Ermitteln des aktuellen Tischzustandes

Aus der aufgenommenen Spielsituation muss der Tischzustand korrekt bestimmt werden. Dieser ist definiert durch folgende Eigenschaften:

- Anzahl der auf dem Tisch befindlichen Kugeln
- Position und Art der Kugeln
- Richtung des Queue bei einer erkennbaren Spielabsicht

Dabei sollen die Kugeln mindestens in die Kategorien Spielball (weiße Kugel) und Objektball unterschieden werden. Eine Spielabsicht wird angenommen, wenn der Queue nahe des Spielballs positioniert wird, da nur dann ein Stoß ausgeführt werden kann.

Simulation des Stoßes

Anhand des aktuellen Tischzustandes und den allgemeinen Eigenschaften des Tisches kann der Kugelverlauf bei einem Stoß der weißen Kugel mit dem aktuellen Queuewinkel bestimmt werden. Bei der Berechnung soll zunächst nur von Stößen ohne Spin ausgegangen werden. Es soll mindestens die Laufbahn des Spielballs in Queuerichtung angezeigt werden.

Ausgabe des Ergebnisses

Es sollen alle vom System erkannten Kugeln mit dem Projektor auf dem Tisch markiert werden. Zusätzlich sollen das Ergebnis der Stoßberechnung und die berechneten Kugeltrajektorien auf dem Billardtisch angezeigt werden.

3.1.2 Anforderungen durch verschiedene Hardwaresysteme

Da das Programm sowohl mit dem stationären als auch mit dem mobilen Hardwaresystem funktionieren soll, ergeben sich weitere funktionale Anforderungen an die Software. Dabei soll die Software so allgemein wie möglich gehalten werden, um sich nicht nur auf diese beiden Systeme zu beschränken. Die resultierenden Anforderungen werden im Folgenden aufgeführt.

Anpassung an die Hardware

Da sich die Hardware des mobilen Systems von der Hardware des stationären Systems unterscheidet, muss eine Möglichkeit bestehen, die entsprechenden Parameter zu ändern. Die Parameter sollen vom System selbst ermittelt werden, soweit dies möglich ist.

Es soll auf folgende Veränderungen reagiert werden können um eine möglichst freie Wahl der Hardware zu ermöglichen:

- variable Kameraauflösung
- variable Projektorauflösung
- variable Kamera- und Projektorposition im Bezug auf die Position über dem Billardtisch
- variable Tuchfarbe des Billardtisches
- variable Taschen und Bandenpositionen im Bezug auf den Billardtisch
- variable Kugelgröße im Bezug auf den Billardtisch
- variable Farben den Kugeln

- variable Beleuchtungsbedingungen

Dabei soll die Hardware, wenn möglich, in ihrem ursprünglichen Zustand gelassen werden. So sollen z.B alle Kugeln verwendet werden können und auch keine Veränderung am Queue, wie eine Anbringung von farbigen Markern, unbedingt erforderlich sein. Dies lässt auch die Möglichkeit offen, in einem späteren Projektzustand Videomaterial von aufgenommenen Billardspielen wie in Chou u. a. (2009) zu analysieren.

Die Anpassung der Tischeigenschaften ist nötig, um eine korrekte Berechnung des Stoßes ausführen zu können.

Inbetriebnahme

Um die Inbetriebnahme des Systems zu erleichtern, soll das System die Funktion bieten, die benötigten Parameter vom Inbetriebnehmer abzufragen. Außerdem soll die Möglichkeit bestehen, das System mit bekannten Einstellungen wieder zu starten.

Zusätzlich soll auf die variable Position des Billardtisches im Bild reagiert werden können. Dazu soll das System eine Möglichkeit bieten, den Billardtisch im aufgenommenen Bild zu lokalisieren. Dabei ist nicht anzunehmen, dass sich die Position des Billardtisches während eines Spiels verändert.

Damit die Objekte auf dem Tisch korrekt markiert werden können, muss es eine Kalibrierungsfunktion für den Projektor zur Verfügung gestellt werden.

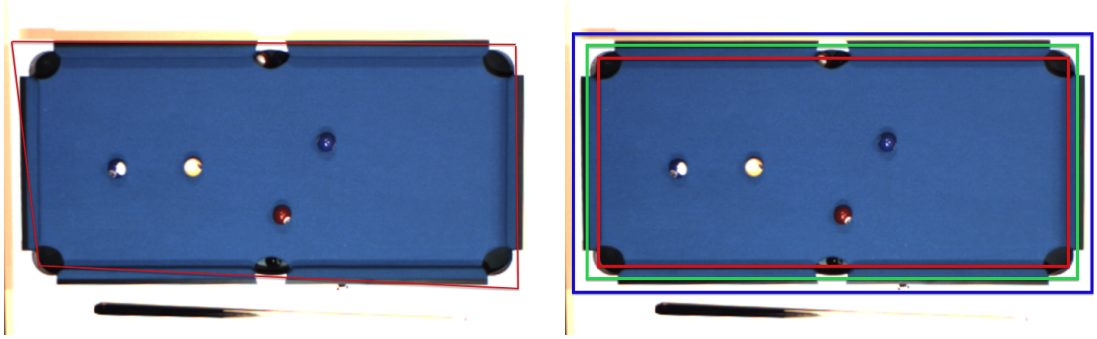
3.2 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen beschreiben Eigenschaften, die zusätzlich zu den reinen Funktionsmöglichkeiten an den Billardtrainer gestellt werden. Diese werden im Folgenden, nach (Kleuker, 2011, S. 77ff.) in unterschiedliche Kategorien eingeteilt, näher erläutert.

Korrektheit

Die verwendeten Objekterkennungsalgorithmen sollten die Objektpositionen genau bestimmen, da sonst die Berechnung des Stoßergebnisses fehlerhaft ist. Die erlaubten Abweichungen der erkannten Positionen von den realen werden im nächsten Abschnitt näher erläutert und abschließend in Tabelle 3.1 zusammengefasst. Ein Genauigkeitsgewinn darf nicht zu einer unzureichend langen Berechnungszeit führen.

Bei der Tischerkennung ist davon auszugehen, dass das Tischfeld gleichmäßig beleuchtet wird und keine starke Schattenbildung stattfindet. Es befindet sich auch nie mehr als ein Billardtisch in dem zu betrachtenden Bild. Ist der Fehler der Positionserkennung der Spielfeldecken zu groß, kann dies zu Problemen führen. Bei einer gleichmäßigen Verschiebung nach außen, kann die Fehlerkennung mit der Anpassung des Modells korrigiert werden. Bei einer nicht rechteckigen Ermittlung des Spielfeldes, welches durch die erkannten Ecken begrenzt ist, ist die Berechnung der Kugelkoordinaten im Tischmodell falsch. Ein Beispiel einer solchen Eckerkennung ist in Abbildung 3.2a zu sehen. Aus diesem Grund soll eine Formverzerrung nicht geduldet werden. Das Rechteck aus den erkannten Ecken muss mindestens ein Pixel größer als das innere Spielfeld sein, damit im Modell Banden hinzugefügt werden können. Dies ist in Abbildung 3.2b rot dargestellt. Eine Abweichung nach außen ist blau dargestellt und die optimale Spielfeldumrandung grün.



(a) Nicht erlaubte Formabweichung (b) Erlaubte Abweichungen

Abbildung 3.2: Abweichungen Tischerkennung

Bei van Balen (2009) wird eine durchschnittliche Abweichung der erkannten Kugelmittelpunkte von 0,84 cm angegeben. Im Bezug auf einen 8-Fuß Billardtisch mit Spielfeldmaßen 224 cm x 112 cm und Standardkugeln mit einem Durchmesser von 57,2 mm ergibt sich eine durchschnittliche Abweichung von 15 Prozent bei der Kugelerkennung im Bezug auf die Kugelmaße und 0,4

Prozent bezogen auf die Spielfeldlänge. Diese Genauigkeit im Bezug auf die Kugelgröße soll für das in dieser Arbeit beschriebene System angestrebt werden.

Die Sicherheit bei der Kugelerkennung bezieht sich auf ruhende Kugeln. Dies gilt für jede im Spiel vorkommende Kugel, auch für Kugeln, deren Farbkontrast zu der Spielfeldfarbe gering ist. Dabei soll die Anzahl der richtig erkannten Kugeln maximiert werden. Eine Vertrauenswahrscheinlichkeit von 70 Prozent soll mindestens erreicht werden. Der Spielball ist mit einer Wahrscheinlichkeit von 80 Prozent korrekt zu erkennen.

Für die Genauigkeit des Queuewinkels gibt es keine Vergleichsangaben, weil keine Queueerkennung bei den in Kapitel 2 genannten Projekten erfolgt. Da der optimale Queuewinkel im zukünftigen Verlauf dieses Projektes berechnet und dem Spieler vorgegeben werden soll, ist die Genauigkeit der Queueerkennung weniger kritisch zu betrachten, als die der Kugelerkennung. Die durchschnittliche Abweichung soll so gering wie möglich gehalten werden, aber 5 Grad nicht übersteigen.

Parameter	**Erlaubte Abweichung bei Objekterkennung**
Abweichung Tischeerkennung	Siehe Abbildung 3.2b
Abweichung Kugelmittelpunkt	15 Prozent im Bezug auf den Durchmesser
Sicherheit der Kugelerkennung	Vertrauenswahrscheinlichkeit von 70 Prozent
Spielballklassifikation	80 Prozent der Bilder
Abweichung Queuewinkel	max 5° im Durchschnitt

Tabelle 3.1: Erlaubte Abweichungen zwischen erkannten Objekten und Realität

Speicher- und Laufzeiteffizienz

Bei Larsen u. a. (2002) ist eine Bildaufnahmerate von 12 fps als ausreichend schnell für die Realisierung eines Billardtrainers angegeben. Die Rechenzeit des Programms soll dabei geringer sein als die Bildaufnahmezeit. Bei einer Bildwiederholrate von 12 fps führt dies zu der maximal zugelassenen Laufzeit von ca. 83 ms, bei einer Bildwiederholrate von 24,5 fps zu einer Laufzeit von unter 40,8 ms.

Änderbarkeit

Da die Software im Zukunft durch weitere Projekt- oder Abschlussarbeiten erweitert werden soll, ist bei der Strukturierung darauf zu achten, dass die Software modular aufgebaut ist. Dies vereinfacht das Ändern von Komponenten und verringert die Einarbeitungszeit.

3.3 Anforderungskatalog

Die wichtigsten in diesem Kapitel beschriebenen Anforderungen werden Tabelle 3.2 und Tabelle 3.3 zusammengefasst.

Nummer	Beschreibung der Anforderung	Priorität
1.	Ermittlung der Kugelanzahl und Positionen	hoch
2.	Erkennung des Spielballs	hoch
3.	Erkennung des Queuewinkels	hoch
4.	Simulation des Stoßergebnisses	mittel
5.	Markierung der Kugeln	hoch
6.	Ausgabe des Simulationsergebnisses	hoch
7.	Anpassungsfähigkeit an die Hardware	hoch
8.	Tischerkennung für Inbetriebnahme	mittel

Tabelle 3.2: Funktionale Anforderungen an den Billardtrainer

Nummer	Beschreibung der Anforderung	Priorität
9.	Einhalten der angegebenen Abweichungen bei der Objekterkennung	hoch
10.	Einhalten der erlaubten Rechenzeit	hoch
11.	Modulare Struktur der Software	mittel

Tabelle 3.3: Nichtfunktionale Anforderungen an die Billardtrainer-Software

Aus diesen Anforderungen ergibt sich die Aufgabenstellung, eine Bildverarbeitunssoftware zu erstellen, welche mit beiden Hardwaresystemen funktionsfähig ist. Die Hauptaufgabe der Bildverarbeitung ist es, den Tischzustand durch die Detektion der Kugeln und des Queue zu gewinnen sowie das Markieren der erkannten Objekte. Außerdem muss für die Inbetriebnahme eine Tischerkennung realisiert werden.

4 Konzeption

In diesem Kapitel werden verschiedene Konzepte für die zu lösenden Teilaufgaben vorgestellt. Zunächst werden die möglichen Entwicklungsumgebungen vorgestellt, danach folgen Konzepte für die Tischerkennung, die Kugelerkennung und die Queueerkennung.

4.1 Wahl der Entwicklungsumgebung

Es gibt eine Vielfalt von Programmbibliotheken oder Programmen die Algorithmen der Bildverarbeitung zur Verfügung stellen. Darunter fallen zum Beispiel ImageJ, Labview, Matrox Imaging Library und SimpleCV.

Für diese Arbeit soll eine Entscheidung zwischen MATLAB R2012b mit allen benötigten Erweiterungen und der Bildverarbeitungbibliothek OpenCV 2.4 getroffen werden. Beide Varianten werden erfolgreich an der HAW Hamburg für Bildverarbeitungsaufgaben eingesetzt.

Es werden zunächst die beiden Möglichkeiten vorgestellt und und im Anschluss in verschiedenen Aspekten verglichen. Das Ergebnis des Vergleiches wird in Tabelle 4.1 zusammengefasst.

Matlab

MATLAB ist eine kommerzielle Software der Firma The MathWorks. Sie besteht aus einer interaktiven Umgebung für numerische Berechnungen, Visualisierungen und Programmierung und verwendet eine eigene Programmiersprache. MATLAB ist primär für die numerische Berechnung mit Hilfe von Matrizen ausgelegt. Es ist für mehrere Betriebssysteme erhältlich und die geschriebenen Skripte sind zwischen den Betriebssystemen übertragbar, da sie direkt aus der Matlabumgebung gestartet werden (vgl. MathWorks).

OpenCV

OpenCV (Open Source Computer Vision) ist eine freie Bibliothek mit mehr als 2500, für die Bildverarbeitung optimierten, Algorithmen. Es werden Schnittstellen für die Nutzung mit verschiedenen Programmiersprachen bereitgestellt, darunter Python, C und C++. Außerdem werden verschiedene Betriebssysteme unterstützt. OpenCV wurde mit einem Fokus auf Echtzeitanwendungen und effiziente Berechnungen entwickelt (vgl. Willowgarage).

Installation und Konfiguration

Da es sich bei Matlab um ein eigenständiges Programm handelt, lässt es sich als ein Softwarepaket installieren. Die für die Bildverarbeitung benötigten Algorithmen und Funktionen sind in dem Programm enthalten und können direkt genutzt werden. Für das Ansteuern der Kamera können Funktionen der *Image Acquisition Toolbox* verwendet werden, Bildverarbeitungsalgorithmen stellt die *Image Processing Toolbox* zur Verfügung.

OpenCV setzt eine funktionierende Programmierumgebung für C/C++ voraus. Für verschiedene Betriebssysteme müssen die entsprechenden Bibliotheksdateien geladen und unter Umständen neu kompiliert werden. Für ein C/C++ Projekt, welches Funktionen von OpenCV verwendet, müssen die Bibliotheksdateien und Header eingebunden werden.

Hilfen und Beispiele

Für beide Varianten gibt es sowohl Bücher für den Einstieg (vgl. Bradski und Kaehler, 2008; Laganière, 2011; Gonzalez u. a., 2009), als auch Tutorials und Beispiele. Außerdem ist für beide Softwarepakete eine ausführliche Dokumentation erhältlich. Im Internet gibt es eine Vielzahl an fertigen Programmiervorlagen. Bei OpenCV kann es jedoch zu Problemen kommen, falls der Versionsunterschied zu groß ist.

Unterstützung der Hardware

Sowohl Matlab als auch OpenCV unterstützen eine große Anzahl von Kameramodellen. Jedoch ist die Nutzung der im BillardTrainer-Projekt verwendeten Kamera mit der Version 2010b von MATLAB nicht möglich. Um diese nutzen zu können, ist eine neuere Version erforderlich. Für C++ Projekte bietet sich die Möglichkeit an, weitere Bibliotheken in das Projekt einzubinden.

Es kann zum Beispiel die Bibliothek des Kameraherstellers Basler eingebunden werden, um weitere Funktionen der Kamera nutzen zu können. Auf diese Weise lässt ein C++ Projekt eine größere Variation bei der verwendeten Hardware zu.

Rechenzeit

Die unterschiedlichen Berechnungszeiten von MATLAB und OpenCV für Bildverarbeitungsalgorithmen wird in Matuska u. a. (2012) untersucht. Dabei wird das Rechenverhalten bei verschiedene Grundaufgaben der Bildverarbeitung wie dem Glätten von Bildern oder das Anwenden von Kantenoperatoren untersucht. Es wird festgestellt, dass der selbe Algorithmus in OpenCV viermal bis dreißigmal weniger Rechenzeit in Anspruch nimmt als Matlab. Im Falle des Erosionsalgorithmus soll die Berechnungsdauer mit OpenCV sogar 100 mal schneller sein (vgl. Matuska u. a., 2012). Bei eigenen Messungen wurde dieses Verhalten bestätigt.

Zusammenfassung

Sowohl Matlab als auch OpenCV sind für das Projekt grundsätzlich geeignet. Sie bieten beide eine Vielzahl von Bildverarbeitungsfunktionen. Auch wenn Matlab durch seine leichtere Installation einen schnelleren Einstieg für weitere Projektteilnehmer bieten würde, wird diese Arbeit in C/C++ mit OpenCV realisiert. Der Hauptgrund ist die zu lange Berechnungsdauer von Matlab, die eine Realisierung des Projektes in Videoechtzeit verhindert.

