HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Bachelorarbeit, Abteilung Informatik

Bildanalyse zur Erkennung von Bewegungen eines Fussballs - Machbarkeitsstudie und Demonstrator

Hochschule für Technik Rapperswil

Herbstsemester 2012
21. Dezember 2012

Autor:	Daniel Stucki
Betreuer:	Prof. Dr. Peter Heinzmann
Experte:	Dr. Hans Grossmann, compar vision sytems & robotics
Gegenleser:	Dr. Ruedi Stoop
Arbeitsperiode:	17.09.2012 - 21.12.2012
Arbeitsumfang:	360 Stunden, 12 ECTS

Abstract

Die HSR/cnlab-Fussball-Tracer Anwendung zeichnet mit Hilfe von GPS-Trackern die Laufwege von Spielern auf. Die Position des Balles wird jedoch nicht erfasst. Im Rahmen dieser Bachelorarbeit werden Möglichkeiten zur optischen Erfassung der Ballpositionen untersucht.
Dazu wurden zuerst Anforderungen an fix positionierte Digitalkameras zur Aufnahme der Spielsituation bestimmt. Anhand geometrischer Überlegungen konnte abgeschätzt werden, welche räumliche Auflösung benötigt wird, um den Ball zu erkennen. Untersucht wurde auch, wie man mit Hilfe von Wärmebild-Kameras Spieler und Ball unterscheiden könnte. Im Rahmen der Arbeit war ein ausgedehntes Studium der Grundlagen zur Bildverarbeitung nötig. Zum besseren Verständnis dieser Algorithmen wurden diverse Demo-Anwendungen realisiert. Die Demo-Anwendungen und der Prototyp zur Erkennung der Ballpositionen wurden in C++ unter Verwendung der OpenCV-Bibliothek realisiert. Die Spielfeldaufnahmen erfolgten mit einer GoPro HD Hero2 Digitalkamera.
Anhand von Bildaufnahmen mit einer fix positionierten Digitalkamera kann die Position des Balles in einem begrenzten Raum von 50 x 30 m bestimmt werden. Die aufgenommene Videosequenz wird mit einem Vordergrundextraktionsalgorithmus von OpenCV verarbeitet. Dieser liefert alle bewegten Objekte auf dem Spielfeld. Mittels Ausschlussverfahren und Berücksichtigung der Grösse und Bewegung des Balles können Spieler vom Ball unterschieden werden. Die resultierenden Ballpositionen werden mittels OpenCV auf Spielfeldpositionen umgerechnet. Die Umrechnung basiert auf zuvor markierten Punkten im Videobild. Der realisierte Prototyp zeigt die Wege des Balles bei übersichtlichen Situationen und für begrenzte Spielfeldgrössen (etwa ein Viertel eines Spielfeldes). Bei den ersten Tests mit Aufnahmen in einer Halle kann der Ball bei rund 67% der Bilder erkannt werden. In einer nächsten Phase soll das System im Frühjahr 2013 auf grossen Spielfeldern getestet werden. Um den Ball auch auf den grossen Spielfeldern erfassen zu können, sind vier parallel betriebene Systeme nötig. Die aktuelle Bestimmung der Position im Feld vernachlässigt die Flughöhe des Balles. Durch den Einsatz weiterer Kameras sollte auch die 3- dimensionale Positionsbestimmung möglich sein.

Aufgabenstellung

Studiengang:	Informatik (I)
Semester:	HS 2012 (17.09.2012-17.02.2013)
Institut:	ITA: Internet-Technologien und Anwendungen
Autor:	Daniel Stucki
Verantwortlicher:	Prof. Dr. Peter Heinzmann
Experte:	Dr. Hans Grossmann, compar vision sytems & robotics
Gegenleser:	Dr. Ruedi Stoop

Mittels GPS-Aufzeichnung werden Bewegungen von Fussballspielern analysiert. Im Rahmen einer Machbarkeitsstudie soll nun abgeklärt werden, wie die Bewegung des Balls anhand von Videobildanalysen erfasst werden kann. Mittelfristig will man die Spielerpositionsdaten bei `www.cnlab.ch/fussball/`mit den Ballpositionsdaten ergänzen. Im Rahmen der theoretischen Untersuchungen sind grundsätzliche Fragen zur Bildverarbeitung zu klären und anhand einfacher Anwendungen zu illustrieren:

- Kamerabasics (Optik, Auflösung, Formate, Datenraten)
- Bildverarbeitungsalgorithmen
- Geometrische Umrechnungen (Auflösung, Positionsbestimmung relativ zum Spielfeld)

Die Machbarkeitsstudie soll zeigen, wie die Erkennung und Positionsbestimmung des Balls anhand von Videoaufnahmen am besten zu realisieren ist:

- Örtliche und zeitliche Auflösung der Ballpositionen (z.B. +/- 1m, eine Position alle 0.2s)
- Anzahl und Positionierung der benötigten Kameras
- Räumliche und zeitliche Auflösung der Videoaufnahmen, Videoformate
- Einfluss Objektive, Tiefenschärfe, Beleuchtung
- Möglichkeiten zur einfacheren Ballerkennung (z.B. spezielle Ballfarben)
- Algorithmen zur Berechnung der Position im Feld (evtl. basierend auf speziell konfigurierten Referenzpunkten)

Es ist ein Prototyp aufzubauen, mit welchem die Ballpositionen bei einfachen Spielsituationen (nur ein Ball und maximal zwei Spieler) im Videobild hervorgehoben und auf einem Spielfeldbild eingezeichnet werden. Der Prototyp soll abschliessend auch bei echten Spielsituationen eingesetzt werden, wobei nur ein Teil des Spielfelds abzubilden ist. Hier geht es darum, die Leistungsgrenzen des Systems aufzuzeigen. Zu verwendende Hardware und Software:

- OpenCV Bilderkennungs-SW: `http://docs.opencv.org/`
- GoPro und Mobotix Kameras: `http://de.gopro.com`, `http://www.mobotix.com`

Diese Aufgabenstellung wird genehmigt vom Betreuer.

Rapperswil 19.12.12 — Prof. Dr. Peter Heinzmann

Ort / Datum

Erklärung zur Urheberschaft[1]

Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellungen und Beiträgen gemäss folgender Aufstellung:

Gegenstand, Leistung	Person	Funktion
Alle Kapitel	Daniel Stucki	Autor der Arbeit
Korrektur	Heinrich Stucki	Lektorat
Korrektur	Christian Stucki	Lektorat
Idee, Aufgabenstellung, allgemeines Pflichtenheft, Betreuung während der Arbeit	Prof. Dr. Peter Heinzmann	Verantwortlicher Professor
Mitarbeit für Bilder und Videos	Lukas Wunderle	Model
Mitarbeit für Videos	Turnverein Oberurnen	Fussball spielen

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt habe,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Rapperswil, 21. Dezember 2012

Daniel Stucki

[1] Diese Erklärung basiert auf der Muster-Erklärung in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 16. Februar 2009.

Vereinbarung zur Verwendung und Weiterentwicklung der Arbeit[2]

1. Gegenstand der Vereinbarung
Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit „Bildanalyse zur Erkennung von Bewegungen eines Fussballs - Machbarkeitsstudie und Demonstrator“ von Daniel Stucki unter der Betreuung von Prof. Dr. Peter Heinzmann (für die Arbeit verantwortlicher Professor) geregelt.

2. Urheberrecht
Die Urheberrechte stehen dem Student zu.

3. Verwendung
Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, d.h. vom Studenten, welcher die Arbeit verfasst hat, sowie vom verantwortlichen Professor verwendet und weiter entwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht, aber nicht Pflicht.