Beschreibung	**MATLAB**	**OpenCV**
Installation und Konfiguration	+	-
Hilfen und Beispiele	+	+
Unterstützung der Hardware	-	+
Rechenzeit	-	+

Tabelle 4.1: Vergleich der Entwicklungsumgebungen

4.2 Tischerkennung

Für das Bewerten der Spielsituation in einem aufgenommenen Bild muss zunächst die Position des Billardtisches in der Aufnahme ermittelt werden. Ausgehend von dieser können die

Positionen der weiteren Objekte im Bezug auf den Billardtisch ermittelt werden. Im Folgenden werden zwei verschiedene Möglichkeiten für die Positionserkennung eines Billardtisches auf einem Bild vorgestellt und im Anschluss verglichen.

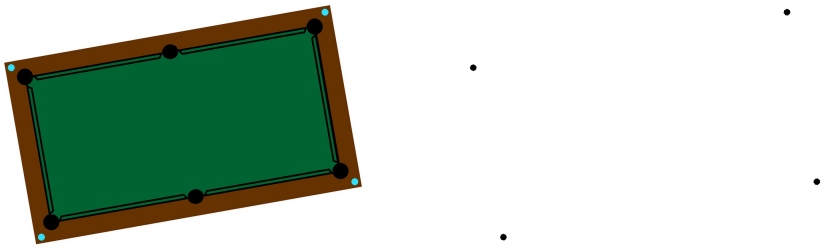
Markerbasierte Erkennung

Um die Objekterkennung zu erleichtern, kann das zu erkennende Objekt mit einem bekannten Muster markiert werden. Diese Muster werden auch als Marker bezeichnet. Dadurch muss nicht mehr das Objekt an sich in dem aufgenommenen Bild lokalisiert werden, sondern der angebrachte Marker. Der Marker sollte so gewählt werden, dass er leicht mit Methoden der Bildverarbeitung erkennbar ist, wie z.B. ein farbiger Kreis auf weißem Hintergrund oder ein binäres zweidimensionales Muster (siehe Abbildung 4.1). Dieses Verfahren eignet sich auch für das Verfolgen von Objekten in einer bewegten Szene (vgl. Kato und Billinghurst, 1999).

(a) Binäres 2d Muster (b) Farbiges Kreismuster

Abbildung 4.1: Beispiele für Marker zur Objekterkennung

Um Position, Größe und Orientierung des Billardtisches auf einem Bild festzustellen, bietet es sich an an jeder Spielfeldecke des Billardtisches einen Marker zu befestigen. Hierfür können z.B farbige Kreise genutzt werden, da diese leicht mit Methoden der Kreiserkennung zu detektieren sind. Die Marker können durch eine Segmentierung des Bildes nach Farbe aus dem Bild herausgefiltert werden. Das Ergebnis einer Segmentierung ist ein Binärbild indem die nicht relevanten Pixel den Wert null haben und die Pixel, welche die Anforderung erfüllen den Wert eins zugewiesen bekommen. Bei einer Farbsegementierung wird allen Pixeln, die sich in den bestimmten Farbbereich befinden eins zugewiesen. Eine Markerdetektion über Farbsegementierung ist in idealisierter Form auf Abbildung 4.2 zu sehen. Danach können die Kreise mit Methoden der Kreiserkennung in dem gefilterten Bild erkannt werden. Die gefundenen Mittelpunkte der Marker werden nun den korrespondierenden Punkten des Billardtisches zugeordnet. Aufgrund seiner Symmetrie ist die Position und Größe des Tisches durch die Lage der vier Eckpunkte zueinander definiert.



(a) Ausgangsbild mit Markern (b) Nach Markerfarbe segmentiertes Bild

Abbildung 4.2: Idealisierte Segmentierung von farbigen Markern

Merkmalsbasierte Erkennung

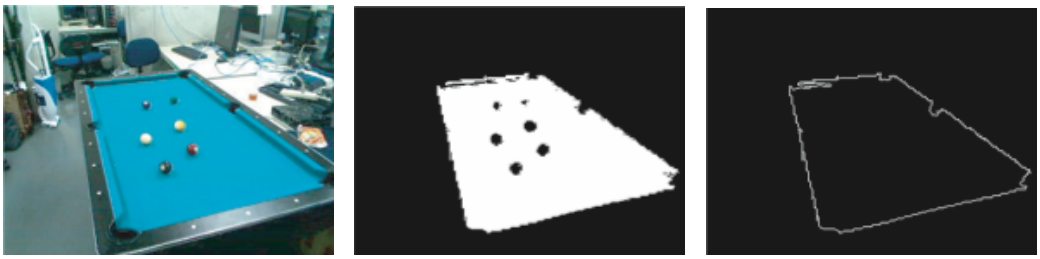
Bei einer Erkennung eines Objektes ohne Marker muss die Erkennung über die spezifischen Eigenschaften des Objektes erfolgen. Die Haupteigenschaft eines Billardtisches ist dabei die einfarbige Spielfläche. Diese hat bei einer Aufnahme von oben ein Seitenverhältnis nahe 2:1.

UCHIYAMA und SAITO haben ein Verfahren zur markerlosen Tischerkennung beschrieben, welches im Folgenden erläutert wird. Die Aufnahme des Tisches muss für dieses Verfahren nicht von oben aufgenommen sein. Jedoch muss sich der Tisch im Zentrum der Aufnahme befinden, um die richtige Zuordnung der Tischkanten zu gewährleisten.

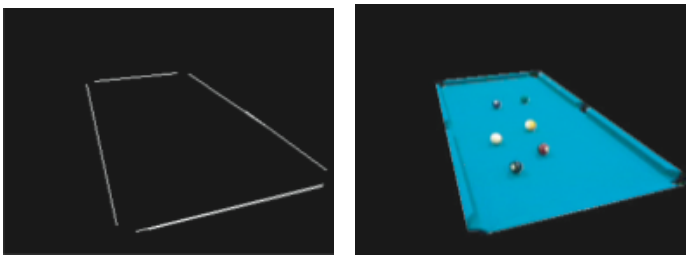
Zunächst wird das aufgenommene Bild anhand der Tuchfarbe segmentiert. Auf diese Weise werden alle Bereiche des Bildes von der weiteren Betrachtung ausgeschlossen, die nicht der angenommenen Tischfarbe entsprechen. In dem resultierenden Binärbild wird eine Konturerkennung ausgeführt, um eine Linienerkennung auf den erkannten Objekten folgen zu lassen. Anhand der Schnittpunkte der erkannten Liniensegmente können die Eckpunkte des Spielfeldes bestimmt werden. Der Vorgang ist in Abbildung 4.3 gezeigt. Bei einer Erkennung von mehr als vier Linien, wird eine weitere Filterung der erkannten Geraden ausgeführt, um die Tischkanten zu bestimmen (vgl. Uchiyama und Saito, 2007).

Zusammenfassung

Beide Verfahren eignen sich für das Lokalisieren eines Billardtisches in einem Bild. Bei einer zu starken Einschränkung der erlaubten Farben, sowohl bei dem markerbasierten als auch bei dem merkmalsbasierten Verfahren, sind beide Methoden anfällig gegenüber Beleuchtungsänderungen oder Farbverschiebungen im aufgenommenen Bild. Wird der erlaubte Farbbereich zu groß



(a) Eingangsbild (b) Farbsegmentierung (c) Tischkontur



(d) Liniensegmente (e) Tischgebiet

Abbildung 4.3: Merkmalsbasierte Tischerkennung (vgl. Uchiyama und Saito, 2007)

gewählt, kann es zu einer Fehlerkennung von anderen Objekten kommen. Das markerbasierte Verfahren ist zusätzlich anfällig für Formveränderungen der Marker. Jedoch bietet es den Vorteil, dass es für unterschiedlichste Objekte verwendet werden kann. Da die Marker nicht genau in den Spielfeldecken angebracht werden können, wird auch ein Teil der Tischumrandung mit betrachtet. Dies stellt kein Problem dar, muss jedoch bei der Ermittlung der Bandenpositionen beachtet werden.

Der Hauptvorteil der merkmalsbasierten Vorgehensweise liegt darin, dass keine Modifikationen an dem Billardtisch vorgenommen werden müssen. Auf diese Weise kann die Methode für jedes aufgenommene Bild eines Billardtisches benutzt werden. Eine feste Einstellung der Tischfarbe ist für unser Projekt jedoch nicht möglich. Außerdem ist die Methode nicht anwendbar, wenn sich auf dem Bild ähnlich gefärbte Gebiete wie der Billardtisch befinden oder die Spielfläche durch Schattenwurf oder ungleichmäßige Ausleuchtung keine konstante Farbe aufweist.

Aufgrund der allgemeinen Anwendbarkeit wird das merkmalsbasierte Verfahren angewendet, es muss jedoch noch erweitert werden, um eine Anpassung an die verschiedenen Spielfeldfarben zu ermöglichen.

Beschreibung	markerbasiert	merkmalsbasiert
schnelle Implemetierung	+	-
Modifikation an Tisch	-	+
Robustheit	-	-

Tabelle 4.2: Vergleich Tischerkennung

4.3 Kugelerkennung

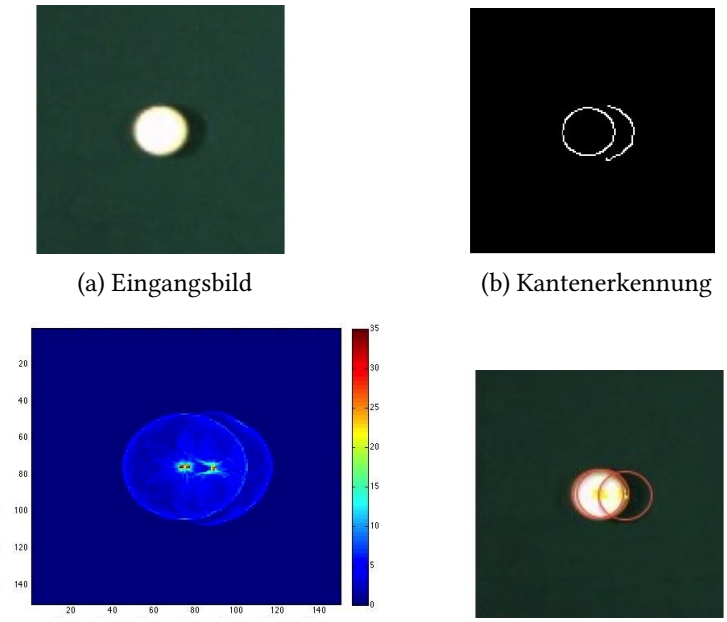
Die Erkennung der Kugelpositionen auf dem Tisch ist eine der grundlegenden Aufgaben eines Billardtrainers. Dafür müssen zunächst die Kugelpositionen im Bild erkannt werden. In einem weiteren Schritt können diese dann in Positionen auf dem Tisch umgerechnet werden. An dieser Stelle soll ein geeignetes Verfahren ausgwählt werden, um die Positionen der Kugeln im Bild zu bestimmen. Dafür werden zunächst zwei Varianten vorgestellt und im Anschluss miteinander verglichen.

Hough-Transformation für Kreise

Da Kugeln im zweidimensionalen Kamerabild als Kreise abgebildet werden, liegt es nahe die Kugelerkennung über eine Kreiserkennung mit Hough-Transformation zu realisieren. Das Verfahren der Hough-Transformation für Kreise ist z.B in (Tönnies, 2005, S. 258f.) beschrieben. Es wird für ähnliche Probleme, wie z.B. das Erkennen eines Fußballs (vgl. D'Orazio u. a., 2002) oder für die Iriserkennung (vgl. Tian u. a., 2004) bereits genutzt. In Abbildung 4.4 ist der Verlauf der Kugeldetektion mit einem festen Radius mit der Hough-Transformation dargestellt. Dabei ist auch ersichtlich, dass es Probleme mit den Kugelschatten gibt. Durch den Schatten entsteht ein weiteres Maximum im Hough-Raum, sodass der Schatten als weitere Kugel erkannt wird.

Kugelerkennung über Flächengröße von Konturen

In den in Kapitel 2 vorgestellten Projekten wird für die Kugelerkennung ein sich ähnelndes Verfahren benutzt. Dabei wird zunächst das Bild in Hintergrund und Vordergrund aufgeteilt. Bei Chou u. a. (2009) geschieht dies durch eine Farbsegementierung nach der Spielfeldfarbe. Bei Greenspan u. a. (2007); Larsen u. a. (2005) wird zunächst ein Hintergrundbild aus mehreren Aufnahmen des leeren Billardtisches erstellt. Durch die Mittelwertbildung der verschiedenen



(a) Eingangsbild (b) Kantenerkennung

(c) Hough-Raum (d) Erkannte Kreise

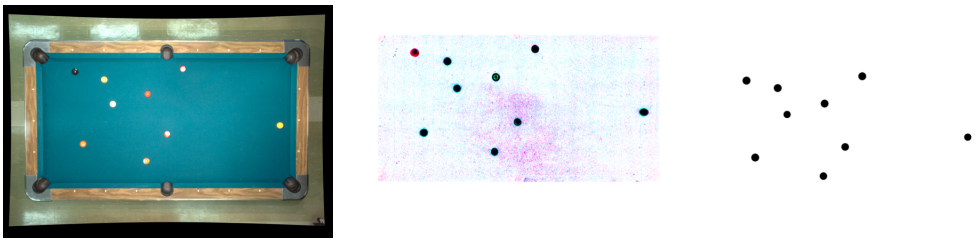
Abbildung 4.4: Kugelerkennung mit Hough-Transformation für einen Radius

Aufnahmen kann der Einfluss von Rauschen auf das Hintergrundbild verringert werden. Das Bild wird mit einer Hintergrundsegmentiertung in Vordergrund und Hintergrund aufgeteilt.

Für die weitere Kugelerkennung werden nur die Vordergrundpixel betrachtet. Das Vordergrundbild wird auf zusammenhängende Strukturen mit einer Konturerkennung untersucht, welche die passende Größe haben, um Kugeln zu sein. Die anderen Vordergrundpixel werden entfernt. Die übrigen Flächen können nun wie bei Greenspan u. a. (2007) beschrieben mit weiteren Objekterkennungsalgorithmen untersucht werden. Alternativ kann der Kugelmittelpunkt auch als Mittelpunkt des die Kontur umgebenen Rechtecks angenommen werden (vgl. Cash, 2003). Der in Greenspan u. a. (2007) beschriebene Prozess der Kugelerkennung kann in Abbildung 4.5 betrachtet werden.

Zusammenfassung

Das vorgestellte mehrstufige Verfahren über Hintergrundsubtraktion und Konturerkennung wird in verschiedenen Billardtrainer-Projekten eingesetzt. Durch die verschiedenen Schritte ist die Implementierung aufwendiger als bei einer Hough-Transformation. Durch die Hintergrundsubtraktion werden auch Einflüsse, des Tischhintergrundes eliminiert, sodass der



(a) Entzerrtes Eingangsbild (b) Hintergrundsubstraktion (c) Extrahierte Kreise

Abbildung 4.5: Kugelerkennung mit Hintergrundsubstraktion (Greenspan u. a., 2007)

Hintergrund keine störenden Einflüsse auf die spätere Kugelerkennung haben kann. Nach der Erstellung des Hintergrundbildes dürfen jedoch keine Änderungen im Hintergrund ausgeführt werden. Wird zum Beispiel die Beleuchtung stark verändert, ist das Hintergrundmodell nicht mehr korrekt.

Je größer das Bild ist, desto aufwändiger ist die Hintergrundsubtraktion, da diese für jedes Pixel des Bildes ausgeführt wird. Es wurde bei Cash (2003) beschrieben, dass dicht zusammenliegenden Kugeln, nicht einzeln erkannt werden. Ob dieses Problem mit dem in Mueller98 vorgestellten Verfahren mit Anwendung eines Erosionsfilters behoben werden kann, muss geprüft werden.

Die Hough-Transformation ist aufgrund ihres allgemeinen Ansatzes als sehr rechenintensiv bekannt. Durch die Anwendung eines Kantenoperators auf jedes Eingangsbild ist das Verfahren im Gegensatz zu dem oben genannten nicht so anfällig für Beleuchtungsänderungen, solange der Kontrast der Kugeln auf dem Spielfeld groß genug bleibt. Da die Kreiserkennung auf dem gesamten Bild stattfindet, können auch Kreise im Hintergrund fälschlicherweise erkannt werden. Außerdem kann es zu Falscherkennungen aufgrund von Schatten kommen. Es ist zu prüfen, ob das Verfahren für die Kugelerkennung geeignet ist und welche Rechenzeit dieses Verfahren in Anspruch nimmt.

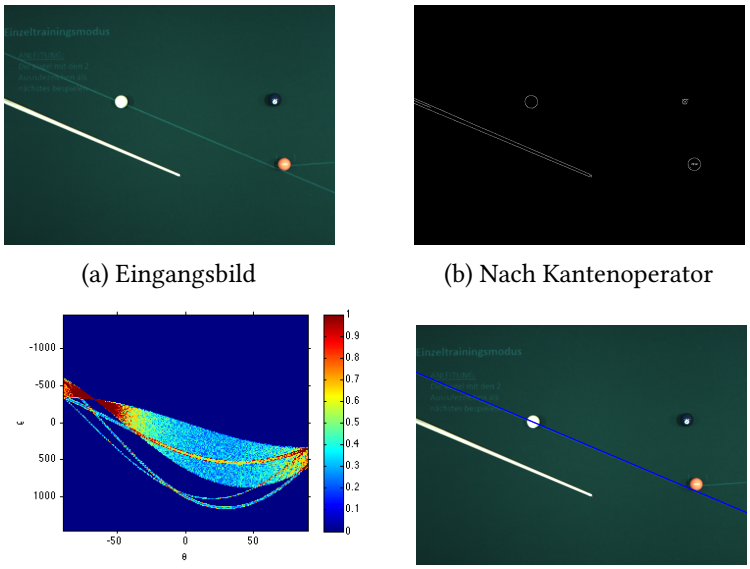
Da beide Verfahren für eine Erkennung der Kugeln geeignet sind, sollen beide Verfahren implementiert werden. Das Ergebnis der Kugeldetektoren soll verglichen werden und der Ansatz mit den besseren Ergebnissen verwendet werden.

4.4 Queueerkennung

Der Billardtrainer soll anhand des aktuellen Queuewinkels die aus einem Stoß resultierenden Kugellaufbahnen berechnen. Zusätzlich zu den Kugelpositionen wird die Queueorientierung in Bezug auf den Tisch benötigt. In diesem Kapitel werden zwei verschiedene Ansätze zur Queueerkennung vorgestellt.

Hough-Transformation

Da der Queue sich deutlich von dem Spielfeldhintergrund abzeichnet und im entzerrten Eingangsbild gerade Kanten hat, ist im Prototypen des BillardTrainers eine Queueerkennung mittels einer Hough-Transformation für Geraden implementiert. Diese wird z.B. in (Jähne, 2012, S. 551ff.) Die Queueerkennung wird auf dem gesamten Bild ausgeführt. Als resultierender Queuewinkel wird der Mittelwert aus den erkannten Geraden gebildet. Dieser Queuewinkel wird mit dem Mittelpunkt der weißen Kugel als Fixpunkt für die Berechnung der Kugellaufbahn des Spielballs genutzt. Der Ablauf der Queueerkennung ist in Abbildung 4.6 dargestellt.



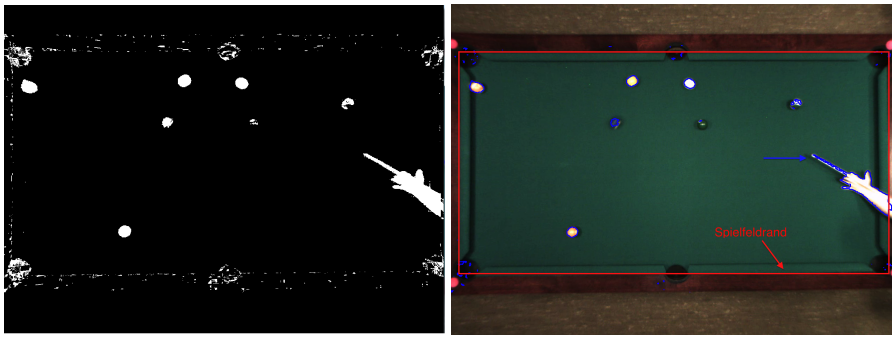
(a) Eingangsbild (b) Nach Kantenoperator

(c) Hough-Raum (d) Berechneter Winkel

Abbildung 4.6: Queueerkennung mit Linien Hough-Transformation

Hintergrundsubstraktion und Konturerkennung

Wie auch bei der Kugelerkennung ist eine Queueerkennung über Konturen möglich. Wird für die Kugelerkennung eine Hintergrundsubstraktion und Konturerkennung durchgeführt, kann das Ergebnis weiterverwendet werden, um den Queue zu finden. Es wurde beobachtet, dass die Kontur, welche den Queue enthält, im Gegensatz zu den Kugelkonturen immer den Spielfeldrand berührt. Auf diese Weise kann die Orientierung des Queue bestimmt werden.



(a) Vordergrundbild (b) Ergebnis nach Konturerkennung

Abbildung 4.7: Queueerkennung über Konturerkennung

Zusammenfassung

Eine Queueerkennung über die Erkennung von Geraden ist eine funktionierende Option. Es bringt jedoch auch einige Probleme mit sich. Zum Beispiel wird das Ergebnis stark verfälscht, wenn andere Kanten im Bild erkannt werden, wie zum Beispiel die Banden. Dies wirkt sich vor allem bei einem kleineren Billardtisch aus. Dieser Fehler wird auf Grund der Mittelwertbildung in das Ergebnis miteinbezogen. Des Weiteren kennt man zwar die Richtung des Queues, jedoch nicht die Position der Queuespitze. Dies war für den Prototypen nicht nötig, da mit einem zentralen Stoß am Kugelmittelpunkt gerechnet werden sollte, dies wurde für beide möglichen Stoßrichtungen betrachtet. Das Bestimmen des Queuewinkels mit Konturerkennung und Hintergrundsubtaktion ist problematisch, da nicht der Queue allein, sondern eine Kombination aus Mensch und Queue erkannt wird. Es ist zu prüfen, wie die Queueerkennung verbessert werden kann, sodass die Ergebnisse ausreichend genau sind.

5 Entwicklung

Um eine schnelle Austauschbarkeit vom Programmkomponenten für die Zukunft des Projektes zu gewährleisten, wird die Software modular gestaltet (siehe Tabelle 3.3). Sowohl die Basler Pylon Bibliothek als auch die OpenCV Bibliothek bieten Interfaces für C++. Da eine objektorientierte Programmierweise den modularen Aufbau von Software unterstützt, wird das Projekt in C++ realisiert.

Im Verlauf dieses Kapitels wird auf die einzelnen Teile des Softwarekonzeptes eingegangen. Es werden die für die Realisierung entwickelten Klassen anhand ihrer verschiedenen Aufgabengebiete vorgestellt. Das jeweillige Klassendiagramm wird am Ende des entsprechenden Abschnittes dargestellt. Am Ende dieses Kapitels folgt eine Zusammenfassung mit dem Klassendiagramm des gesamten Systems.

5.1 Verwaltung der einzelnen Teilaufgaben

Für die Verwaltung der einzelnen Module und die Realisierung der benötigten Abläufe ist die Klasse *MainSystem* verantwortlich, sie repräsentiert das BillardTrainer-System als Ganzes. Der BillardTrainer stellt zwei verschiedene Hauptfunktionen zur Verfügung, zum einen die Inbetriebnahme *startConfiguration*, zum anderen den zyklischen Hauptablauf *mainGame*. Außerdem wird eine Möglichkeit zum Speichern und Laden der Parameter bereitgestellt. Die Klasse verwaltet die für den BillardTrainer benötigten Objekte, welche im Folgenden näher erläutert werden. In Abbildung 5.1 ist das Klassendiagramm der Klasse *MainSystem* zu sehen.

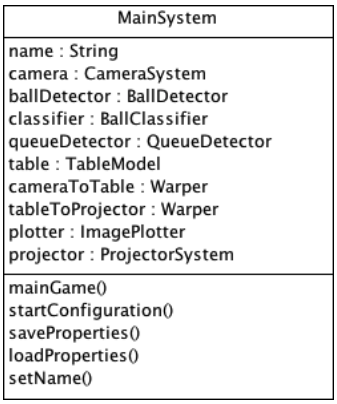


Abbildung 5.1: Klassendiagramm der Klasse *MainSystem*

5.2 Einlesen und Entzerren des Bildes

Der BillardTrainer benötigt zum Bewerten der Spielsituation eine entzerrte Aufnahme einer Spielsituation eines Billardspiels. Für das Einlesen und Korrigieren der Bilddateien ist die Klasse *CameraSystem* verantwortlich. Sie bietet die Möglichkeit ein Bild aus verschiedenen Quellen einzulesen und mit bekannten Kameraparametern zu korrigieren. Die Bildauflösung, die Kameraparameter für die Entzerrung und die Art der Ansteuerung sind dabei parametrierbar.

Um die Art der Ansteuerung möglichst flexibel zu gestalen, wird ein Interface *ICamera-Control* für die Realisierung der Ansteuerung verwendet. Auf diese Weise kann die Art der Ansteuerung auch während der Laufzeit des Programms verändert werden. Es werden drei verschiedene Ansteuerungsarten bereitgestellt. Die Ansteuerung über die Funktionen der OpenCV Bibliothek ist für die meisten Kameratypen einsetzbar und wird in der Klasse *CameraControlOpenCV* implementiert. Sie ist auch für die verwendete Baslersteuerung anwendbar, jedoch bietet die Ansteuerung über die Basler Pylon Library verschiedene Vorzüge, wie bessere Einstellungsmöglichkeiten oder eine schnellere Einlesezeit der Bilddaten. Diese Ansteuerung wird in der Klasse *CameraControlBasler* realisiert. Als dritte Möglichkeit wird des Einlesen von Videodaten in der Klasse *VideoInput* realisiert. Sie ermöglicht es die verschiedenen Objekterkennungsalgorithmen mit identischen Videomaterial zu testen. Für die Erzeugung der Objekte vom Typ ICameraControl ist die Klasse *CameraControlFactory* verantwortlich. Sie implementiert das in (Helm u. a., 2011, S. 107ff.) vorgestellte Fabrikmuster.

Für die Inbetriebnahme ist die Funktion *startConfiguration* verantwortlich, welche die benötigten Parameter vom Nutzer abfragt und die Kalibrierungsfunktion über die Klasse *CameraCalibrator* zur Verfügung stellt. Die Bestimmung der Kameraparameter wird mit den in OpenCV enthaltenen Funktionen nach der in (Laganière, 2011, S. 291ff) beschriebenen Klasse realisiert. Für die Kalibrierung wird ein Kalibrierungsmuster in Schachbrettoptik benötigt.

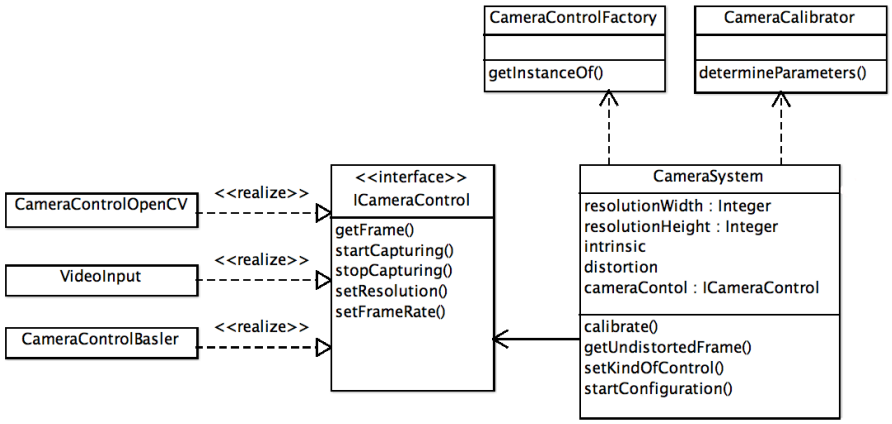


Abbildung 5.2: Klassendiagramm Bildgewinnung

5.3 Ermittlen des Tischzustandes

5.3.1 Kugelerkennung

Die Erkennung der Kugelpositionen wir im entzerrten Eingangsbild ausgeführt. Die Klasse *BallDetector* implementiert Funktionen, um die Kugelmittelpunkte in dem enzerrten Eingansbild zu lokalisieren. In der Klasse werden alle Parameter gespeichert, die für die Kugeldetektion notwendig sind. Für beide Verfahren soll die zu untersuchende Bildfläche auf den Bildbereich, in dem sich der Tisch befindet, begrenzt werden können. Dies wird mit der Funktion *setROI* realisiert.

Um eine Verleich der beiden in Abschnitt 4.3 vorgestellten Methoden vorzunehmen werden beide implementiert und in Abschnitt 6.3 getestet.

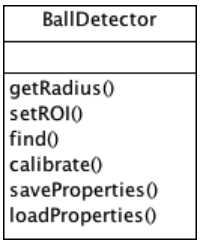


Abbildung 5.3: Klassendiagramm *BallDetector*

Kugelerkennung über Konturerkennung

Die Kugelerkennung über Konturerkennung wird in Anlehnung an die in Abschnitt 4.3 vorgestellte Art implementiert.

Das Eingangsbild wird mithilfe einer Hintergrundsubtraktion in Hintergrund und Vordergrund segmentiert. Dabei wird nur der Wert des Farbtonkanals im HSV-Raum für die Segementierung berücksichtigt, da dies den Rechenaufwand verringert. Um Fehler aufgrund von Rauschen zu unterdrücken, kann ein Erosionsfilter mit einstellbarer Filtergröße auf das Ergebnis der Segmentierung angewendet werden. Die Erosion ist eine morphologische Operation, welche z.B in (Tönnies, 2005, S. 277 ff.) beschrieben ist. Bei einem Binärbild führt die Erosion zu einer Verkleinerung von Gebieten mit dem Wert 1.

Das resultierende Bild wird auf Konturen untersucht. Um zu ermitteln, ob die gefundene Kontur eine Kugel ist, wird die Fläche der Kontur berechnet. Ist diese größer als die vom Benutzer eingestellte minimale Kugelfläche *minAreaBall* und kleiner als ein die Kugel umgebendes Quadrad der Größe d^2 wird angenommen, dass es sich bei der gefundenen Kontur um eine Kugel handelt. Es wird der Mittelpunkt des kleinsten die Kontur enschließenden Kreises, als Mittelpunkt der gefundenen Kugel angenommen.

Liegen Kugeln sehr dicht beieinander wird eine gemeinsame Kontur erkannt. Dies ist ein bekanntes Problem des verwendeten Verfahrens (vgl. Cash, 2003) und wird bei Müller (1998) mit einem Erosionsfilter behoben. Da eine zu starke Errosion des segmentierten Bildes das Verschwinden von Kugeln verursachen kann, wird die weitere Erosion nur auf Gebiete angewendet, in welchen Konturen mit einer Fläche größer als d^2 liegen. Diese Gebiete werden nach der Erosion auf Konturen untersucht und passende Konturen werden, zu den im ersten Schritt gefundenen Konturen, hinzugefügt.

Der komplette Algorithmus ist in Algorithm 1 beschrieben. Das Hintergrundmodell wird aus mehreren aufgenommenen Bilder des leeren Billardtisch durch eine Mittelwertbildung der Bilder bestimmt. Veränderbare Parameter für die Kugelerkennung sind der Kugeldurchmesser d, die minimale Fläche *minAreaBall*, die eine erkannte Kontur haben muss um eine Kugel zu sein , die Filtergröße des angewendeten Erosionsfilter *FilterSize* und der Abstand *delta* für die Segmentierung.

Algorithm 1 Kugelerkennung mit Konturerkennung

$mask \leftarrow calcSegment(image, backgroundModel)$
$mask.errode(nFilterSize)$
$contours \leftarrow findContours(mask)$
for $i = 0 \rightarrow contours.size()$ **do**
  if $contour[i].area > ballAreaMin$ **and** $contour[i].area < d^2$ **then**
    $center \leftarrow minEnclosingCircle(contour[i])$
  end if
  {–optinales Splitten der Konturen mit Erosion–}
  if $contour[i].area > d^2$ **then**
    $image.setROI(boundingBox(contour[i]))$
    $ROI.errode()$
    $contours.add(findContours(ROI))$
  end if
end for

Kugelerkennung über Hough-Transformation

Für die Kugelerkennung mit Hough-Transformation wird die Funktion *HoughCircles* der OpenCV Bibliothek benutzt. Die implementierte Methode ist eine auf Geschwingikeit optimierte Methode und teilt das Problem der Kreisfindung in zwei Phasen ein. Die Methode ist in Yuen u. a. (1990) beschrieben. Für die Nutzung der Methode wird das Eingangsbild in ein Grauwertbild gewandelt.

Als Parameter können der Kugelradius r, der Schwellwert des Akkumulators *threshCircle*, der Schwellwert des integrierten Canny-Filers *threshCanny* und der erlaubte Abstand zu dem angegebenen Radius *deltaRadius* angegeben werden. Als Minimale Distanz zwischen zwei erkannten Kreisen wird $2r$ angegeben.

Für die Kugelerkennung mit Hough-Transformation wird eine automatische Kalibrierungsfunktion bereitgestellt. Ziel dieser Funktion ist es, die Parameter zu bestimmen, sodass eine

sichere Erkennung aller Kugeln möglich ist. Dabei muss der Nutzer die Kugeln in festgelegte Bereiche auf dem Tisch legen. Auf diese Weise, können die Bildabschnitte sortiert, nach der enthaltenen Kugel übergeben werden. Der Radius der Kugeln wird anhand des erkannten Radius der ersten Kugel festgelegt. Die Parameter werden Schrittweise verändert, bis in jedem Bildabschnitt eine Kugeln gefunden wird.

Algorithm 2 Kalibrierungsfunktion Kreiserkennung HoughCircles

```
while i < 16 do
  circles ← ballDetector.find(imageParts[i]))
  if numberOf(circles) < 1 then
    adjustParameters()
    i ← 0
  end if
end while
```

5.3.2 Kugelklassifikation

Um den Spielball zu erkennen, werden die Kugeln nach Art klassifiziert. Da in der Zukunft des Projekts eine Unterscheidung zwischen vollen und halben Kugeln notwendig ist, wird eine Unterscheidung in die Kugeltypen *Black*,*White*,*Solid* und *Striped* implementiert. Dabei ist die Erkennung des Spielballs am wichtigsten.