Rapperswil, den

Daniel Stucki

Rapperswil, den

Prof. Dr. Peter Heinzmann

[2]Diese Vereinbarung basiert auf den Muster-Vereinbarungen in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 16. Februar 2009.

Management Summary

Ausgangslage

Die HSR/cnlab-Fussball-Tracer Anwendung zeichnet mit Hilfe von GPS-Trackern die Laufwege von Spielern auf. Die Position des Balles wird jedoch nicht erfasst. Im Rahmen dieser Bachelorarbeit werden Möglichkeiten zur optischen Erfassung der Ballpositionen untersucht.

Vorgehen

Zuerst wurden Anforderungen an fix positionierte Digitalkameras zur Aufnahme der Spielsituation bestimmt. Anhand geometrischer Überlegungen konnte abgeschätzt werden, welche räumliche Auflösung benötigt wird, um den Ball zu erkennen. Schnell wurde klar, dass mit einer Kamera nicht ein ganzes Fussballfeld abgedeckt werden kann.
Der erste Ansatz in der Detektion des Balles war die Suche nach Kreisen auf dem Kamerabild, dazu bietet OpenCV eine Implementation der Hough Transformation an. Dieser Ansatz hat sich aber als nicht verlässlich gezeigt. Mit der Extraktion von bewegten Objekten aus dem Videobild und einer nachfolgenden Unterscheidung in Ball / Mensch konnten bessere Resultate erreicht werden.
Für die Unterscheidung von Mensch und Ball wurde eine Wärmebildkamera der Marke Flir ausprobiert. Die Unterscheidung mit Hilfe der Wärmebildkamera verlief einwandfrei. Zusätzlich war im Rahmen der Arbeit ein ausgedehntes Studium der Grundlagen zur Bildverarbeitung nötig. Zum besseren Verständnis dieser Algorithmen wurden diverse Demo-Anwendungen realisiert. Die Demo- Anwendungen und der Prototyp zur Erkennung der Ballpositionen wurden in C++ unter Verwendung der OpenCV-Bibliothek realisiert. Die Spielfeldaufnahmen erfolgten mit einer GoPro HD Hero2 Digitalkamera.

Ergebnis

Anhand von Bildaufnahmen mit einer fix positionierten Digitalkamera kann die Position des Balles in einem begrenzten Raum von 50 x 30 m bestimmt werden. Die aufgenommene Videosequenz wird mit einem Vordergrundextraktionsalgorithmus

von OpenCV verarbeitet. Dieser liefert alle bewegten Objekte auf dem Spielfeld. Mittels Ausschlussverfahren und Berücksichtigung der Grösse und Bewegung des Balles, können Spieler vom Ball unterschieden werden. Die resultierenden Ballpositionen werden mittels OpenCV auf Spielfeldpositionen umgerechnet. Die Umrechnung basiert auf zuvor markierten Punkten im Videobild. Der realisierte Prototyp zeigt die Wege des Balles bei übersichtlichen Situationen und für begrenzte Spielfeldgrössen (etwa ein Viertel eines Spielfeldes). Bei den ersten Tests mit Aufnahmen in einer Halle kann der Ball bei rund 67% der Bilder erkannt werden.

Ausblick

In einer nächsten Phase soll das System im Frühjahr 2013 auf grossen Spielfeldern getestet werden. Um den Ball auch auf den grossen Spielfeldern erfassen zu können, sind vier parallel betriebene Systeme nötig. Die aktuelle Bestimmung der Position im Feld vernachlässigt die Flughöhe des Balles. Durch den Einsatz weiterer Kameras sollte auch die 3-dimensionale Positionsbestimmung möglich sein.

Inhaltsverzeichnis

[3] Dieser Abschnitt beruht auf Angaben und Regeln in [Fus]
[4] [Re11, p272-277]
[5] [BK08, p115-117]

Kapitel 1

Einleitung

In diesem Kapitel wird der grössere Zusammenhang der Arbeit erläutert. Dafür wird in einem ersten Schritt kurz auf die HSR/cnlab-Fussball-Tracer Applikation[1] eingegangen.

1.1 HSR/cnlab-Fussball-Tracer Anwendung

Die HSR/cnlab-Fussball-Tracer Anwendung stellt ein System dar, welches Positionen von Fussballspielern aufzeichnet. Dies geschieht mit Hilfe von GPS-Trackern, welche die Spieler an sich tragen. Die Positionen liegen in einer zeitlichen Auflösung von 5Hz und einer Genauigkeit von ± 1m vor. Nach dem Spiel werden die von den Trackern gesammelten Positionen ausgelesen und über eine Webseite verfügbar gemacht. Die Webseite bietet verschiedene Auswertungen an, welche für einen Fussballtrainer von Nutzen sind. In Abbildung 1.1 ist ein Teil der Funktion der Webseite aufgezeigt, nämlich das Visualisieren der Spielerpositionen auf eine schematische Darstellung eines Fussballfeldes. Dabei werden pro Spieler jeweils die letzten fünf Positionen dargestellt und mittels Linien verbunden.

Zurzeit wird die Position des Balles in der Applikation nicht aufgezeichnet

1.2 Fragestellung und Vorgehen

Diese Arbeit klärt im Sinne einer Machbarkeitsstudie ab, ob es möglich ist, einen Fussball mit Hilfe von optischen Kameras zu detektieren und die Position so umzurechnen, dass die Fussball-Tracer Anwendung den Ball darstellen kann. Dafür wird ein Prototyp entwickelt, welcher aus einer Videodatei Bild für Bild den Ball detektiert und auf ein schematisches Fussballfeld überträgt. Der Prototyp ist in C++ geschrieben und benutzt die frei zugängliche OpenCV Programmbibliothek.

[1] `http://cnlab.ch/fussball`

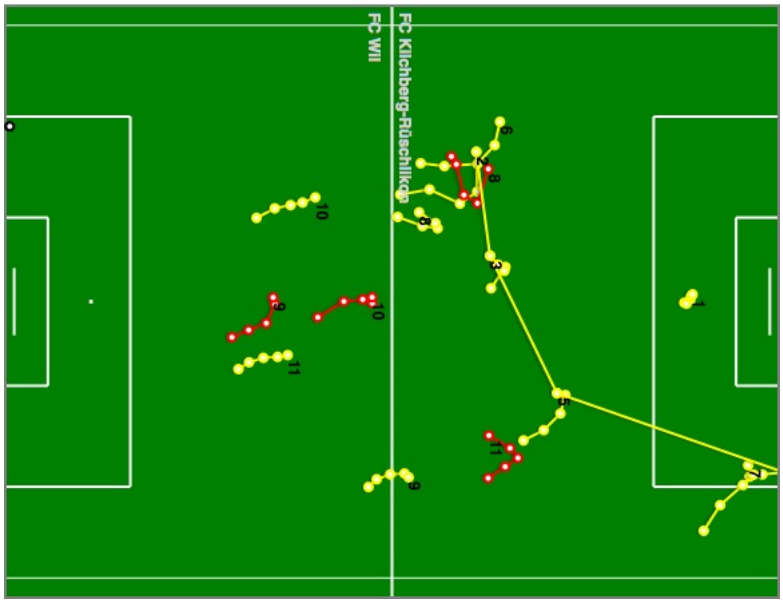


Abbildung 1.1: Screenshot aus cnlab/HSR Fussballtracking System

Zur Bearbeitung der Fragestellung wird folgendermassen vorgegangen. Zuerst werden die Anforderungen an fix positionierte Digitalkameras zur Aufnahme der Spielsituation festgelegt. Anhand geometrischer Überlegungen wird abgeschätzt, welche räumliche Auflösung benötigt wird, um den Ball zu erkennen. Für die Detektion des Balles werden unterschiedliche Ansätze verfolgt und deren Verlässlichkeit diskutiert. Ein spezielles Augenmerk gilt dabei auch der Unterscheidung von Ball und Mensch. Der Wahl des Aufnahmemediums (Art der Digitalkamera oder gar Wärmebildkamera) und des Bildverarbeitungsalgorithmus kommt in der ganzen Machbarkeitsstudie eine zentrale Rolle zu.

Die vorliegende Arbeit ist wie folgt aufgebaut. Im zweiten Kapitel werden die theoretischen Grundlagen betreffend Kamerabasics beschrieben, welche für die praktische Anwendung der Fragestellung relevant sind. Im dritten Kapitel wird die Thematik der Bildverarbeitungsalgorithmen vertieft behandelt. Basierend auf diesen erarbeiteten Grundlagen und Erkenntnissen wird im vierten Kapitel die Realisierung des Prototyps sowie der Demoapplikationen erläutert. Im Schlusskapitel werden die wichtigsten Ergebnisse zusammengefasst, das Endprodukt vorgestellt und ein Ausblick in die Zukunft gegeben.

Kapitel 2

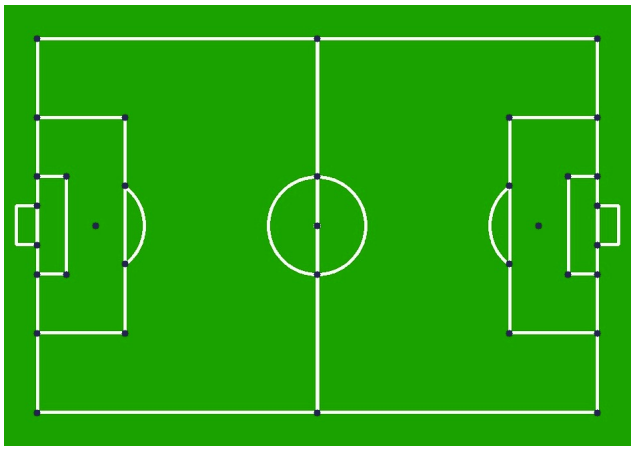
Theoretische Grundlagen

Dieses Kapitel behandelt theoretische Grundlagen, welche im Zusammenhang dieser Arbeit erarbeitet worden sind und für Folgerungen und Entscheide genutzt werden. Die Folgerungen aus diesem Kapitel werden im letzten Abschnitt angesprochen.

2.1 Fussballfeld und Fussball [1]

Die Ausmasse eines Spielfeldes sind 64m x 100m, bei Vorliegen einer Zustimmung des Regionalverbandes können diese Ausmasse jedoch variieren. Als Beispiel für einen abweichenden Platz sei auf '*Platz 1*' '*Allmeind Niederurnen*'[4] verwiesen.

Abbildung 2.1: Schematisches Fussballfeld mit 35 Schnittpunkten[3]



[1] Dieser Abschnitt beruht auf Angaben und Regeln in [Fus]

[3] Bild generiert mittels Prototyp Applikation

[4] `http://www.al-la.ch/de/Amateur-Liga/Verband-Amateur-Liga/Vereine-Amateur-Liga/Verein-AL.aspx/v-653457/sa-568264/a-sa/`, zuletzt besucht 17.12.12

Neben den variabel auftretenden Ausmassen des Fussballfeldes bestehen fixe Grössen für alle eingezeichneten Linien und Kreise. Aus den Schnittpunkten von Linien und der in den Reglementen vorgeschriebenen Abstände lassen sich für ein Fussballfeld 35 Schnittpunkte definieren.

Diese Schnittpunkte entstehen durch das Kreuzen beziehungsweise (nachfolgend bzw.) Aufeinandertreffen von Spielfeldlinien oder das Aufeinandertreffen der Torpfosten auf die Grundlinie. In der Abbildung 2.1 ist ein schematisches Fussballfeld dargestellt, bei welchem jeder der 35 Schnittpunkte als blauer Punkt dargestellt wird.

Der Fussball muss diversen Anforderungen genügen. So muss er kreisförmig sein und einen Umfang von 68 - 70cm aufweisen. Dies entspricht einem möglichen Durchmesser von 21.65 bis 22.28cm. Über die Farbe bzw. Muster macht das Regelwerk keine Angaben, was dazu führt, dass die Bälle je nach Hersteller und Modell jeweils anders aussehen. So wurde beim Bundesligaspiel zwischen Freiburg und Dortmund am 27.10.12 beispielsweise ein Fussball in pinker Farbe verwendet, um ihn im Schneegestöber sichtbar zu machen[5]. Der Unterschied der zwei Bälle, welche als offizielle Spielbälle der Bundesliga für die Saison 2012/13 fungieren, ist in Abbildung 2.2 ersichtlich.

Abbildung 2.2: Offizieller Spielball der deutschen Bundesliga. normale Ausführung links, Ausführung bei Schneefall rechts[7]

2.2 Digitale Bilder

Um Informationen aus einem digitalen Bild (nachfolgend Bild genannt) zu gewinnen, sind einige Grundkenntnisse nötig, welche in diesem Abschnitt vermittelt

[5] `http://www.welt.de/sport/fussball/article110317627/In-der-Bundesliga-spielen-sie-nun-mit-rosa-Ball.html`, zuletzt besucht 04.11.12

[7] Beide Bilder von `http://www.fussballhandel.de/`, zuletzt besucht 17.12.12

werden. Der Umfang dieser Arbeit reicht jedoch nicht aus, um alle Facetten der digitalen Bildverarbeitung abzudecken, sondern beschränkt sich auf die Kenntnisse, welche im Verlauf dieser Arbeit zum Vorschein kamen.

2.2.1 Vektor- und Rastergrafiken

Im Zusammenhang mit digitalen Bildern wird zwischen zwei Arten von Bildern unterschieden. Zum einen die Vektorgrafik und zum andern die Rastergrafik.

In einer Vektorgrafik werden Objekte dargestellt, welche durch Parameter darstellbar sind (zum Beispiel (nachfolgend z.B.) Rechtecke, Kreise, Polygone ...). Der Vorteil dieser Art von Grafik ist die unendliche Möglichkeit zur verlustfreien Skalierung. Der Nachteil hingegen ist, dass nicht alle Gegebenheiten der realen Welt mit Vektoren beschreibbar sind. Deshalb ist eine digitale Fotografie nicht als Vektorgrafik darstellbar. Aus diesem Grund sind Vektorgrafiken für den weiteren Verlauf dieses Berichts von keiner Bedeutung.