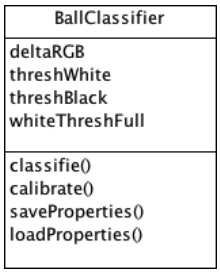


Abbildung 5.4: Klassendiagramm *BallClassifier*

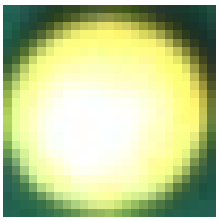


Abbildung 5.5: Kugelgebiet

Die Klasse *BallClassifier* stellt die Funktionen für die Kugelklassifikation zur Verfügung. Das Klassendiagramm ist auf Abbildung 5.4 zu sehen. Um die Kugelart zu bestimmen, müssen dem Klassifizierer die Kugelpositionen, der Radius und das Bild übergeben werden, in welchem sie detektiert wurden. Um die Kugelart zu bestimmen wird für jede gefundene Kugel im Bildausschnitt der Kugel (siehe Abbildung 5.5) die Verteilung der enthaltenen weißem, schwar-

zen, und anderen Pixel bestimmt. Die Bewertung der Pixel findet im RGB-Farbraum statt, da schwarze und weiße Pixel sich dort durch gleiche Werte in jedem Farbkanal auszeichnen. Eine Unterscheidung zwischen schwarz und weiß wird über die Höhe des Wertes ausgeführt. Über den Anteil der jeweilligen Pixelanzahlen im betrachteten Kugelgebiet wird die Unterscheidung der Kugelarten wie in Algorithm 3 beschrieben ausgeführt. Dafür werden Grenzwerte für die Rate der weißen Pixel für den Spielball $threshWhite$, die Rate der schwarzen Pixel für die schwarze Kugel $threshBlack$, und die Hochstrate an weißen Pixeln für eine volle Kugel $whiteThreshFull$ benötigt.

Algorithm 3 Farbklassifizierung

```
blackRate ← blackPixel/anzPixel
whiteRate ← whitePixel/anzPixel
otherRate ← otherPixel/anzPixel
if whiteRate > threshWhite then
  balltype ← White
else
  if blackRate > threshBlack then
    balltype ← Black
  else
    if otherRate > whiteThreshFull then
      balltype ← Solid
    else
      balltype ← Striped
    end if
  end if
end if
```

Um den Kugelklassifikator zu kalibrieren, werden Bildabschnitte erstellt in welchem sich eine definierte Kugel befindet. Für die sichere Erkennung der weißen Kugel werden die Parameter $whiteRate$ und $deltaRGB$ solange angepasst, bis der Spielball als solcher erkannt wird. Für die schwarze Kugel wird entsprechendes mit den Parametern $blackRate$ und $deltaRGB$ ausgeführt. Wird eine halbe oder volle Kugel als schwarz oder weiß erkannt wird dies als korrekt akzeptiert um die Parameter nicht so zu ändern, dass keine sichere Erkennung der weißen und schwarzen Kugel mehr möglich ist. Für die Unterscheidung zwischen vollen und halben Kugeln wird der Parameter $whiteThreshFull$ angepasst.

5.3.3 Queueerkennung

Die Queueerkennung wird in der Klasse *QueueDetector* realisiert. Dieser sucht den Queue in dem ihm übergebenen Bildabschnitt. Das Klassendiagramm ist auf Abbildung 5.6 zu sehen.

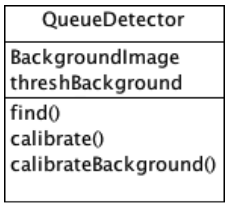


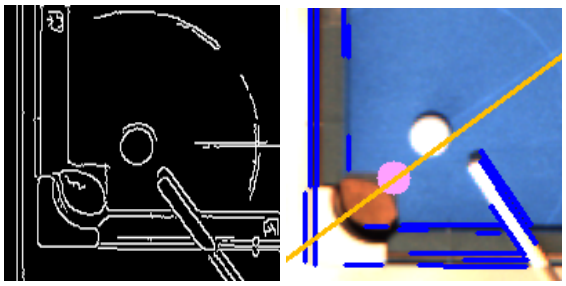
Abbildung 5.6: Klassendiagramm *QueueDetector*

Der zu untersuchende Bereich für die Queueerkennung kann stark begrenzt werden, da nur von einer Spielabsicht des Nutzers ausgegangen werden muss, wenn sich der Queue in der Nähe des Spielballs befindet. Ein Bereich sechsmal größer als der Durchmesser des Spielballs hat sich als geeignet erwiesen um den Queue zuverläsig zu finden.

Der in Abschnitt 4.4 vorgestellte Algorithmus zur Queueerkennung hat sich für den Prototypen als ausreichend bewiesen. Jedoch bestehen Probleme, wenn sich ein Kantenabschnitt in dem betrachteten Bildbereich befindet. Dies wirkt sich vor allem bei dem mobilen Hardwareaufbau aus und kann in Abbildung 5.7 betrachtet werden. Durch die eventuell gedrehte Lage des Billardtisches im entzerrten Eingangsbild, können fehlerhaft erkannte Banden auch nicht ohne weiters aus dem Suchbereich entfernt werden. Um dieses Problem zu verhindern, wird eine Segmentierung mit einer Hintergrundsubtraktion über die Grauwerte des Bildes im betrachteten Bildausschnitt ausgeführt. Dies wird in Abbildung 5.8 dargestellt.

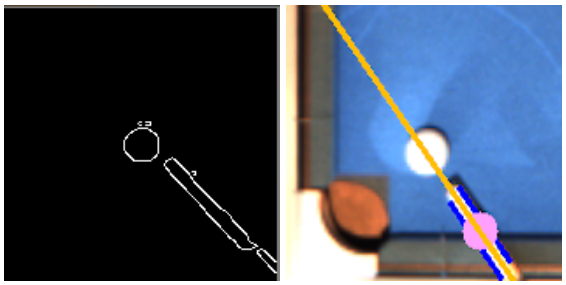
Durch die Reduzierung des betrachteten Bildausschnittes kann die Rechenzeit verkürzt werden. Um die Genauigkeit zu verbessern, werden die erkannten Linien in zwei Winkelbereiche eingeteilt und der mittlere Winkel aus dem jeweiligen Bereich für die Bestimmung des resultierenden Winkels benutzt. Auf diese Weise kann der Fehler bei der Mittelung reduziert werden, welcher entsteht, wenn in einem Winkelbereich sehr viel mehr Geraden gefunden wurden als in einem anderen. Da die Queuekanten nicht parallel sonder leicht zulaufend verlaufen haben die Queueseiten verschiedene Winkelbereiche.

Um die Richtung der Queuespitze zu bestimmen wird der Mittelpunkt aus den Eckpunkten der erkannten Liniensegmente gebildet. Die Koordinaten dieses Punktes werden zusammen mit dem erkannten Queuewinkel als Ergebnis zurückgegeben. Je nach Position der weißen Kugel wird der Queuewinkel entsprechend korrigiert.



(a) Kantenbild (b) Detektionsergebnis

Abbildung 5.7: Fehlerhafte Queuedetektion ohne Hintergrundsubtraktion



(a) Kantenbild (b) Detektionsergebnis

Abbildung 5.8: Queuedetektion mit Hintergrundsubtraktion

5.4 Simulation des Stoßes

Tischmodell

Bei dem Zusammenspiel der Hardwarekomponenten des Billardtrainers, gibt es zunächst zwei verschiedene Koordinatensysteme. Zum einen das Kamerakoordinatensystem, welchem das aufgenommene Bild zugrunde liegt, zum anderen das Koordinatensystem des Projektorbildes.

Für die Simulation des Stoßergebnisses wird ein Tischmodell eingeführt. Das Tischmodell, ist ein allgemeines Modell eines Billardtisches. Es besitzt eine individuelle Höhe und Breite

und damit auch ein eigenes Koordinatensystem. In Abbildung 5.9 sind die drei verschiedenen Koordinatensysteme im entzerrten Eingangsbild zur Verdeutlichung eingezeichnet.

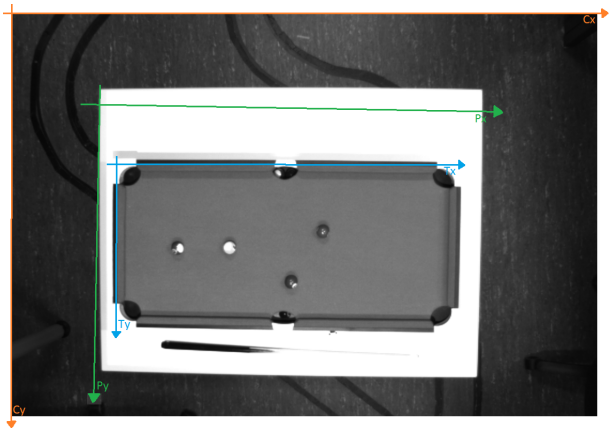


Abbildung 5.9: Vergleich der Koordinatensysteme

Das Tischmodell besitzt sechs Banden und sechs Taschen, sowie eine Anzahl von Kugeln. Die Taschen und Banden werden zunächst mit Rechtecken angenähert. Die Kugeln werden mit der Klasse *Ball* dargestellt. Für die Realisierung des Tischmodells wird die Klasse *TabelModel* erstellt. Sie bietet die Funktion Kugeln hinzuzufügen und einen Stoß zu simulieren. Als Ergebnis der Simulation werden die Eckpunkte der Kugeltrajektorien zurückgegeben. Das Klassendigramm des Tischmodells ist in Abbildung 5.10 dargestellt.

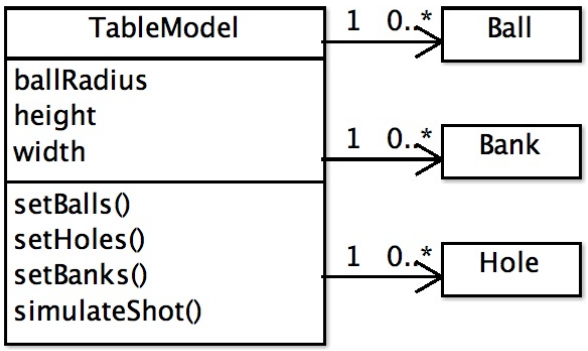


Abbildung 5.10: Klassendiagramm *TabelModel*

5.4.1 Überführung der Kugelkoordinaten

Für die Umrechnung zwischen zwei Koordinatensystemen ist jeweils ein Objekt der Klasse *Warper* verantwortlich. Das Klassendiagramm der Klasse *Warper* ist in Abbildung 5.11 zu sehen. Für die Überführung der Koordinaten wird eine perspektivische Transformation ausgeführt. Diese ist z.B. in (Jähne, 2012, S.83ff.)beschrieben. Es werden vier Punktkorrespondenzen für die Berechnung der Transformationsmatix benötigt. Die Klasse Warper berechnet anhand von vier Quellpunkten src und vier Zielpunkten dst die 3x3 Transformationsmatrix $\mathbf{T}$ nach Gleichung 5.1.

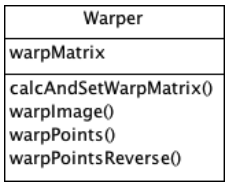


Abbildung 5.11: Klassendiagramm *Warper*

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{5.1}$$

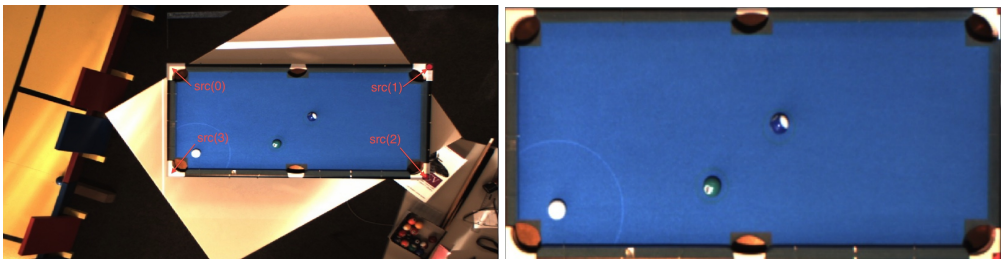
$$Mit : dst(i) = (x'_i, y'_i) src(i) = (x_i, y_i) i = 0, 1, 2, 3$$

Für die Berechnung der Transformationsmatrix zwischen dem Kamerakoordinatensystem und dem Tischkoordinatensystem sind die Positionen der Spielfeldeckpunkte im entzerrten Eingansbild geeignet, da über diese sowohl die Größe, als auch die Rotation des Billardtisches im Bild erkennbar ist. Die Eckpunkte werden beginnend von oben links im Uhrzeigersinn in aufsteigender Reihenfolge als Quellpunkte $src(0..3)$ für die Berechnung der Transformationsmatrix genutzt. Es wird davon ausgegangen, dass sich zwischen $src(0)$ und $src(1)$, sowie zwischen $src(2)$ und $src(3)$ die langen Seiten des Spielfeldes befinden. Für die Ermittlung der Zielpunkte müssen die Maße *height* und *width* des Tischmodells bekannt sein. Die Koordinaten der Zielpunkte werden wie folgt festgelegt:

- $dst(0) = (0, 0)$
- $dst(1) = (width, 0)$
- $dst(2) = (width, height)$
- $dst(3) = (0, height)$

Dabei sind die Parameter *height* und *width* frei wählbar. Um eine leichte Übertragung zwischen dem von System bestimmten Koordinaten und den realen Positionen auf dem Billardtisch herzustellen, bietet sich eine Angabe in Relation zu den realen Maßen des Tisches an. Bei einer Relation von 1 Pixel zu 1cm oder 1mm können die ermittelten Positionswerte mit einfachen Hilfsmitteln nachgeprüft werden.

Die Transformation ist in Abbildung 5.12 mit allen Punkten des Eingansbildes ausgeführt um die Auswirkungen darzustellen. Im Verlauf des Programms werden nur die benötigten Punkte transformiert, da dies weniger Rechenzeit beansprucht.



(a) Quellkoordinaten im Eingangsbild (b) Tranformiertes Bild

Abbildung 5.12: Von Eingansbild zum Tischmodell

5.4.2 Automatische Tischerkennung

Um die Inbetriebnahme zu erleichtern, wird in Abschnitt 3.2 gefordert, dass die benötigten Referenzpunkte in einem geführten und soweit dies möglich ist automatischen Prozess ermittelt werden. Die Tischposition kann dabei am leichtesten über die Position der Spielfeldecken ermittelt werden, da dort ein Übergang von der Spielfeldfarbe zu der Umrandung stattfindet. Die Suche der Referenzpunkte wird in der Klasse *TableDetector* realisiert.

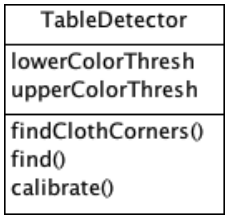


Abbildung 5.13: Klassendiagramm *TableDetector*

Als Orientierung für die Entwicklung der Tischerkennung gilt das in Abschnitt 4.2 vorgestellten merkmalsbasierten Verfahren. Da die Spielfeldfarbe jedoch vor der Erkennung nicht bekannt ist, muss diese zunächst ermittelt werden.

Ermitteln der Spielfeldfarbe

Um den Billardtisch in einem Bild zu identifizieren, muss sich dieser stark von Hintergrund unterscheiden. Es wurde die Annahme getroffen, dass der Untergrund auf dem sich der Billardtisch befindet nicht die selbe Farbe hat wie das Tuch des Tisches. Befindet sich eine große homogengefärbte Fläche im Bild, wird diese als dominate Farbe erkannt. Dies ist der Fall, wenn die Spieltuchfarbe des Billardtisches im Bild gleichmäßig gefärbt ist und das Spieltuch mehr als die Hälfte der Bildfläche einnimmt. Bei einem von oben bildfüllend aufgenommenen Billardtisch trifft dies zu. Damit ist bei einer genügend großen Aufnahme des Tisches davon auszugehen, dass die Tuchfarbe die dominante Farbe des Bildes ist. Die Bestimmung der dominanten Farbe findet im HSV-Farbraum statt.

Um die Bestimmung zu verbessern, besteht die Möglichkeit der Funktion untere und obere Schwellwerte für jeden Farbkanal für die Filterung vor der Bestimmung zu übergeben. Farbwerte die außerhalb dieser Schwellwerte liegen werden nicht für die Bestimmung berücksichtigt. Da die Spielfeldfarbe in der Regel eine kräftige Farbe ist, wird als unterer Schwellwert standardmäßig für den Sättigungs- und Helligkeitskanal 50 angenommen, sofern nichts anderes angegeben ist. Die Schwellwerte sind in Abbildung 5.14 eingezeichnet für einen beliebigen Farbton. Es wird ein dreidimensionales Histogramm für alle gültigen Werte ermittelt mit einer Auflösung von je zehn Werten für Farbton, Helligkeit und Sättigung. Zurückgegeben wird der am häufigsten vorkommende Wert.

Finden der Spielfeldecken

Für die automatische Detektion des Billardtisches wird Algorithmus 4 angewendet. Nach der Ermittlung der dominanten Bildfarbe wird das Bild für das Finden der Spielfeldecken (*findClothCorners*) eingeschränkt. Ist das Auffinden der Ecken nicht gelungen, wird der Farbbereich erweitert. Dies geschieht jedoch maximal 20 Mal um die Berechnungszeit einzuschränken. Bei vier erkannten Eckpunkten werden die Eckpunkte sortiert und die Tischerkennung wird erfolgreich beendet.

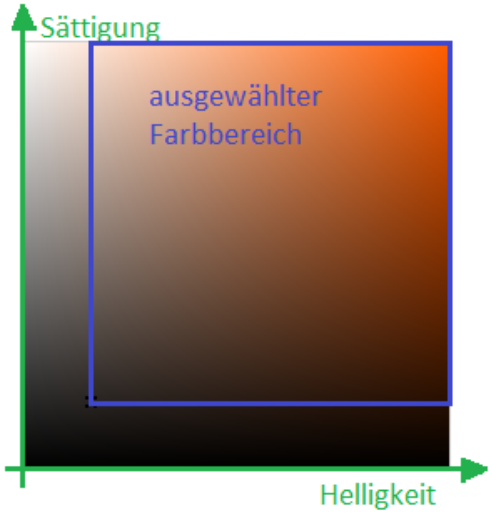
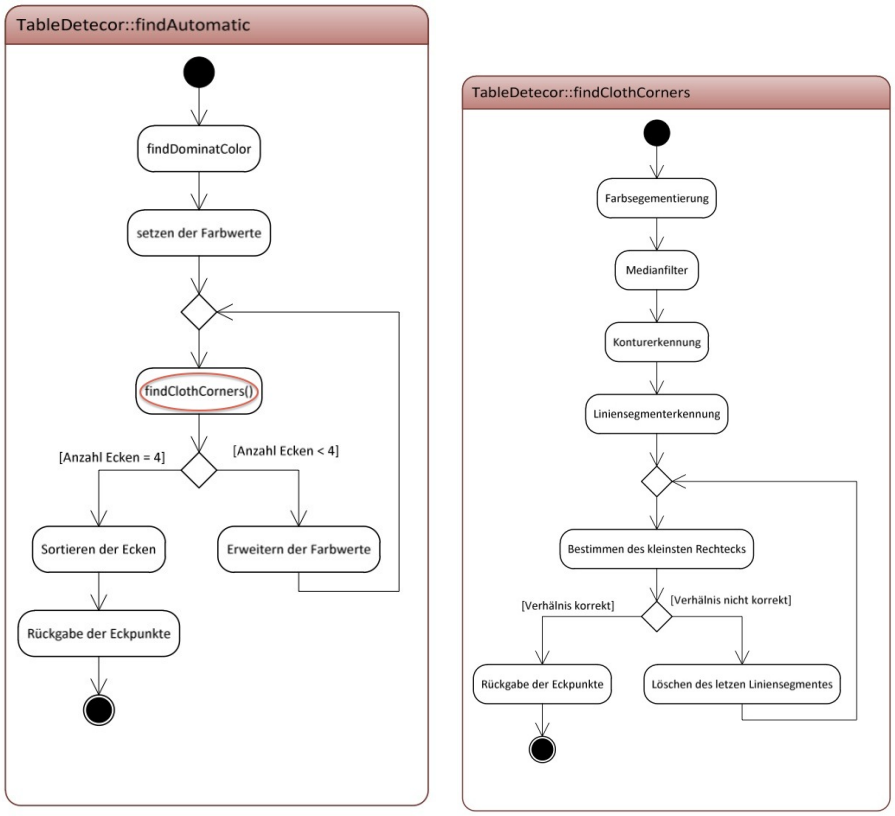


Abbildung 5.14: Standardschwellwert der Funktion findDominantColor

Um die Position der Spielfeldecken zu bestimmen, wird zunächst das Eingangsbild mit den eingestellten Schwellwerten gefiltert. Um das Bild zu glätten, wird ein Medianfilter auf das segmentierte Bild angewendet. Da es sich bei dem Tisch um eine zusammenhängende Kontur handelt, wird eine Konturerkennung ausgeführt um die äußerste zusammenhängende Kontur festzustellen. Für das Unterschieden der Spielfeldkanten von anderen Konturen wird eine Liniensegmenterkennung basierend auf der Hough-Transformation auf das gezeichnete Konturbild angewendet. Die in OpenCV implementierte Liniensegmenterkennung ist in Matas u. a. (2000) beschrieben.

Um die Spielfeldfläche zu lokalisieren, wird das Rechteck mit der minimalsten Fläche bestimmt, welches alle Liniensegemente enthält. Dieses wird im Anschluss auf seine Längenverhältnisse geprüft, da bei der Aufnahme eines Billardtisches von oben davon ausgegangen werden kann das die Spielfeldbegrenzungen ein Verhälnis nahe 2:1 hat. Entspricht das erkannte Rechteck nicht diesen Vorgaben, wird das schwächste erkannte Liniensegment entfernt und die Spielfelderkennung mit den übrigen Liniensegmenten wiederholt. Bei der Erfüllung der Voraussetzungen werden die Eckpunkte des Spielfeldes an die automatische Tischerkennung zurückgegeben. Der Ablauf kann in Abbildung 5.15b verfolgt werden.



(a) Tischerkennung (b) findClothCorners

Abbildung 5.15: Aktivitätsdiagramme zur Tischerkennung

Die ermittelten Spielfeldeckpunkte werden an den Warper übergeben um die Transformationsmatix zu bestimmen. Auf diese Weise können die Kugelkoordinaten in das Tischkoordinatensystem überführt werden. Der erkannte Queuewinkel wird transformiert, in dem zwei Punkte auf der Geraden transformiert werden und der neue Winkel zwischen diesen bestimmt wird.

5.4.3 Simulation des Stoßes

Nachdem dem Tischmodell alle Objektparameter übergeben worden sind, kann mit der Simulation des Stoßergebnisses begonnen werden. Für die Simulation wird zunächst eine möglichst einfach zu implementierende Physik genutzt. Es findet eine Beschränkung auf das Wesentliche statt. Es gibt zwei verschiedene Ereignisse, die eintreten können. Zum einen die Kollision einer Kugel mit einer Bande, zum anderen die Kollision einer Kugel mit einer anderen. Dabei wird

Algorithm 4 TableDetector::findAutomatic

```
maincolor ← TableDetector :: findDominantColor(image)
delta ← 25
lowerValues ← (maincolor.hue − delta, 50, 50)
upperValues ← (maincolor.hue + delta, 255, 255)
i = 0
corners =
while corners.size < 4 AND i < 20 do
  corners = this− > findClothCorners(frame)
  if corners.size < 4 then
    lowerValues− = (3, 1, 1)
    upperValues+ = (3, 0, 0)
  end if
  i = i + 1
end while
```

immer nur die aktuell betrachtete Kugel als bewegt angenommen, die zweite Kugel wird als ruhend angesehen, da das gezielte Anspielen einer Kugel mit dem Spielball der Regelfall beim Billardspiel ist.

Die Simulation wird schrittweise durchgeführt. Dabei wird für jede Kugel nacheinander gepüft, ob sich die Kugel in Bewegung befindet, ist dies der Fall wird die Position der Kugel für den nächsten Schritt nach Gleichung 5.2 berechnet.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i + v_x \\ y_i + v_y \end{bmatrix} \tag{5.2}$$

Für die berechnete Position wird zunächst geprüft, ob sich die Position innerhalb einer Tasche befindet. Ist dies der Fall wird der Parameter $isOnTable$ auf $false$ gesetzt. Die Kugel wird nicht mehr für weitere Rechnungen betrachtet. Ist $isOnTable$ $true$ wird zunächst geprüft, ob die Kugel mit einer Bande kollidiert. Als Kollision wird es gewertet, wenn die neu berechnete Position der Kugel innerhalb des rechteckigen Banden- oder Taschenbereich liegt. Ist dies der Fall wird die Bewegungsparameter v_x und v_y je nach Bandenart neu berechnet. Für horizontale Banden wird v_y invertiert, für vertikale Banden v_x.

Für die Prüfung auf eine Kollision der Kugel $k1$ mit einer anderen Kugel $k2$ wird zunächst geprüft, ob die Rechtecke um die Kugeln sich schneiden. Ist dies der Fall wird genau gepüft, ob

die Kugeln sich berühren. Dies ist der Fall, wenn der Abstand zwischen den Mittelpunkten kleiner als der Durchmesser der Kugeln ist. Die neuen Bewegungsparameter werden nach Gleichung 5.3 bis Gleichung 5.5 berechnet. Findet eine Kollision statt, wird der Kollisionspunkt der Trajektorienliste der betroffenen Kugel hinzugefügt. Auf diese Weise lässt sich der Verlauf jeder Kugel anhand der Punkte in der Trajektorienliste verfolgen. Der zeitliche Verlauf des simulierten Stoßes wird durch dieses Modell nicht berücksichtigt, genauso wenig die Reibungsverluste bei der Bewegung oder die Energieverluste beim Zusammenstoß der Kugeln. Dieses Modell ist ausreichend um die Fähigkeiten des BillardTrainers zu demonstierten. Im Zukunft, kann durch die Modularität das Modell leicht erweitert und die Stoßsimulation verändert werden. Die Stoßsimulation ist stark vereinfacht, bezieht jedoch alle Kugeln im Gegensatz zu der Version des BillardTrainer-Prototypen mit ein.

$$mz = \frac{|(y(k1) - y(k2))|}{|(x(k1) - x(k2))|} \tag{5.3}$$

$$\begin{bmatrix} v_x(k2) \\ v_y(k2) \end{bmatrix} = \begin{bmatrix} v_x(k1) \\ v_x(k1) * mz \end{bmatrix} \tag{5.4}$$

$$\begin{bmatrix} v_x(k1) \\ v_y(k1) \end{bmatrix} = \begin{cases} \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \text{falls zentraler Stoß} \\ \begin{bmatrix} v_x(k1) \\ v_x(k1) \cdot mz^-1 \end{bmatrix}, & \text{sonst} \end{cases} \tag{5.5}$$

5.5 Ausgabe des Ergebnisses

Die Simulation des Stoßes gibt als Ergebnis pro Kugel alle Punkte an, an welchen sich die Fortbewegung der Kugel geändert hat. Zwischen den Punkten wird ein linearer Verlauf angenommen. Die Koordinaten der Punkte liegen im Tischkoordinatensystem vor.

Die Klasse *ImagePlotter* bietet die Funktionen Kugeln, Banden und Taschen des Tischmodells, sowie die Kugeltrajektorien zu dem Ausgabebild hinzuzufügen (siehe Abbildung 5.16). Durch das schrittweise Hinzufügen von Objekten kann variabel auf den Tischzustand reagiert werden, wird z.B. kein Queue erkannt, müssen auch keine Trajektorien hinzugefügt werden. Das Ausgabebild wird im Tischkoordinatensystem erstellt.

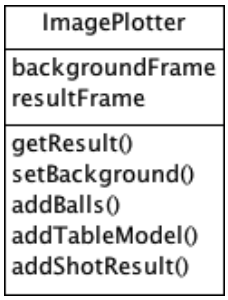


Abbildung 5.16: Klassendiagramm *ImagePlotter*

Um die Kugelpositionen und -arten für den Nutzer erkenntlich zu machen, werden folgende Farben als Standardfarben gewählt, schwarz für Kugeln des Typs *Black*, weiß für Kugeln des Typs *White*, grün für Kugeln des Typs *Striped* und blau für Kugeln des Typs *Solid*. Als Hintergrund wird ein Bild in der Größes des Tischkoordinatensystems mit der Hintergrundfarbe des Projektors genutzt. Um den Bereich in dem der Queue gesucht wird für den Nutzer besser kenntlich zu machen, wird der Kreis um die weiße Kugel entsprechend vergrößert. Die anderen Kreise sind doppelt so groß, wie der angenommene Kugelradius. Ein beispielhaftes Ausgabebild ist in Abbildung 5.18b zu sehen.

Für den Projektor wird die Klasse *ProjektorSystem* eingeführt. Das Klassendiagramm ist in Abbildung 5.17 zu sehen. Über die Parameter *xPos* und *yPos* kann die Anzeigeposition des Projektorfensters angepasst werden. Auf diese Weise kann das Bild, z.B bei einer Multimonitor-Nutzung, auf den Projektorbildschirm verschoben werden. Das Projektorfenster wird mit den Möglichkeiten der in OpenCV enthaltenen Funkionen erschaffen. Über *width* und *heigt* kann die Auflösung des Projektorfensters angepasst werden. Mit dem Parameter *backgroundColor* kann auf verschiedene Lichtverhältnisse reagiert werden. Der Wert ist dabei von 0 (schwarz) bis 255 (weiß) frei einstellbar. Auf diese Weise kann das Tischfeld allein mit einem lichtstarken Projektor beleuchtet werden. Durch die Beleuchtung von oben wird die Schattenbildng minimiert.

Um das im Tischkoordinatensystem erstellte Bild in das Projektorkoordinatensystem zu wandeln, wird ein weiteres Objekt der Klasse Warper benutzt. Die Referenzpunke werden durch eine Kalibrierungsmethode des Projektors bestimmt. Es werden Koordinaten im Tischkoordinatensystem als Ziel vorgegeben. Auf dem Projektor wird ein Kreis gezeichnet beginnend in der Mitte des Bildes. Dieser wird solange verschoben bis der Mittelpunkt des Kreises das Ziel im Tischkoordinatensystem erreicht. Dies wird mit vier verschiedenen Zielpunkten

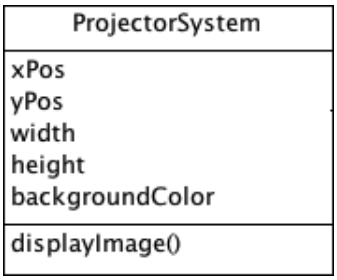
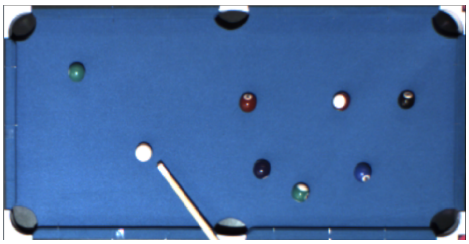
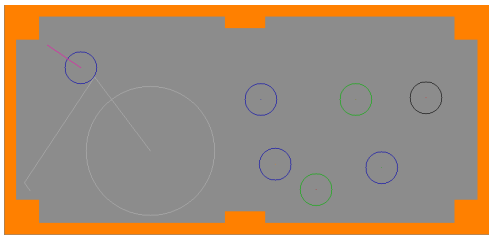


Abbildung 5.17: Klassendiagramm *ProjectorSystem*

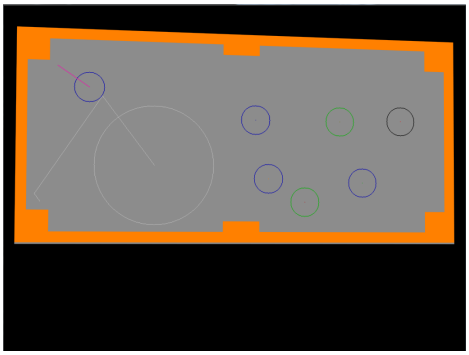
wiederholt. Auf diese Weise können vier verschiedene Referenzpunkte gewonnen werden. Die Transformation des Bildes geschieht nach den oben genannten Prinzipien. Ein Beispiel eines transformierten Ergebnisbildes ist in Abbildung 5.18c gezeigt. Durch den schwarzen Hintergrund wird nur das Tischfeld durch den Projektor erhellt.



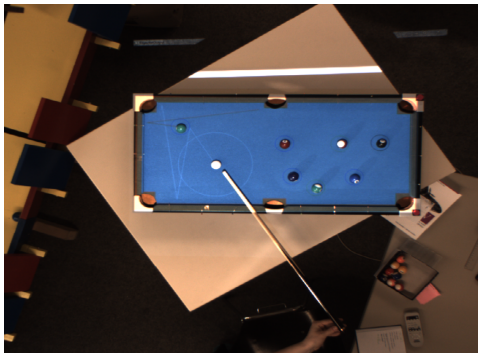
(a) Eingangsbild im Tischkoordinatensystem



(b) Ausgabebild im Tischkoordinatensystem



(c) Ausgabebild im Projektorkoordinatensystem



(d) Projektion des Ausgabebildes auf den Billardtisch

Abbildung 5.18: Erstellen des Ausgabebildes und Anzeigen auf dem Tisch

5.6 Zusammenfassung

Bei der Strukturierung des Programms wird das Hauptaugenmerk auf die Anpassbarkeit der entwickelten Module gelegt. Auf diese Weise kann die Software an die unterschiedlichsten Hardwarekomponenten angepasst werden. Die einzelnen Module stellen eine Kalibrierungsfunktion zur Verfügung um den Inbetriebnahmeprozess zu erleichtern. Diese Kalibrierungsfunktionen werden in einem strukturierten Prozess von der Hauptkalibrierungsfunktion der Klasse MainSystem ausgeführt. Der Ablauf dieses Inbetriebnahmeprozesses wird im folgenden Kapitel erläutert. Durch den modularen Aufbau ist es in Zukunft möglich die Software schnell anzupassen oder zu verändert. Man könnte zum Beispiel das Billardtischmodell durch ein Minigolfmodell austauschen, um den Kugelverlauf beim Minigolf zu berechnen.

Ein Überblick des gesamten Systems ist in dem Klassendigramm in Abbildung 5.19 zu sehen.

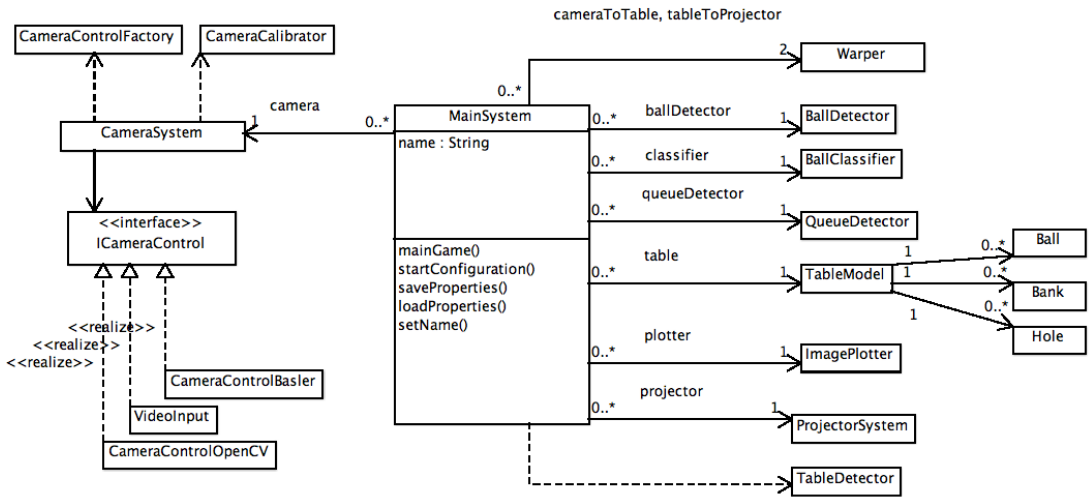


Abbildung 5.19: Klassendiagramm BillardTrainer Überblick

6 Realisierung und Test

Im Folgenden wird die Realisierung des in dieser Arbeit weiterentwickelten BillardTrainers vorgestellt. Zunächst wird ein Überblick über die implementierten Abläufe gegeben. Im Anschluss werden einzelne Bereiche näher betrachtet und die erreichten Funktionalitäten mit den Anforderungen verglichen.