Eine Rastergrafik wird durch eine Vielzahl von rasterförmig angeordneten Punkten, denen eine Farbe zugeordnet ist, beschrieben. Jeweils ein Punkt wird auch als Pixel (abgekürzt *px*) bezeichnet.
Das Wort Pixel ist aus den beiden englischen Begriffen 'picture' (Bild) und 'element' (Element) zusammengesetzt.[8] In einem Pixel, zu deutsch auch Bildpunkt genannt, ist die Farbe, bzw. der Wert der Farbe, gespeichert, welche durch denjenigen Pixel dargestellt wird. Der Wert der Farbe ist abhängig vom Farbmodell und der Farbtiefe, welche für das Bild definiert ist. Eine Rastergrafik wird grundsätzlich durch ihre Höhe sowie Breite beschrieben. Diese Information wird in Pixeln angegeben. Die Standarddarstellung der Grösse einer Rastergrafik sieht wie folgt aus: *[Breite in pixe] x [Höhe in pixel]*, womit ein Bild mit einer Breite von 1024 und einer Höhe von 768 wie folgt beschrieben wird: *1024 x 768*. Die Höhe und die Breite des Bildes wird auch als Auflösung bezeichnet. Der Nachteil einer Rastergrafik ist die fehlende Möglichkeit zur verlustfreien Skalierung. So tritt beim Vergrössern einer Rastergrafik, wie in der Abbildung 2.3 zu erkennen ist, der Effekt der Verpixelung auf.

Abbildung 2.3: Original Bild links, Ausschnitt nach 5-facher Vergrösserung rechts

[8] Duden Rechtschreibung, `http://www.duden.de/rechtschreibung/Pixel`, aufgerufen am 23.11.2012.

Der Effekt der Verpixelung ist in der Bildverarbeitung soweit zu beachten, dass ein Bild, welches eine zu geringe Auflösung aufweist, möglicherweise schon von Anfang an eine Nutzung verunmöglicht.

2.2.2 Farbtiefe, Kanäle, Farbmodelle

Um den Aufbau eines Bildes zu verstehen, sind die Begriffe Farbtiefe, Farbmodell, und Kanal von eminenter Bedeutung.

Die *Farbtiefe* eines Bildes gibt an, wie viele bit pro Pixel verfügbar sind, um eine Farbe darzustellen. Die Farbtiefe wird in bit angegeben, woraus die maximale Anzahl Farben wie folgt berechnet wird: $2^{Farbtiefe[inbit]}$. Aus der Abbildung 2.4 ist ablesbar, wie viele Abstufungen für eine speziell gewählte Farbtiefe darstellbar sind.

Abbildung 2.4: Anzahl mögliche Abstufungen für 1, 2, 4 und 8 bit Farbtiefe [10]

Eine Bilddarstellung wird mittels Kanälen vollführt, wobei jeder Kanal eines Bildes die oben erwähnte Farbtiefe besitzt. Ein Kanal genügt sowohl zur Darstellung von *Graustufen-* sowie *Binärbildern*. Für Farbbilder werden mehrere Kanäle, abhängig vom für das Bild gewählten *Farbmodell*, benötigt. Ein Graustufenbild besteht, wie der Name schon andeutet, nur aus Pixeln, welche sich in der Helligkeit unterscheiden (siehe Abbildung 2.4). Ein Binärbild ist ein Bild mit einem Kanal mit einer Farbtiefe von 1 bit. Somit kann ein Pixel in einem Binärbild nur die Werte schwarz oder weiss annehmen. Der Farbverlauf für die 8 bit Farbtiefe in Abbildung 2.4 erweist sich als vom menschlichen Auge stufenlos, weshalb in der Praxis meist eine 8 bit Farbtiefe gewählt wird. Zusätzlich möglich wären natürlich auch Farbtiefen von 16 bzw. 32 bit.

Physikalisch gesehen existieren unendlich viele Farben[13]. Jedoch besitzt kein Computer oder optisch-digitales Instrument die Möglichkeit, eine unendliche Anzahl Farben darzustellen. Aus diesem Grund wurden mehrere *Farbmodelle* definiert. Ein Farbmodell bietet die Möglichkeit, die Farben des *Farbraums* zu beschreiben. Für die digitale Bildbearbeitung und den weiteren Einsatz für diese Arbeit ist Verständnis über die Farbmodelle Rot-Grün-Blau und Hue-Saturation-Value nötig.

[10] `http://upload.wikimedia.org/wikipedia/de/5/54/Farbtiefe.svg`, 25.11.12, Lizenz: gemeinfrei

[12] Foto aus eigenem Besitz

[13] `http://www.filmscanner.info/Farbmodelle.html`, 25.11.12

Abbildung 2.5: Farbbild links, Graustufenbild mitte, Binärbild rechts [12]

Mit RGB wird ein additives Farbmodell beschrieben, welches auf der Trichomatrischen Theorie[14] aufbaut. Diese Theorie besagt, dass man aus drei Lichtern, welche in Primärfarben leuchten, jeden beliebigen Farbton mischen kann. Diese Primärfarben können willkürlich gewählt werden. Die einzige Einschränkung ist jedoch, dass aus zwei Primärfarben nicht die dritte gemischt werden kann. RGB wählt die Primärfarben als Rot, Grün und Blau. Dies hat den Ursprung in der menschlichen Anatomie, da das menschliche Auge Rezeptoren für Rot, Blau und Grün besitzt[15]. Pro Primärfarbe wird in einem Bild ein Kanal benutzt, womit ein Bild, welches mit dem RGB Farbmodell beschrieben wird, zwingend 3 Kanäle besitzen muss. In der Abbildung 2.6 wird anhand eines dreidimensionalen Würfels die Funktionsweise der Farbfindung für die Farbe 80 (R) , 200 (G) , 130 (B) illustriert. Die Werte für R, G respektive B werden aus dem Wert der y-, x- respektive z-Achse gelesen. Der Wertebereich jeder der Achsen ist von 0 bis 255, womit 256 Werte dargestellt werden können. In diesem Beispiel wird mit einer Farbtiefe von 8 bit pro Kanal gearbeitet. Mit einem 8 bit pro Kanal RGB Modell lassen sich $(2^8)^3$ (= 16'777'216) Farben darstellen. Im RGB Modell haben Graustufen die Eigenschaft, dass die Farbanteile der drei Primärfarben den gleichen Wert haben oder als Formel R = G = B.

Einen anderen Ansatz eines Farbmodells stellt das HSV Farbmodell dar. HSV steht für Hue, Saturation, Value, wobei *Hue* den Farbton, *Saturation* die Farbsättigung und *Value* die Helligkeit darstellt. Der Farbton wird meist in einem 360° Farbkreis gewählt. Die Werte für Farbsättigung sowie Helligkeit können zwischen 0 und 1 annehmen[18]. Das HSV Farbmodell entspricht vielfach der Wahrnehmung

[14] `http://www.psychologie.uni-heidelberg.de/ae/allg/lehre/wct/w/w5_farbe/w530_trichromatische_theorie.htm`, 26.11.2012

[15] `http://de.wikipedia.org/wiki/Zapfen_(Auge)`, 15.12.12

[17] http://upload.wikimedia.org/wikipedia/commons/0/03/RGB_farbwuerfel.jpg , 25.11.12, Lizenz: public domain

[18] Andere Wertebereiche sind möglich, OpenCV verwendet die Wertebereiche H=0-180, S=0-255,V=0-255

Abbildung 2.6: Farbfindung im RGB Farbmodell[17]

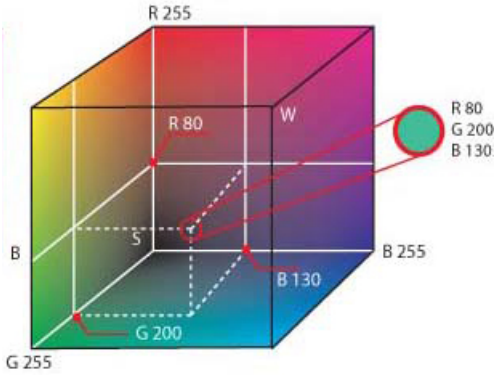


Tabelle 2.1: HSV Werte zu den drei Farben in Abbildung 2.7

Bild-Nr.	Farbton	Farbsättigung	Helligkeit
1	240	1	1
2	240	0.8	1
3	240	0.6	1
4	240	1	1
5	240	1	0.8
6	240	1	0.6

des Menschen. Denn in der Sprache ist beispielsweise die Rede von blau, mittelblau und hellblau. Im HSV Farbmodell werden diese drei Farben durch den gleichen Farbton beschrieben, jedoch durch unterschiedliche Werte für Farbsättigung und Helligkeit charakterisiert.