6.1 Programmablauf

In der BillardTrainer-Software werden zwei verschiedene Abläufe realisiert, zum einen der Hauptablauf, zum anderen die geführte Inbetriebnahme. Diese erfüllen die in Unterabschnitt 3.1.1 genannten Anforderungen. Die beiden Abläufe werden an dieser Stelle vorgestellt.

Hauptablauf

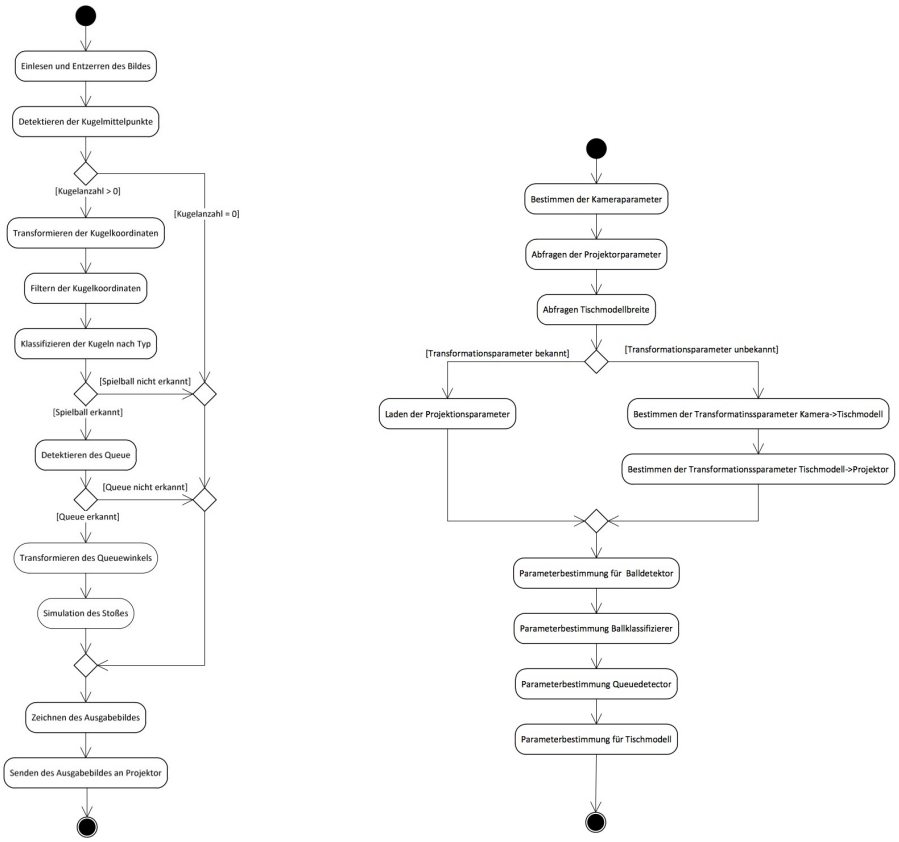
Der Hauptablauf wird zyklisch ausgeführt, solange ein Spiel läuft. Er ist in Abbildung 6.1a dargestellt und wird hier beschrieben.

Das aufgenommene Bild wird in das Programm eingelesen und durch die in OpenCV implementierten Methoden der Kamerakalibierung entzerrt. Im Anschluss findet die Detektion der Kugeln statt. Die erkannten Kugelkoordinaten werden in das Tischkoordinatensystem transformiert. Die Kugelpositionen werden geprüft und können an dieser Stelle gefiltert werden. Danach findet die Klassifikation der Kugeltypen statt. Bei einer erfolgreichen Indentifizierung des Spielballs wird die Queueerkennung ausgeführt. Der erkannte Queuewinkel wird in das Tischkoordinatensystem umgerechnet und an das Tischmodell übergeben. Dieses kann mit den erhaltenen Daten der Detektoren die Simulation des Stoßes ausführen. Das Ergebnisbild

wird abhängig von den erkannten Objekten erzeugt und mit dem Projektor auf dem Spielfeld angezeigt.

Inbetriebnahme

Die Inbetriebnahme ist in einem geführten Prozess realisiert. Dieser ermittelt die benötigten Parameter vom Nutzer oder selbständig. Die Interaktion mit dem Nutzer findet über die Konsole und über eine graphische Oberfläche statt. Der Inbetriebnahmeprozess ermittelt die Parameter in der benötigten Reihenfolge. Der Ablauf der Inbetriebnahme ist in Abbildung 6.1b zu sehen.

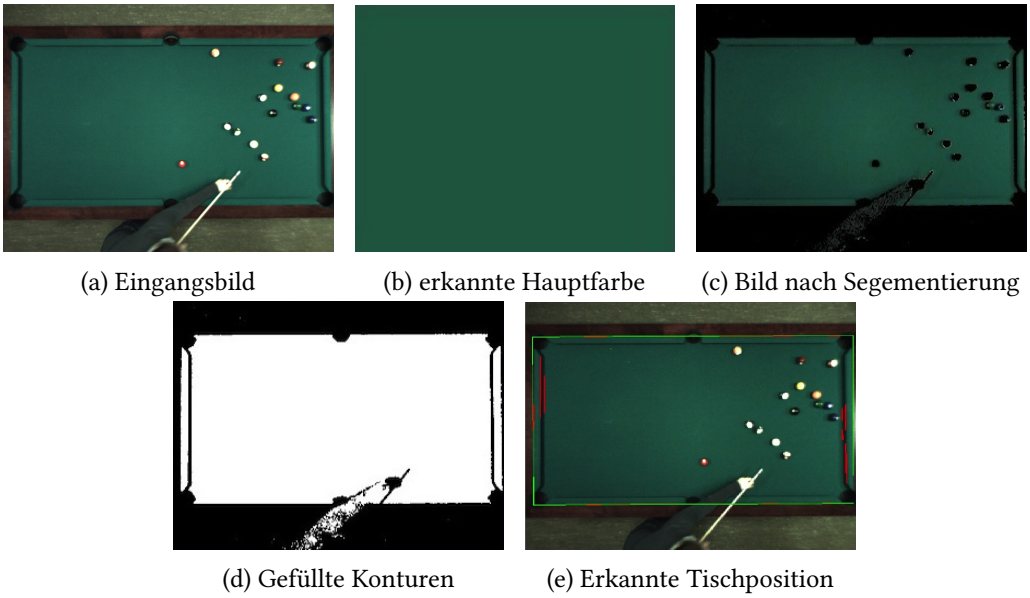


(a) Hauptablauf (b) Inbetriebnahme

Abbildung 6.1: Programmablauf

6.2 Tischerkennung

Für die Inbetriebnahme wird eine automatische Tischerkennung zur Verfügung gestellt.Die einzelnen Schritte können in Abbildung 6.2 nachvollzogen werden (vgl. Unterabschnitt 5.4.2).



(a) Eingangsbild (b) erkannte Hauptfarbe (c) Bild nach Segementierung

(d) Gefüllte Konturen (e) Erkannte Tischposition

Abbildung 6.2: Ablauf Tischerkennung

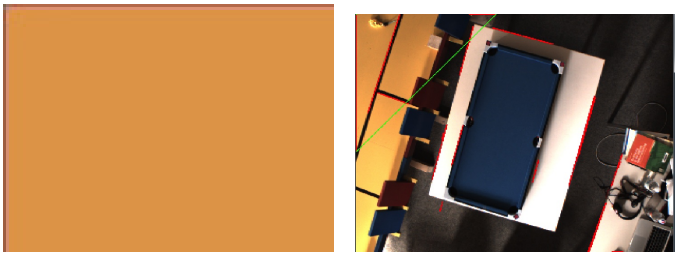
Um die Tischerkennung zu testen, wird der Tischdetektor mit 20 Bildern des stationären Systems getestet. Das Kamerabild ist von oben aufgenommen und der Tisch ist bildfüllend abgebildet.

Ergebnisse

Der Tisch wird in 20 von 20 Bildern erkannt. Die durchschnittliche Abweichung der erkannten Eckkoordinaten beträgt 2,5 Pixel (ca. 5 mm). Damit ist die Tischerkennung für das stationäre Hardwaresystem ausreichend zuverlässig und genau. Durch die Festlegung auf einen rechteckigen Bereich, wird keine Formveränderung geduldet.

Da der Billardtisch im mobilen Hardwaresystem nicht bildfüllend abgebildet ist und die Spielfläche nur ca. 20 Prozent des Bildes bedeckt, entspricht die bestimmte dominate Farbe nicht der Spielfeldfarbe. Dieses Problem tritt vor allem bei farbigen Flächen im Hintergrund

auf, da diese nicht durch die Filterung im Sättigungs- und Helligkeitsbereich entfernt werden. Eine fehlerhafte Erkennung ist auf Abbildung 6.3 gezeigt.



(a) erkannte Hauptfarbe (b) erkannte Liniensegmente

Abbildung 6.3: Fehlerhafte Tischerkennung

Um dieses Problem zu beheben, wird eine graphische Oberfläche erstellt, mit welcher der Inbetriebnehmer die Farbparameter für die Tischerkennung manuell verändern kann. Auf diese Weise kann ein falsches Ergebnis der automatischen Tischerkennun behoben werden. Dem Nutzer werden dabei für die aktuellen Parameter die Eckpunkte des Detektionsergebnisses angezeigt. Außerdem wird der Farbfilter auf das Eingangsbild angewendet, sodass der Nutzer direkt die Auswirkung der eingestellten Parameter erkennt. Die graphische Oberfläche ist in Abbildung 6.4 zu sehen.

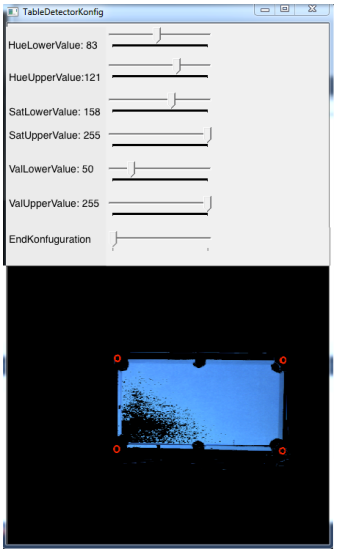


Abbildung 6.4: Graphische Oberfläche der Tischerkennung

6.3 Kugelerkennung

Für die Kugelerkennung sind zwei verschiedene Algorithmen implementiert worden. Um zu entscheiden, welcher der beiden in Unterabschnitt 5.3.1 vorgestellten Algorithmen die Anforderungen bezüglich Rechengeschwindigkeit , Genauigkeit und Zuverlässigkeit besser erfüllt, wird ein Test für jeden dieser Punkte durchgeführt. Der Test wird mit identischen Bildmaterial durchgeführt und die Suchbereich wird auf die Bildregion des Spielfeldes beschränkt.

Der Test wird mit je zwei Detektoren jeder Art ausgeführt. Die Detektoren $H1$ und $H2$ arbeiten nach dem Hough-Verfahren und unterschieden sich im betrachteten Radiusbereich. Damit soll untersucht werden, wie sich der betrachtete Radiusbereich auf das Verhalten des Detektors auswirkt. Die Detektoren $BK1$ und $BK2$ arbeiten nach dem kombinierten Verfahren aus Hintergrundsubstraktion und Konturerkennung. Sie unterscheiden sich sich nicht in den Parametern, sondern in ihrem Umgang mit zu großen Flächen. $BK1$ ignoriert diese, $BK2$ wendet das in Abbildung 5.3.1 aufgeführte Splitten auf die Flächen an. Durch den Vergleich soll die Auswirkung der Erosion auf das Detektionsverhalten geprüft werden. Es findet keine Erosion auf der gesamten Maske statt. Die Parameter der verwendeten Detektoren sind in Tabelle 6.1 und in Tabelle 6.2 aufgeführt. Der Test wird mit dem Rechner des mobilen Billardsystems ausgeführt. Dieser besitzt einen Prozessor mit 4 Kernen a 3,10 GHz und 8GB Arbeitsspeicher. Die einzelnen Tests und ihre Ergebnisse werden in den nächsten Abschnitten vorgestellt.

Parameter	**H1**	**H2**
r	15 Pixel	15 Pixel
$threshCircle$	12.0	12.0
$threshCanny$	79.0	79.0
$deltaRadius$	0	4

Tabelle 6.1: Eingestellte Parameter der getesteten Hough-Detektoren

6.3.1 Rechengeschwindigkeit

Für den Vergleich der benötigten Berechnungszeiten für die Kugelerkennung der einzelnen Detektoren wird die in Microsoft Visual Studio 2010 enthaltene Performancemessung benutzt. Der Test wird mit einem gespeicherten Video eines Billardspiels durchgeführt, um

Parameter	BK1	BK2
$FilterSize$	3 Pixel x 3 Pixel	3 Pixel x 3 Pixel
$minAreaBall$	159 Pixel^2	159 Pixel^2
d	30 Pixel	30 Pixel
δ	15	15
Splitten von Konturen	ja	nein

Tabelle 6.2: Eingestellte Parameter der kombinierten Detektoren

die Vergleichbarkeit der Ergebnisse sicherzustellen. Bei diesem wird für jeden Detektor die Kugeldetektion 500 mal aufgerufen.

Ergebnisse

Die Ergebnisse für die einzelnen Detektoren sind in Tabelle 6.3 und Tabelle 6.4 aufgeführt. Dort sind zusätzlich die durchschnittlichen Berechnungszeiten für die aufgerufenen Unterfunktionen ersichtlich. Die aufgeführten Zeiten sind die Mittelwerte aus den 500 Einzelmessungen.

Aufgerufene Funktion	Benötigte Zeit H1	benötigte Zeit H2
Kugeldetektion insgesamt	10,10 ms/Bild	10,26 ms/Bild
HoughCircles	9,33 ms/Bild	9,49 ms/Bild
cvtColor	0,69 ms/Bild	0,69 ms/Bild
andere	0,08 ms/Bild	0,08 ms/ Bild

Tabelle 6.3: Rechenzeit H1 und H2

Aufgerufene Funktion	Benötigte Zeit BK1	benötigte Zeit BK2
Kugeldetektion insgesamt	16,43 ms/Bild	8,96 ms/Bild
findContours	6,57 ms/Bild	1,20 ms/Bild
calcForegroundMask	6,53 ms/Bild	6,56 ms/Bild
calcContourArea	0,08 ms/Bild	0,06 ms/ Bild
erode	0,00 ms/Bild	0,04 ms/ Bild
andere	3,25 ms/Bild	1.02 ms/ Bild

Tabelle 6.4: Rechenzeit BK1 und BK2

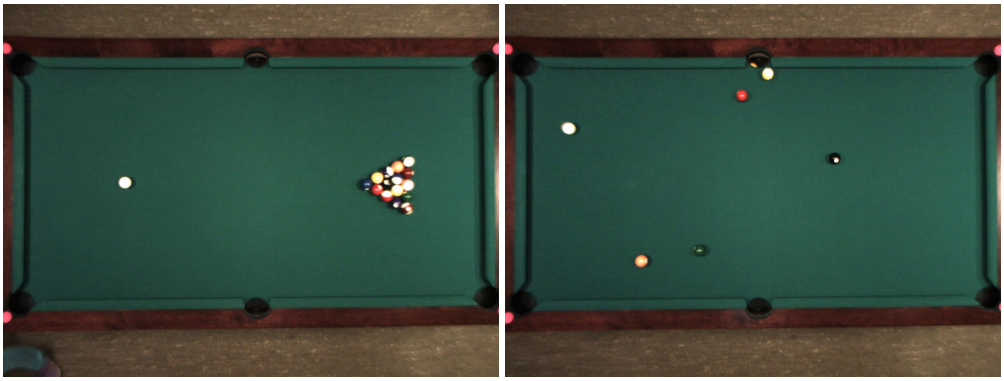
Bei den Detektoren H1 und H2 nimmt die Kreiserkennung mittels Hough-Transformation mit etwa 92 Prozent den größten Teil der Rechenzeit in Anspruch. Die Erhöhung der betrachteten

Radien von einem auf neun durch die Veränderung von $deltaRadius$ führt dabei nur zu einer Verlängerung der Rechenzeit um 1,6 Prozent.

Durch das rekursive Splitten der Flächen, welches in Abbildung 5.3.1 beschrieben ist, wird die Rechenzeit des Detektors BK1 in Vergleich zu BK2 fast verdoppelt. Dafür ist weniger die Erosionsfunktion verantwortlich, als das vermehrte Aufrufen der Konturfindung. Diese wird bei BK2 nur einmal pro Bild aufgerufen, bei BK1 jedoch im Durchschnitt 3,8 Mal. Die Aufteilung in Vordergrund und Hintergrund benötigt eine Zeit von ca. 6,5 ms pro Bild, dies sind etwa 68 Prozent der Zeit, welche die Kreiserkennung mit Hough-Transformation bei H1 und H2 benötigt. In Folge dessen ist es für eine Reduzierung der Rechenzeit nicht sinnvoll eine Kombination aus Hough-Transformation und Hintergrundsubstraktion für die Kugelerkennung zu implementieren.

6.3.2 Zuverlässigkeit

Für die Bewertung der Zuverlässigkeit werden zwei verschiedene aufgezeichnete Szenen mit je 20 Bildern untersucht. Während einer Szene verändern sich die Kugelpositionen auf dem Tisch nicht. Es treten jedoch sonstige Veränderungen, wie Personen oder Queues im Bild auf. Als erste Testszene wird eine Spielszene vor dem Anstoß ausgewählt. Mit dieser Szene soll die Anfälligkeit der Detektoren für zusammenhängende Kugelformationen gestestet werden. Als Vergleichsszene wird eine Spielszene aus der Mitte eines Spiels gewählt. Das erste Bild jeder Szene ist zur Verdeutlichung in Abbildung 6.5 dargestellt.



(a) Szene1 (b) Szene2

Abbildung 6.5: Ausgewählte Szenen

Für die Bewertung des Ergebnisses werden die in Tabelle 6.5 aufgeführten Fälle unterschieden.

Bezeichnung	**Kugel vorhanden**	**Kugel erkannt**
$TruePositive$ (TP)	ja	ja
$TrueNegative$ (TN)	nein	nein
$FalsePositive$ (FP)	nein	ja
$FalseNegative$ (FN)	ja	nein

Tabelle 6.5: Mögliche Testergebnisse

Um die Ergebnisse zu überpüfen, werden Referenzkoordinaten für die Kugelmittelpunkte der Bilder manuell bestimmt und ein rechteckiger Bereich um den Kugelmittelpunkt mit der Seitenlänge des Kugeldurchmessers markiert. Für das Testen der TrueNegative Werte werden zusätzlich zehn Markierungen, in kugelfreien Bereichen angebracht. Die Position der Markierungen wird mit den Positionen der erkannten Kugeln verglichen. Die angebrachten Markierungen des ersten Bildes sind in Abbildung 6.6 zur Verdeutlichung eingezeichnet.

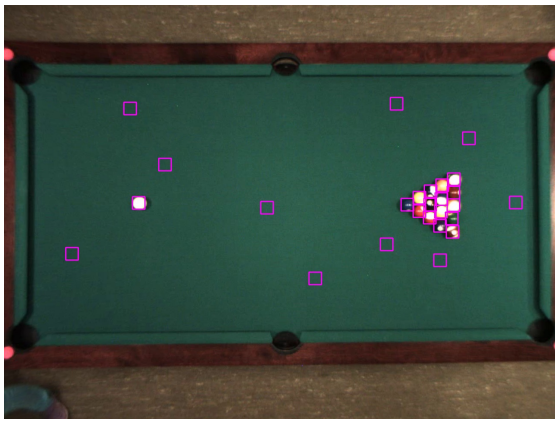


Abbildung 6.6: Gelabeltes Bild Szene1

Ein implementierter Algorithmus bestimmt die Testergebisse automatisch. Eine Kugel gilt als erkannt, wenn innerhalb des Kugellabels ein Kugelmittelpunkt erkannt wird.

Ergebnisse

Um die Ergebnisse der einzelnen Detektoren vergleichen zu können, wird die Vertrauenswahrscheinlichkeit ACC aus den Ergebnissen berechnet. Das Ergebnis dieser Berechnung ist in Tabelle 6.6 zu sehen. Die einzelnen Messergebisse sind in Abschnitt 2 aufgeführt. ACC ist definiert als das Verhältnis der richtig klassifizierten Objekte zu der Anzahl aller klassifizierten Objekte (siehe Gleichung 6.1).

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

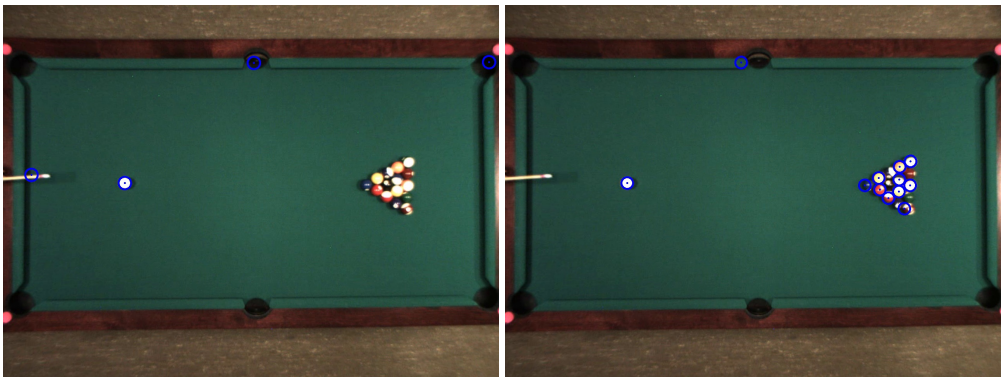
	H1	**H2**	**BK1**	**BK2**
Szene1	0,64	0,72	0,36	0,37
Szene2	0,88	0,92	0,75	0,75

Tabelle 6.6: Vertrauenswahrscheinlichkeit der Kugeldetektoren

Aus den Ergebnissen ist ersichtlich, dass alle Detektoren bei einer Häufung der Kugeln schlechtere Ergebnisse erzielen. Die Vertrauenswahrscheinlichkeit wird jedoch bei den kombinierten Detektoren um ca. 40 Prozent reduziert, während die Reduzierung bei den Hough-Detektoren nur ca. 20 Prozent beträgt. Verantwortlich für die insgesamt schlechteren Werte der kombinierten Detektoren ist, dass weniger Kugeln erkannt werden und vor allem in Randbereich oder den Taschen falsche Kugeln detektiert werden. Da schwarz kein eindeutiger Farbtonwert im HSV-Farbraum zugeordnet werden kann, werden die Taschen nicht durch die angewendete Hintergrundsubstraktion entfernt. In Abbildung 6.7 sind zwei verschiedene Detektionsergebnisse dargestellt, in welchen dieses Problem zu erkennen ist.

Aus den Ergebnissen ist ersichtlich, dass das Splitten zu keiner deutlichen Verbesserung des Detektionsergebisses führt. Dies ist damit zu begründen, dass für die Trennung von Kugeln durch Erosion ein Hohlraum zwischen den Flächen bestehen muss. Dieser Hohlraum ensteht jedoch nicht bei der Hintergrundsubstraktion, da die Spielfeldfläche nicht zwischen den Kugeln zu erkennen ist. Somit bietet die Erosion keinen Vorteil für die Erkennung der Einzelkugeln in dem getesteten System.

Um die Anzahl der falsch positiven Werte zu verringern, wird eine Filterung der erkannten Kugelpositionen durch die Klasse TableModel implementiert. Das Tischmodell prüft die ihm



(a) Detektionsergebis BK1 (b) Detektionsergebnis H2

Abbildung 6.7: Vergleich Detektionsergebnis Szene1

übergebenen Kugelpositionen und entfernt Kugeln, welche sich nicht auf der Spielfeldfläche befinden. Auf diese Weise werden die falsch positiven Werte stark reduziert und die Vertrauenswahrscheinlichkeit erhöht sich, wie in Tabelle 6.7 dargestellt. Falsche Detektionsergebnisse, die sich innerhalb des Spielfeldes befinden, wie z.B Körperteile des Spielers, können jedoch mit diesem Verfahren nicht reduziert werden.

	H1	**H2**	**BK1**	**BK2**
Szene1	0,64	0,76	0,42	0,42
Szene2	0,88	1,00	0,93	0,92

Tabelle 6.7: Vertrauenswahrscheinlichkeit der Kugeldetektoren mit Filterung

6.3.3 Genauigkeit

Um die Genauigkeit der Kugelerkennung zu bestimmen, wird die euklidische Distanz zwischen dem bestimmten Wert und der Position im Bezug auf die Kugelkoordinaten im entzerrten Eingangsbild bestimmt. In einem weiteren Schritt werden die erkannten Koordinaten im Bezug auf die realen Positionen der Kugeln auf dem Tisch bestimmt, um die entstehende Abweichung aus Bildentzerrung und Kugeldetektion zu bestimmen.

Abweichung Position im Bild zu erkannter Position

Für die Bestimmung der Abweichung durch den Kugeldetektor wird für jeden Detektor die Positionen der im vorherigen Test richtig erkannten Kugeln mit dem vorgegebenen Mittelpunkt des jeweiligen Referenzmittelpunktes verglichen. Es wird die mittlere Abweichung $\overline{a}$ jedes Detektors pro Szene bestimmt, wie auch die maximale Abweichung a_{max}.

Die Ergbenisse sind in Tabelle 6.8 zu finden. Es fällt auf, dass das Splitten keinen Vorteil für die Detektionsgenauigkeit mit sich bringt. Die Detektoren BK1 und BK2 sind durch die Bestimmung des Kugelmittelpunktes durch den Mittelpunkt des die Konturen umgebenden Rechteckes ungenauer bei der Positionsbestimmung als die Hough-Detektoren.

Die Genauigkeit der kombinierten Detektoren in Szene1 höher ist als in Szene2. Dies beruht jedoch darauf, dass in Szene1 in jedem Bild nur der Spielball erkannt wird. Für diesen ist es aufgrund des hohen Kontrastes einfacher den Mittelpunkt korrekt zu bestimmen. Die maximalen Abweichungen für die Hough-Detektoren sind in Szene1 auch größer als die der kombinierten Detektoren. Dies beruht auf der erschwerten Erkennung von Kugeln in einer Gruppe. In Abbildung 6.8 ist eine starke Abweichung bei der Erkennung der blauen Kugel zu erkennen.

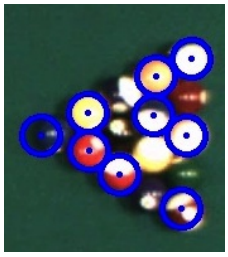


Abbildung 6.8: Große Abweichung bei Kugelerkennung mit H2

Für die zweite Szene sind alle Detektionsergebnisse deutlich besser. Dies ist auch die Szene mit der die Genauigkeit bewertet wird, da Szene1 nur einen Sonderfall an Anfang eines Billardspiels darstellt. Die Kugeln sind auf dem Bild mit einem Durchmesser von 30 Pixeln dargestellt. Bezogen auf diesen Durchmesser haben die Detektoren eine Abweichung von 4,7 Prozent bis 7,3 Prozent.

	H1	H2	BK1	BK2
Szene 1 a_{max}	12,0 Pixel	14,0 Pixel	3,2 Pixel	3,2 Pixel
Szene 1 $\overline{a}$	1,5 Pixel	2,2 Pixel	1,8 Pixel	1,8 Pixel
Szene 2 a_{max}	3,2 Pixel	7,1 Pixel	9,2 Pixel	9,2 Pixel
Szene 2 $\overline{a}$	1,4 Pixel	1,8 Pixel	2,2 Pixel	2,2 Pixel

Tabelle 6.8: Ergebnisse des Genauigkeitstest für den Kugeldetektor

Abweichung reale Position zu erkannter Position

Um die Abweichung der Kugelpositionen auf dem Tisch zu den erkannten Positionen zu bestimmen, werden die Kugeln an definierte Positionen auf dem Spielfeld platziert. Durch die Verwerdung eines Tischmodells mit der Beziehung von einem Millimeter pro Einheit können die erkannten Positionen mit den Kugelpositionen auf den Billardtisch direkt verglichen werden. Es wird ein Detektor nach dem Hough-Verfahren genutzt, dessen Werte automatisch von System während des Kalibrationsprozesses bestimmt werden. Der Test wird mit den mobilen Hardwaresystem ausgeführt, da das stationäre Hardwaresystem zum Testzeitpunkt nicht verfügbar ist. Es werden 40 Bilder der in Abbildung 6.9 abgebildeten Szene bewertet. Während der betrachteten Szene werden die Kugelpositionen nicht verändert, jedoch treten sonstige Veränderungen im Bild durch Queues oder Menschen auf.

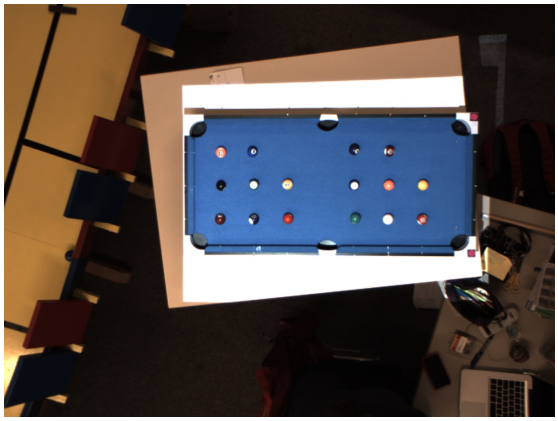


Abbildung 6.9: Testszene mobiles Hardwaresystem

Die ermittelten Ergebnisse werden in Tabelle 6.9 dargestellt.

Das mobile Hardwaresystem hat eine höhere Abweichung bei der Kugeldetektion, als das stationäre Hardwaresystem durch die automatische Bestimmung der Detektorparameter sowie

	Abweichung Kugeldetektor	Abweichung System
a_{max}	11,4 Pixel	8,1 mm
$\overline{a}$	5,5 Pixel	3,5 mm

Tabelle 6.9: Ergebnisse des Genauigkeitstest für das mobile Hardwaresytem

den kleineren Abbildungsmaßstab begründet. Eine Abweichung von 5,5 Pixeln bezogen auf einen Kugeldurchmesser von 24 Pixeln im entzerrten Eingangsbild bedeutet eine prozentuale Abweichung von 22,9 Prozent für die reine Kugeldetektion. Für das gesamte System aus Kugelerkennung und Bildentzerrung ergibt sich eine mittlere Abweichung von 3,5 mm . Bezogen auf einen Kugeldurchmesser von 32 mm ergibt sich eine prozentuale Abeichung von 10,9 Prozent.

6.3.4 Zusammenfassung

Für die Auswahl der geeigneten Implementierung der Kugeldetektion werden an dieser Stelle die Ergebnisse aus den durchgeführten Test zusammengefasst und anhand dieser Ergebnisse eine Entscheidung für die Implementierung getroffen.

Test	H1	H2	BK1	BK2
Rechenzeit	10,10 ms/Bild	10,26 ms/Bild	16,43 ms/Bild	8,96 ms/Bild
ACC Szene1	0,64	0,72	0,36	0,37
ACC Szene2	0,88	1,0	0,93	0,92
Abweichung	1,4 Pixel	1,8 Pixel	2,2 Pixel	2,2 Pixel

Tabelle 6.10: Ergebnisse des Genauigkeitstest für den Kugeldetektor

Aus Tabelle 6.10 ist ersichtlich, dass der Detektor BK1 aufgrund des Splittens von großen Flächen deutlich langsamer ist als Detektor BK1. Da das Splitten keine deutliche Verbesserung bei der Zuverlässigkeit oder der Genauigkeit mit sich bringt, wird die Implementierung nach der Vorlage von BK1 für das Projekt ausgeschlossen.

Auch wenn die Rechenzeit von BK2 um ca 7 Prozent schneller ist als die von H2, wird die Implementierung ausgeschlossen, da die Zuverlässigkeit bei Kugelhäufungen deutlich geringer ist als die von H1 und H2.

Da sich die Rechenzeit durch die Betrachtung von mehreren Radien nur wenig ändert, sich jedoch die Zuverlässigkeit erhöht, wird die Betrachtung von mehreren Radien erlaubt.

6.4 sonstige Überprüfungen

In diesem Abschnitt werden die Genauigkeit der Queueerkennung, die Zuverlässigkeit der Spielballklassifikation und die Rechenzeit des Systems für den Spielablauf überprüft.

Queuedetektion

Um die Queueerkennung zu überprüfen, wird der Winkel in 4 Szenen mit je 5 Bildern ermittelt und mit dem manuell ermittelten Winkel verglichen. In einer Szene verändert sich die Queueposition nicht. Es befinden sich in manchen Szenen andere Kugeln sowie Banden im Suchbereich des Queues, um die Fehleranfälligkeit im normalen Spielbetrieb zu testen. Es wird sowohl die mittlere, als auch die maximale Abweichung bestimmt.

Es ergibt sich für die Queueerkennung eine mittlere Abweichung von 1,7° für das mobile Hardwaresystem und 1,0° für das stationäre Hardwaresystem.

Klassifikation des Spielballs

Um die Zuverlässigkeit des Kugelklassifikators zu testen, wird ein Video eines Billardspiels zum Testen verwendet. Dabei werden die Parameter des Klassifikators mittels der automatischen Kalibrierungsfunktion bestimmt. Die Kugelklassifikation wird 1000 Mal aufgerufen. Die weiße Kugel befindet sich in jedem Bild auf dem Billardtisch.

In diesem Testfall wird mindestens eine weiße Kugel auf 966 von 1000 Bildern erkannt. Dabei wird in 965 Bildern der Spielball korrekt als Kugel von Typ *White* erkannt. In drei Bildern wird eine falsche Kugel oder mehr als eine Kugel als weiße Kugel erkannt. Es werden nur die Ergebnisse als korrekt angesehen, in welchen genau ein Spielball gefunden wird. Der Spielball wird somit zu 96,3 Prozent richtig klassifiziert.

Einhaltung der Rechenzeiten

Für die Bestimmung der Rechenzeiten wird die in Microsoft Visual Studio 2008 enthaltene Performancemessung genutzt. Dabei wird die Zeit ermittelt, die das System für die Funktion *MainGame* und ihre aufgerufenen Unterfunktionen benötigt. Die Funktion wird 1000 Mal aufgerufen. Dabei sind alle unnötigen Ausgabefunktionen deaktiviert. Das ermittelte Ergebnis ist der Mittelwert aus allen Messungen. Da nicht jede Funktion in jedem der 1000 Messungen aufgerufen wird, wird das Ergebnis pro Aufruf in Tabelle 6.11 dargestellt.