Abbildung 2.7: Farbvariationen mit 240° Farbton und unterschiedlichen Werten für Farbsättigung und Helligkeit[20]

In der Abbildung 2.7 sind Veränderungen der Sättigung sowie der Helligkeit bei Beibehaltung des Farbtons illustriert. Die Bilder mit der Nummer 1 und 4 stellen die gleiche Farbe dar, wie aus 2.1 zu entnehmen ist. Die Bilder 1 - 3 unterscheiden sich jeweils nur durch die Sättigung. Die Helligkeit ist auf dem Maximalwert eingestellt. Je kleiner die Sättigung, desto heller erscheint das resultierende Bild.

[20] Foto aus eigener Kreation

Umgekehrt sind die Bilder 4 - 6 entstanden, so beträgt die Sättigung den Maximalwert, die Helligkeit wird jedoch von links nach rechts immer kleiner, womit dunklere Farben erscheinen. Die Farben in den Bildern 1 - 6 werden vom Menschen jedoch immer noch als eine Derivation von blau angesehen. Im HSV Modell haben Graustufen die Eigenschaft, dass der Farbton sowie die Sättigung 0 betragen und die Helligkeit einen beliebigen Wert annehmen kann.
Die Farbmodelle HSV und RGB sind kompatibel, das heisst ein Bild in einem der zwei Farbmodelle lässt sich in ein Bild des anderen Farbmodells konvertieren. Nachfolgend ist die Umrechnung von RGB zu HSV in Theorie begleitend mit einem Beispiel aufgeführt.
Startwert RGB (8 bit pro Kanal): 255, 128, 64

Als Vorbedingung müssen die RGB Werte im Bereich zwischen 0 und 1 sein.

$$R,G,B \in [0,1] \tag{2.1}$$

Um die gemäss 2.1 vorgeschriebene Vorbedingung zu erfüllen, wird der R, G sowie B Wert durch den maximalen Wert des Wertebereichs dividiert. In diesem Beispiel ist dieser Wert 255. Daraus können folgende Startwerte berechnet werden:

$$R = \frac{R}{255}, G = \frac{G}{255}, B = \frac{B}{255} \tag{2.2}$$

Aus der Berechnung 2.2 ergeben sich folgende angepasste RGB Werte: 1, 0.502, 0.251. Zusätzlich wird der maximale sowie der minimale Wert aus R, G, B benötigt.

$$\begin{aligned} MAX &= \max(R,G,B) = R \\ MIN &= \min(R,G,B) = B \end{aligned} \tag{2.3}$$

Der aus dem RGB Wert nach der Konvertierung erhaltene Farbton wird je nach Situation mit Hilfe einer leicht abgeänderten Formel berechnet. Für den Fall, dass R = G = B entspricht, wird der Farbton sowie die Sättigung auf 0 gesetzt, was aus der bereits angesprochenen Eigenschaft von Graustufen herrührt. In den übrigen Fällen wird in Betracht gezogen, welcher Wert der Maximale und welcher der Minimale ist.

$$H = \begin{cases} 0, & [MAX = MIN] \\ 60 \times (0 + \frac{G-B}{MAX-MIN}), & [MAX = R] \\ 60 \times (2 + \frac{B-R}{MAX-MIN}), & [MAX = G] \\ 60 \times (4 + \frac{R-G}{MAX-MIN}), & [MAX = B] \end{cases} \tag{2.4}$$

Die Konstanten 0, 2, 4 sind so gewählt, da Rot den Farbton 0, Blau 120 und Grün 240 hat. Für das gewählte Beispiel kann, durch Anwendung der Formal 2.4, der Farbton als $60° \times 0 \frac{0.502-0.251}{1-0.251}$ = 20° berechnet werden.
Die Berechnung der Sättigung wird in zwei Fälle unterschieden. Ist der MAX Wert

0, d.h. R = G = B = 0, wird die Sättigung als 0 angegeben. Ansonsten ist die Sättigung als die Differenz zwischen MAX und MIN und dem aus einer Division mit MAX erhaltenen Wert bestimmt.

$$S = \begin{cases} 0, & [R = G = B = 0] \\ \frac{MAX-MIN}{MAX}), & [alle\,anderen\,Fälle] \end{cases} \tag{2.5}$$

Aus der Formel 2.5 kann für das Beispiel die Sättigung als $\frac{1-0.251}{1} = 0.75$ berechnet werden.
Die Helligkeit wird durch den maximalen Wert MAX dargestellt.

$$V = \ MAX \tag{2.6}$$

Nach Anwendung der Formel 2.6 ergibt sich für das Beispiel der Wert für die Helligkeit = 1.
Das Resultat der Umwandlung von RGB nach HSV für den als Beispiel gewählten RGB Wert ist unter 2.7 ersichtlich.

$$\left\{ \begin{array}{cc} R & 255 \\ G & 128 \\ B & 64 \end{array} \right\} \Longrightarrow \left\{ \begin{array}{cc} H & 20 \\ S & 0.75 \\ V & 1 \end{array} \right\} \tag{2.7}$$

Die umgekehrte Konvertierung, d.h. von HSV zu RGB ist im Verlauf der Arbeit nicht gebraucht worden, weshalb dieser Vorgang weggelassen wird.

2.3 Optik und Camera Obscura

Um an Bilder bzw. Filmsequenzen[21] zu gelangen, ist eine Kamera nötig. Dieser Abschnitt deckt die theoretischen Grundlagen zu Kameras, sowie die Kameras, welche während der Arbeit benutzt wurden, ab. Unter diesen Kameras befinden sich zwei Digital- sowie eine Wärmebildkamera. Der Begriff Kamera [*lat. camera*, dt. "Kammer"][22] stammt von einer Urform der heutigen Kamera, nämlich der *Camera Obscura* [dt. "dunkle Kammer"].

In der Abbildung 2.8 ist schematisch die Funktionsweise einer Camera Obscura dargestellt. Die Camera Obscura ist ein dunkler geschlossener Raum, bei dem in einer Seitenfläche eine kleine Öffnung vorhanden ist. Durch diese Öffnung fallen Lichtstrahlen ein und projizieren ein auf dem Kopf stehendes Abbild auf die

[21] Filmsequenzen bestehen aus mehreren Bildern, welche in einer angegebenen Rate nacheinander angezeigt werden. Die Rate wird als 'Frames per Second' fps angegeben.

[22] `http://dela.dict.cc/?s=camera`, 29.11.12

[24] `http://upload.wikimedia.org/wikipedia/commons/9/9d/Lochkamera_prinzip.jpg`, 29.11.12, `http://de.wikipedia.org/w/index.php?title=Datei:PinHoleCameraGeometry.png&filetimestamp=20070729150834`,06.12.12 Lizenz: GNU Free Documentation License

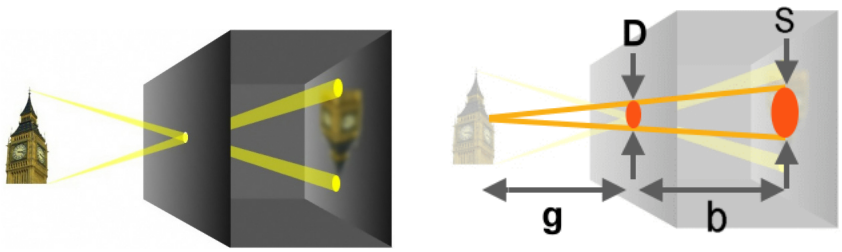


Abbildung 2.8: Funktionsweise Camera Obscura (links), geometrische Abhängigkeiten (rechts) [24]

Innenseite der dem Loch gegenüber liegenden Fläche. Jeder Punkt des Gegenstandes wird durch eine Scheibe S (Abbildung 2.8) auf der Abbildfläche dargestellt. Je grösser S ist, desto mehr Überlappungen entstehen, was einem unscharfen Abbild entspricht. Die Grösse der Scheibe S, in Abhängigkeit der Öffnungsgrösse D, kann mittels Strahlensatz wie in Formel 2.9 gezeigt, berechnet werden. Da eine klassische Lochkamera keine Linse besitzt, ist die Schärfe des Bildes ausschliesslich von der Lochgrösse D abhängig. Aus Gründen der minimal erforderten Lichtstärke kann D jedoch nicht beliebig klein gewählt werden, weshalb ein Abbild einer Lochkamera immer unscharf ist.

$$\frac{B}{G} = \frac{b}{g} \qquad (2.8) \qquad\qquad S(D) = \frac{D(b+g)}{g} \qquad (2.9)$$

Die Formel 2.8 definiert die erste Linsengleichung, welche besagt, dass die Grösse der Abbildung einzig durch die Gegenstandsgrösse G, des Abstands des Gegenstands zum Loch der Lochkamera (Gegenstandsweite g) sowie dem Abstand der Rückfläche zum Loch (Bildweite b) abhängig ist. Diese Gleichung lässt sich aus dem Strahlensatz ableiten und zeigt, dass die Wahl der Lochgrösse D keinen Einfluss auf die Bildgrösse hat.

Auf Basis der Grundlagen, welche mit dem Modell der Lochkamera erarbeitet sind, haben sich nach und nach die heutigen Kameras entwickelt. Die Unterschiede der heutigen Kameras zur Lochkamera werden im folgenden Abschnitt angesprochen.

2.4 Digitalkameras

Heutige Kameras, es wird hier nur auf Digitalkameras eingegangen, beruhen auf dem Grundprinzip der Lochkamera, jedoch haben sie, je nach Kameramodell, mehrere Linsen in einem *Objektiv* verbaut.
Anstatt einer Rückwand ist ein digitaler *Bildsensor* eingebaut, welcher die einfal-

lenden Strahlen als zweidimensionales Abbild aufnimmt. Mit Hilfe des Objektivs lässt sich die von der Lochkamera bekannte und unvermeidbare Unschärfe eliminieren und ein Motiv fokussieren. Als Fokussierung wird der Vorgang verstanden, bei welchem die Bildweite b anhand der Linsengleichung auf die Gegenstandsweite g angepasst wird. Dabei lässt sich jeweils nur ein Bereich von Distanzen von der Linse entfernt scharf stellen. Je nach Objektiv wird nur ein Bereich von Distanzen scharf dargestellt, der Rest wirkt verschwommen. Dieser Effekt wird als *Tiefenschärfe* bezeichnet. Durch die Blende, Brennweite sowie Aufnahmeentfernung wird die Tiefenschärfe beeinflusst. Für den Einsatz in dieser Arbeit ist eine hohe Tiefenschärfe von Vorteil, da nicht ein fest definierter Abstand zur Kamera scharf erscheinen soll, sondern möglichst der ganze Fussballfeldbereich,welcher von der Kamera abgedeckt wird.
In der Abbildung 2.9 wird der Einfluss der Tiefenschärfe auf eine Fotografie dargestellt. Im linken Bild ist der Fokus auf die Blumen in der Mitte gelegt, womit mit einer kleinen Tiefenschärfe die Blumen weiter hinten und vorne unscharf erscheinen. Im Bild rechts hingegen ist die Blende weiter geöffnet, womit die resultierende Tiefenschärfe grösser ist und das ganze Blumenfeld scharf erscheint.

Abbildung 2.9: Visualisierung Tiefenschärfe, links: kleine Tiefenschärfe, rechts: hohe Tiefenschärfe [26]

2.4.1 Vergleich von Videokameras

Für diese Arbeit sind eine GoPro HD Hero2 [GoP] sowie eine Mobotix M24M-Sec [AG] Kamera zur Verfügung gestellt worden. In der Tabelle 2.2 sind die technischen Daten der beiden Kameras aufgelistet. Die Sortierung der Daten beruht auf der Wichtigkeit für den Einsatz im Rahmen dieser Arbeit.

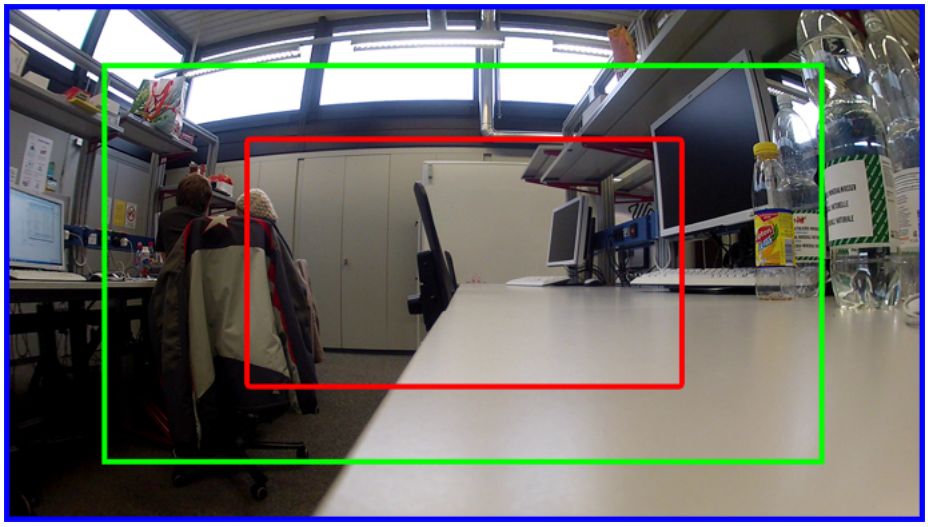
[26] `http://de.wikipedia.org/wiki/Sch%C3%A4rfentiefe`, zuletzt besucht 18.12.12, Lizenz: Gemeinfrei

Tabelle 2.2: Technische Übersicht GoPro und Mobotix

	Mobotix M24M-Sec	GoPro HD Hero2
Auflösung Framerate	2048 x 1536, 20 fps 1280 x 960, 30 fps 704 x 576, 30 fps 640 x 480, 30 fps	1920 x 1080, 30fps 1280 x 960, 30/48fps 1280 x 720, 30/60fps 838 x 480, 60/120fps
Bildwinkel	90°	170° (alle Modi) 127° (1080,720,838) 90° (1080,720)
Speisung	Power over Ethernet	Akku (ca.2.5h Laufzeit)
Preis	798€	279 Fr. (ohne WiFi) 415 Fr. (mit WiFi)
Online Videoverarbeitung	Ja	Nein
Masse (HxBxT) [mm]	140 x 230 x 190	42 x 60 x 30
Gewicht [g]	750	94 (ohne Gehäuse) 167 (mit Gehäuse)

Der *Bildwinkel* einer Kamera beschreibt den Winkel, welcher von der Kamera eingeschlossen wird. Der Bildwinkel ist abhängig von der Grösse des Bildsensors und der Bildweite.[27] Daraus lässt sich schliessen, dass der Bildwinkel für ein Objektiv, sofern es sich nicht um ein Zoomobjektiv handelt, gleichbleibend ist. Die GoPro realisiert die drei verschiedenen Bildwinkel, indem, je nach gewähltem Modus, nur ein Teil des vom Sensor erfassten Abbilds verwendet wird und auf die Ausgangsauflösung skaliert wird. Der Unterschied der drei Bildwinkel ist unter Abbildung 2.10 abgebildet. Dabei ist der 90° Bildwinkel rot, der 127° grün sowie der 170° blau umrahmt.