Aufgerufene Funktion	**Benötigte Zeit**
MainSystem::MainGame insgesamt	55,38 ms/Aufruf
CameraSystem::getUndistortedFrame	31,75 ms/Aufruf
ProjectorSystem::displayImage	9,45 ms/Aufruf
Warper::warpImage	6,04 ms/Aufruf
BallDetector::find	4,73 ms/Aufruf
QueueDetector::find	1,75 ms/Aufruf
TableModel::simulateShot	0,66 ms/Aufruf

Tabelle 6.11: Rechenzeiten der Funktion MainGame

Aus den Messergebnissen ist zu erkennen, dass das Einlesen und Entzerren des Eingangsbildes mehr als 50 Prozent der Rechenzeit in Anspruch nimmt. Die Funktion wird für Aufruf der Funktion MainGame einmal aufgerufen. Da die Kugelerkennung bei dem mobilen Hardwaresystem auf einem kleinen Bildausschnitt ausgeführt wird, ist die Rechenzeit deutlich geringer als in Unterabschnitt 6.3.1 ausgeführten Test.

6.5 Soll-Ist-Vergleich

An dieser Stelle sollen die implementierten Funktionen mit den in Kapitel 3 erstellten Anforderungen verglichen werden. Zunächst werden in Tabelle 6.12 die funktionalen Anforderungen verglichen. Es werden alle funktionalen Anforderungen an das System erfüllt.

Für die nichtfunktionalen Anforderungen sind zunächst die Ergebisse im Bezug auf ihre Korrektheit überprüft. In Tabelle Tabelle 6.13 sind die Ergebnisse zusammengefasst. Es ist ersichtlich, dass die maximal zugelassenen Abweichungen in allen Punkten unterschritten werden.

Nummer	Anforderung	Status
1.	Ermittlung der Kugelanzahl und Positionen	erfüllt
2.	Erkennung des Spielballs	erfüllt
3.	Erkennung des Queuewinkels	erfüllt
4.	Simulation des Stoßergebnisses	erfüllt
5.	Markierung der Kugeln	erfüllt
6.	Ausgabe des Simulationsergebnisses	erfüllt
7.	Anpassungsfähigkeit an die Hardware	erfüllt
8.	Tischerkennung für Inbetriebnahme	erfüllt

Tabelle 6.12: Status der funktionalen Anforderungen

Parameter	Soll	Ist
Abweichung Tischeerkennung	Siehe Abbildung 3.2b	durschnittlich 3 Pixel, keine Formabweichung
Abweichung Kugelmittelpunkt	15 Prozent im Bezug auf Kugeldurchmesser	10,9 Prozent im Bezug auf Kugeldurchmesser
Sicherheit der Kugelerkennung	$ACC > 0{,}7$	$ACC = 0,88 frH2$
Spielballklassifikation	80 Prozent der Bilder	96,3 Prozent der Bilder
Abweichung Queuewinkel	max 5° im Durchschnitt	1,45° im Durchschnitt

Tabelle 6.13: Soll-Ist-Vergleich der Abweichungen

Die erzielte durschnittliche Rechenzeit für den Hauptablauf beträgt 55,38 ms. Damit ist die Zeit unter den maximal zugelassenen 83 ms, jedoch über den 40 ms, die für die Berechnung jedes Bildes bei einer Bildwiederholrate von 24,5 fps unterschritten werden müssen. Somit kann nicht jedes Bild ausgewertet werden.

Nummer	Beschreibung der Anforderung	Status
9.	Einhalten der angegebenen Abweichungen bei der Objekterkennung	erfüllt
10.	Einhalten der erlaubten Rechenzeit	erfüllt
11.	Modulare Struktur der Software	erfüllt

Tabelle 6.14: Status der nichtfunktionalen Anforderungen

7 Zusammenfassung

Ziel dieser Arbeit war es eine Bildverarbeitungssoftware für die Anwendung eines computergestützen Billardtrainers zu entwickelt. Ein Billardtrainer soll die Spielsituation eines Billardspiels erfassen können und dem Spieler bei seiner Stoßwahl unterstützen.

7.1 Erreichte Ergebnisse

In dieser Arbeit ist eine modulare Softwarestruktur entwickelt worden, welche einen hohen Freiheitsgrad bei der Hardwareauswahl und den Umgebungsbedingungen zulässt. Durch die individuelle Einstellung der Parameter können die verschiedensten Kameras, Projektoren, Billardtische und Rechensysteme miteinander verbunden werden. Dadurch ist eine Nutzung mit beiden an der HAW Hamburg vorhandenen Hardwaresystemen wie auch die Verwendung von Videos möglich.

Es findet eine Kugel- und eine Queueerkennung statt, um den Spielzustand zu bestimmen. Das eingeführte Tischmodell bildet den Tischzustand ab und übernimmt die Simulation des Stoßergebnisses. Die berechneten Ergebnisse werden an den Nutzer übertragen, indem die erkannten Objekte und die berechneten Stoßergebnisse direkt auf dem Billardtisch mittels eines Projektors markiert werden.

Für die Inbetriebnahme wird ein geführter Prozess implementiert, der alle benötigten Parameter abfragt oder selbst bestimmt. Das System kann die Position des Billardtisches im Bild automatisch bestimmen und auch die Parameter für die Kugelerkennung und -klassifikation an die Umgebungsverhältnisse anpassen.

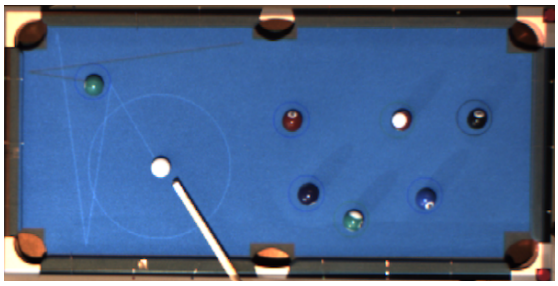


Abbildung 7.1: Ausgabe der Ergebnisse des Billardtrainers

7.2 Ausblick

Die entwickelte Software bietet eine gute Basis für zukünftige Weiterentwicklungen. Durch die modulare Struktur kann gezielt an einem Bereich gearbeitet werden, was auch die Einarbeitungszeit verringert, da nur die Funktionsweise von einem Teilgebiet und dessen Schnittstellen zum Gesamtsystem bekannt sein muss. Dadurch ist auch eine parallele Bearbeitung der Software durch mehrere Projektteilnehmer in Zukunft leichter zu realisieren, welche auch durch die eingerichtete Versionskontrolle unterstützt wird.

Im Zukunft ist die Laufzeit des Programms so zu reduzieren, dass die Berechnungszeit unter 40 ms liegt. Dies kann durch das Parallelisieren der Programmbearbeitung in die Blöcke Bildaufnahme, Berechnung und Ergebnisausgabe erreicht werden.

Zur Zeit verändern sich die berechneten Kugellaufbahnen zum Teil sehr sprunghaft, bedingt durch die Abweichungen bei der Queue- und Kugelerkennung. Um diese Schwankungen zu glätten, ist es sinnvoll die Kugelpositionen nur komplett neu zu bestimmen, wenn ein Schuss ausgeführt wurde. Da sich die Kugelpositionen nur dann verändern können. Wird kein Schuss ausgeführt kann zum einen der Suchradius für die Kugeln auf ein Gebiet um den zuletzt bekannten Ort beschränkt werden, zum anderen können die erkannten Positionen gefiltert werden und so die Kugelerkennung verbessert werden.

Das im Tischmodell hinterlegte Physikmodell ist noch sehr einfach gehalten. Dadurch kommt es zum Teil zu starken Abweichungen zwischen dem simulierten Ergebnis und dem realen Stoßergebnis. Im Zukunft kann das Tischmodell so verändert werden, dass es unterschiedliche Kugelgeschwindigkeiten, Reibungsverluste und ein dezentrales Anstoßen des Spielballs unterstützt.

Es gibt also noch eine Vielzahl von Funktionen um die die aktuelle Version des BillardTrainers erweitert werden kann.

Literaturverzeichnis

[Basler] : *Basler-acA2500- 14gc*. – URL http://www.baslerweb.com/products/ace.html?model=171. – Zugriffsdatum: 2013-07-22

[van Balen 2009] Balen, Joris van: *A Virtual Billiard Assistant*, University of Twente, Diplomarbeit, 2009

[Bradski und Kaehler 2008] Bradski, Gary ; Kaehler, Adrian: *Learning OpenCV: Computer vision with the OpenCV library*. O'reilly, 2008

[Cash 2003] Cash, D G.: *Video Analysis, Spatial Reasoning and Visualisation of Pool and Billiards*, Leeds University, UK, Dissertation, 2003

[Chang 1994] Chang, SWS: *Automating skills using a robot snooker player*, Ph. D. Dissertation, Bristol University, Dissertation, 1994

[Chou u. a. 2009] Chou, Chen-Wei ; Tien, Ming-Chun ; Wu, Ja-Ling: Billiards wizard: A tutoring system for broadcasting nine-ball billiards videos. In: *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, 2009, S. 1921–1924. – ISSN 1520-6149

[D'Orazio u. a. 2002] D'Orazio, T. ; Ancona, N. ; Cicirelli, C. ; Nitti, M.: A ball detection algorithm for real soccer image sequences. In: *Pattern Recognition, 2002. Proceedings. 16th International Conference on* Bd. 1, 2002, S. 210–213 vol.1. – ISSN 1051-4651

[e.V 2007] e.V, DBU: *Materialnormen Pool*. Deutsche Billard-Union e.V. (Veranst.), 12 2007

[Gonzalez u. a. 2009] Gonzalez, Rafael C. ; Woods, Richard E. ; Eddins, Steven L.: *Digital image processing using MATLAB*. Bd. 2. Gatesmark Publishing Knoxville, 2009

[Greenspan u. a. 2008] Greenspan, M. ; Lam, J. ; Godard, M. ; Zaidi, I. ; Jordan, S. ; Leckie, W. ; Anderson, K. ; Dupuis, D.: Toward a Competitive Pool-Playing Robot. In: *Computer* 41 (2008), Nr. 1, S. 46–53

[Greenspan u. a. 2007] Greenspan, M. ; Lam, J. ; Leckie, W. ; Godard, M. ; Zaidi, I. ; Anderson, K. ; Dupuis, D. ; Jordan, S.: Toward a Competitive Pool Playing Robot: Is Computational Intelligence Needed to Play Robotic Pool? In: *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, 2007, S. 380–388

[Helm u. a. 2011] Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Pearson Deutschland GmbH, 2011

[Jähne 2012] Jähne, Bernd: *Digitale Bildverarbeitung, 7. Auflage*. Springer-Verlag GmbH, 2012

[Kato und Billinghurst 1999] Kato, H. ; Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*, 1999, S. 85–94

[Kleuker 2011] Kleuker, Stephan: *Grundkurs Software-Engineering mit UML : der pragmatische Weg zu erfolgreichen Softwareprojekten*. 2., korrigierte und erw. Aufl. Wiesbaden : Vieweg+Teubner Verlag / Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2011

[Laganière 2011] Laganière, Robert: *OpenCV 2 computer vision application programming cookbook*. Packt Publishing, 2011

[Larsen u. a. 2005] Larsen, Lars B. ; Jensen, Rene B. ; Jensen, Kasper L. ; Larsen, Søren: Development of an automatic pool trainer. In: *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. New York, NY, USA : ACM, 2005 (ACE '05), S. 83–87. – ISBN 1-59593-110-4

[Larsen u. a. 2002] Larsen, L.B. ; Jensen, M.D. ; Vodzi, W.K.: Multi modal user interaction in an automatic pool trainer. In: *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, 2002, S. 361–366

[Matas u. a. 2000] Matas, Jiri ; Galambos, Charles ; Kittler, Josef: Robust detection of lines using the progressive probabilistic hough transform. In: *Computer Vision and Image Understanding* 78 (2000), Nr. 1, S. 119–137

[MathWorks] MathWorks: *MATLAB*. – URL http://www.mathworks.de/products/matlab/. – Zugriffsdatum: 2013-09-06

[Matuska u. a. 2012] Matuska, S ; Hudec, R ; Benco, M: The comparison of CPU time consumption for image processing algorithm in Matlab and OpenCV. In: *ELEKTRO, 2012* IEEE (Veranst.), 2012, S. 75–78

[Müller 1998] Müller, Arnd: *Ein Billardroboter - praktische Realisierung von ausgewählten Konzepten der Robotik*, Universität Leipzig, Diplomarbeit, 1998

[Shih 2010] Shih, Chihhsiong: Aiming strategy error analysis and verification of a billiard training system. In: *Knowledge-Based Systems* 23 (2010), Nr. 7, S. 732 – 742. – URL http://www.sciencedirect.com/science/article/pii/S0950705109001609. – ISSN 0950-7051

[Tian u. a. 2004] Tian, Qi-Chuan ; Pan, Quan ; Cheng, Yong-Mei ; Gao, Quan-Xue: Fast algorithm and application of Hough transform in iris segmentation. In: *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on* Bd. 7, 2004, S. 3977–3980 vol.7

[Tönnies 2005] Tönnies, Klaus D.: *Grundlagen der Bildverarbeitung*. Bd. 1. Pearson Studium, 2005

[Uchiyama und Saito 2007] Uchiyama, Hideaki ; Saito, H.: AR Display of Visual Aids for Supporting Pool Games by Online Markerless Tracking. In: *Artificial Reality and Telexistence, 17th International Conference on*, 2007, S. 172–179

[Willowgarage] Willowgarage: *OpenCVWiki*. – URL http://opencv.willowgarage.com/wiki/. – Zugriffsdatum: 2013-08-20

[Yuen u. a. 1990] Yuen, HK ; Princen, John ; Illingworth, John ; Kittler, Josef: Comparative study of Hough transform methods for circle finding. In: *Image and Vision Computing* 8 (1990), Nr. 1, S. 71–77

Abbildungsverzeichnis

Anhang

1 Inhalt der CD

Dateiname	Beschreibung
thesishilbert.pdf	Diese Arbeit als .pdf Datei
Billardtrainer	Projektordner des erstellten Programms

2 Ergebnisse Zuverlässigkeitstest

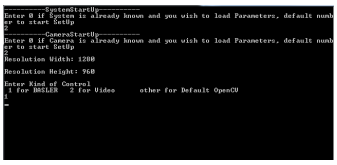
		H1	**H2**	**BK1**	**BK2**
Szene1	True Positive	133	203	20	20
	False Positive	0	36	83	80
	False Negative	187	117	300	300
	True Negative	200	200	200	200
Szene2	True Positive	82	120	95	95
	False Positive	0	29	74	72
	False Negative	38	0	25	25
	True Negative	200	200	200	200

Tabelle 1: Testergebnisse ohne Filterung

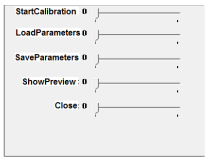
		H1	**H2**	**BK1**	**BK2**
Szene1	True Positive	133	203	20	20
	False Positive	0	12	1	1
	False Negative	187	117	300	300
	True Negative	200	200	200	200
Szene2	True Positive	82	120	95	95
	False Positive	0	0	0	0
	False Negative	38	0	25	24
	True Negative	200	200	200	200

Tabelle 2: Testergebnisse nach Filterung

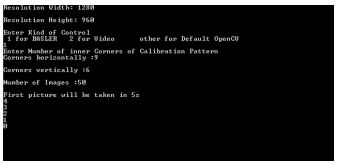
3 Graphische Interaktion bei der Inbetriebnahme



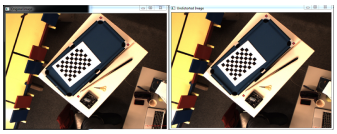
(a) Abfrage Kameraparameter



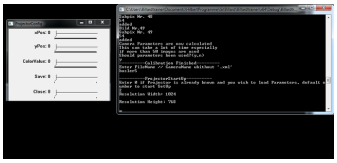
(b) GUI Kamerakonfiguration



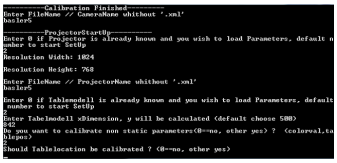
(c) Start der Kamerakalibierung



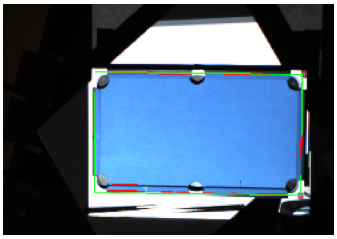
(d) Kalibierungsergebnis



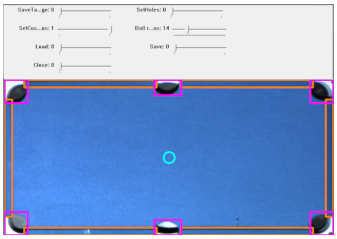
(e) Abfrage Projektorparameter



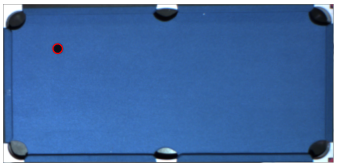
(f) Abfrage Tischmodellparameter



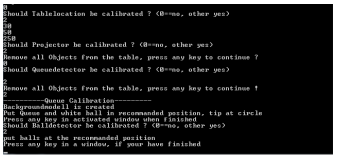
(g) Ergebnis Tischdetektion



(h) Einstellung Tischparameter



(i) Projektorkalibierung



(j) Queuekalibierung

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 20.09.2013 Ines Hilbert