Abbildung 2.10: Vergleich der drei verfügbaren Bildwinkel der GoPro



[27] http://www.elmar-baumann.de/fotografie/lexikon/bildwinkel.html, zuletzt besucht 19.12.12

Die Auflösung bietet zusammen mit dem Bildwinkel Grundlage zu geometrischen Berechnungen. Als Überblicksgrafik soll Abbildung 2.11 dienen. Dabei steht α für den Bildwinkel der Kamera. $c1$ steht für die Breite des Bereichs, welcher im Abstand d von der Kamera abgedeckt werden soll. Mit $c2$ ist die Breite des Bereiches angegeben, welcher im Abstand d+b von der Kamera abgedeckt wird.
Die Tabellen 2.3 - 2.5 zeigen die berechneten Werte für d, $c2$. Daraus kann abgeleitet werden, durch welche Anzahl Pixel eine Länge von 1m an der Stelle von $c2$ dargestellt wird. Tabelle 2.3 zeigt die Werte, wenn die Kamera das gesamte Spielfeld (100m x 64m) abdecken soll. Für ein halbes Spielfeld dient die Tabelle 2.4 und ein Viertel Spielfeld wird durch die Tabelle 2.5 dargestellt.

Abbildung 2.11: Geometrische Abhängigkeit Blickwinkel

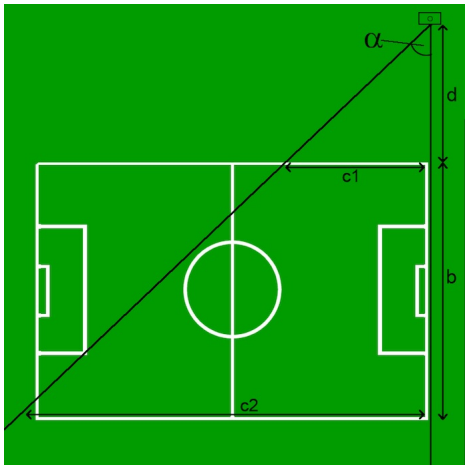


Tabelle 2.3: Auswirkung der Bildwinkel. $c1$ = 100m, b = 64m

	170°	127°	90°
d	4.4m	25m	50m
$c2$	1564m	357m	228m
pixel/m bei c2	1.23	5.38	8.42

Tabelle 2.4: Auswirkung der Bildwinkel. $c1$ = 50m, b = 64m

	170°	127°	90°
d	2.2m	12.5m	25m
$c2$	1513m	307m	178m
pixel/m bei c2	1.27	6.25	10.79

Aus der Angabe der Pixel pro Meter (basierend auf einer Auflösung von 1920 x 1080) ist ersichtlich, dass die Bildwinkel 127° und 170° für den angestrebten

Tabelle 2.5: Auswirkung der Bildwinkel. $c1$ = 50m, b = 32m

	170°	127°	90°
d	2.2m	12.5m	25m
$c2$	782m	180m	114m
pixel/m bei c2	2.46	10.67	16.84

Zweck keine Option sind.

2.4.2 Wärmebildkamera

Eine spezielle Form von Kameras bilden Wärmebildkameras. Die Funktionsweise dieser Kameras beruht auf einer Gegebenheit der Physik, welche besagt, dass die Temperatur eines Körpers durch die thermische Bewegung seiner Teilchen bestimmt wird. Je tiefer die Temperatur des Körpers, desto langsamer bewegen sich die Teilchen. Ab einer gewissen Temperatur stehen die Teilchen still, die Moleküle und Atome sind eingefroren, womit keine tiefere Temperatur mehr möglich ist. Diese Temperatur wird der *absolute Nullpunkt* genannt und befindet sich bei $\sim$-273.15° [28]. Jeder Körper über dem absoluten Nullpunkt emittiert Wärmestrahlung, welche im infraroten Bereich liegt und somit für das menschliche Auge unsichtbar ist. Je grösser die Temperatur eines Körpers, desto höher wird die Intensität der Wärmestrahlung.
Eine Wärmebildkamera misst die Intensität der Wärmestrahlen und setzt diese mittels dem in Formel 2.10[29] illustrierten Stefan-Boltzmann-Gesetz in Temperaturen um. Aus der Formel ist ersichtlich, dass eine um 2° höhere Temperatur T eine sechzehnfache Intensität der Wärmestrahlung nach sich zieht.

$$W = \sigma \times T^4 \quad \begin{array}{l} W = \text{Intensität der Strahlung} \\ \sigma = \text{Stefan-Boltzmann-Konstante} = 5,6710^{-8}\,\frac{W}{m^2K} \\ T = \text{gemessene Temperatur in Kelvin} \end{array} \tag{2.10}$$

Wärmestrahlung wird von Glas reflektiert, weshalb die Linsen des Objektivs einer Wärmebildkamera nicht aus Glas gefertigt werden können. Deshalb sind Objektive von Wärmebildkameras aus einkristallinen Halbleitermaterialien wie Germanium und Zinkselenid gefertigt[30].
Anders als bei einer Digitalkamera speichert der Sensor einer Wärmebildkamera pro Bildpunkt nur einen Wert, nämlich die Intensität der Wärmestrahlung. So entstehen grundsätzlich Graustufenbilder, bei denen ein hellerer Grauton eine höhere

[28] `http://www.physik.uzh.ch/data/peter/PhysikD/PraktikumPhysikD/AbsoluterNullpunkt.pdf`, 5.2.3

[29] `http://www.flir.com/cs/emea/de/view/?id=41523`, zuletzt besucht 17.12.12

[30] `http://de.wikipedia.org/wiki/W%C3%A4rmebildkamera`,06.12.12

Temperatur identifiziert. Erst im Nachhinein wird bei gängigen Wärmebildkameras das Graustufenbild eingefärbt. Dies, da der Mensch Farbunterschiede leichter unterscheiden kann als Helligkeitsunterschiede.

Während dieser Arbeit ist eine Wärmebildkamera des Typs FLIR T620[FLI] zum Einsatz gekommen. Diese Wärmebildkamera besitzt eine Wärme- sowie eine 'normale' Digitalkamera, womit jeweils ein Wärmebild sowie ein 'normales' Digitalbild gespeichert wird. Die T620 bietet die Möglichkeit, Videoaufnahmen mit 30fps aufzunehmen, wobei bei dieser Variante nur ein Wärmebildvideo erstellt wird. Die maximale Auflösung der Bilder beträgt 640 x 480. Der Bildwinkel beträgt mit dem Objektiv 25°, liesse sich jedoch mittels Ersatzobjektiven vergrössern. Die aufgenommenen Wärmebilder können später mit der Applikation 'Flir Tools' bearbeitet werden. Für Videosequenzen besteht diese Möglichkeit nicht, weshalb es nötig ist, den Bereich der Temperatur bei der Aufnahme bereits richtig eingestellt zu haben.

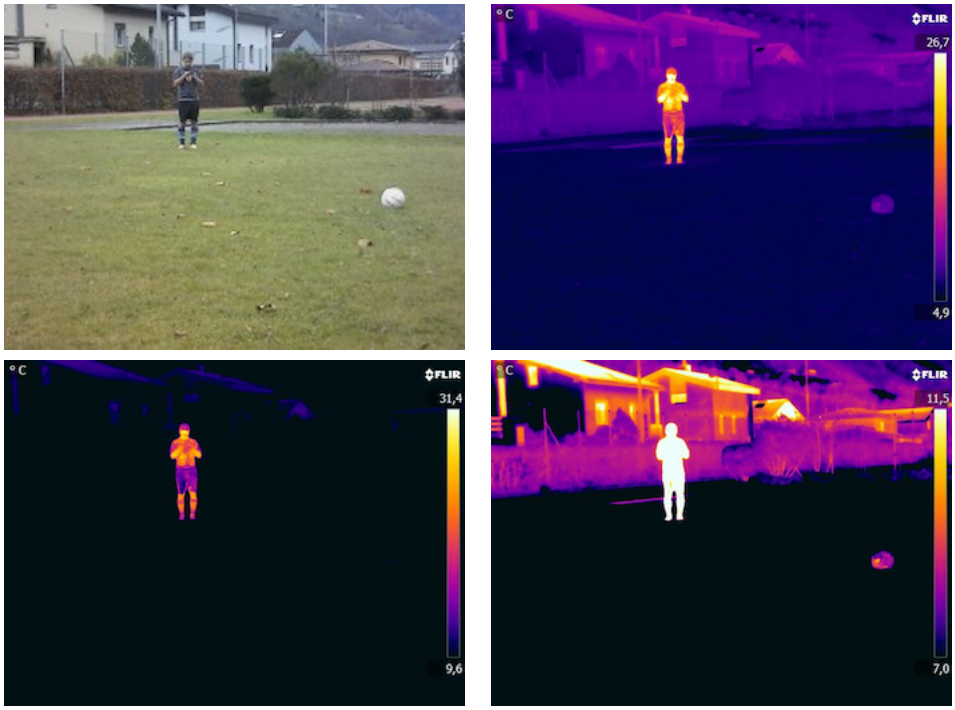


Abbildung 2.12: Normales Bild, Wärmebild mit automatischem Bereich, Wärmebild mit unvorteilhaftem Wärmebereich, Wärmebild mit optimalem Wertebereich (o.l. nach u.r.)

In den Bildern 2.12 ist oben links das mit der 'normalen' Kamera der T620 aufgezeichnete Bild sichtbar. Die übrigen drei Bilder sind aus dem gleichen Wärmebild entstanden, mit einer Anpassung des Wärmebereichs mit Hilfe der 'Flir Tools' Applikation. Der Wertebereich im Bild unten rechts ist als optimal anzusehen, weil der Spieler und der Ball durch Entfernen der weissen Bereiche unterschieden wer-

den können. Die zwei anderen Wertebereiche sind nicht optimal gewählt, da der Spieler durch unterschiedliche Farben dargestellt wird.

2.5 Folgerungen

Aus den theoretischen Überlegungen der vorherigen Abschnitte lassen sich einige Folgerungen ziehen.

- Als Kameramodell wird die GoPro der Mobotix vorgezogen. Die wichtigsten Gründe dazu sind einerseits die durch den Akku gegebene Unabhängigkeit von einer Verkabelung des Drehorts. Andererseits bietet die GoPro mit dem WiFi BacPac und einer App[31] eine ansprechende Lösung um ohne viel Aufwand und ohne einen separaten Computer mit Hilfe eines iPhones die Aufnahme zu starten bzw. zu stoppen.
- Als Bildwinkel für die Aufnahme der Videosequenzen wird der 90° Modus gewählt. Die beiden anderen Bildwinkel würden einen zu grossen Bereich abdecken, als dass der Fussball noch in genügend grosser Ausprägung auf dem Bild erscheint.
- Die Auflösung wird mit 1920 x 1080 gewählt, damit der Ball auch in einiger Entfernung sichtbar bleibt. Zusätzlich wird so der Effekt der Verpixelung durch nachträgliches Skalieren vermieden.
- Für die Abdeckung des ganzen Feldes werden mindestens vier Kameras benötigt, welche in der diagonalen Verlängerung der Eckpunkte des Fussballfeldes situiert werden. Diese Position erlaubt es, die Kamera näher am Spielgeschehen zu haben. Mittels Stativen sollten sich die Kameras auf einer dem Spielfeld erhöhten Position befinden. Damit kann eine Kamera einen Bereich von 50 x 32m abdecken.
- Farb- bzw. musterbasierte Ansätze wie unter [VJ01] erwähnt, können für Bilder nicht gewählt werden, da der Fussball keiner Regelung bezüglich Farbe und Muster unterliegt.
- Für die Unterscheidung Spieler / Ball kann, im Falle der Wärmebildkamera, ein einfaches Farbthresholding angewendet werden. Dies ist jedoch nur mit einem optimal eingestellten Temperaturbereich möglich.

[31] vgl. Anhang

Kapitel 3

Bildverarbeitungsalgorithmen

Für die Arbeit sind Grundlagen der häufigsten Bildverarbeitungsalgorithmen nötig, welche in diesem Kapitel behandelt werden. Alle der nachfolgend erwähnten Algorithmen sind zum Einsatz gekommen, entweder in einer der Demoapplikationen oder dem Prototyp.

3.1 Kantenerkennung

Als Kantenerkennung (engl. *edge detection*) wird der Versuch bezeichnet, flächige Bereiche, welche sich in Helligkeit unterscheiden, zu segmentieren. Für diese Aufgabe haben sich im Laufe der Zeit mehrere Verfahren entwickelt, wobei in dieser Arbeit der Operator von John F. Canny [Can86] erläutert wird. Um Helligkeitsunterschiede zu detektieren, benutzt der Canny Operator die in Formel 3.1 dargestellten Matrizen G_x und G_y.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3.1}$$

G_x wird für die Kantenerkennung in x-Richtung benutzt, G_y hingegen in die y-Richtung. Diese Matrizen werden jeweils über jedes Pixel des Bildes angewendet, womit für jedes Pixel jeweils ein G_x sowie ein G_y Wert entsteht. Aus diesen beiden Werten kann mittels der unter 3.2 gezeigten Formel G als Stärke der Kante, sowie θ als Winkel der Kante bestimmt werden. θ wird dann auf die möglichen Werte 0°, 45°, 90° oder 135° gerundet.

$$\begin{aligned} G &= \sqrt{G_x^2 + G_y^2} \\ \theta &= \arctan(\tfrac{G_y}{G_x}) \end{aligned} \tag{3.2}$$

Des Weiteren werden Pixel, welche nicht als Teil einer Kante angesehen werden, ignoriert. Dies geschieht mit Hilfe der als *Non-maximum suppression* bekann-

ten Methode. Dabei wird der Wert eines Pixels mit jedem seiner 8 benachbarten Pixel verglichen. Nachbarn, welche nicht in der Richtung der Kante liegen und einen geringeren Wert aufweisen, wird der Grauwert auf Null gesetzt. Ziel der *non-maximum suppression* ist, dass eine Kante maximal ein Pixel breit ist.

Der letzte Schritt besteht darin, mit Hilfe von Schwellwerten die schlussendlichen

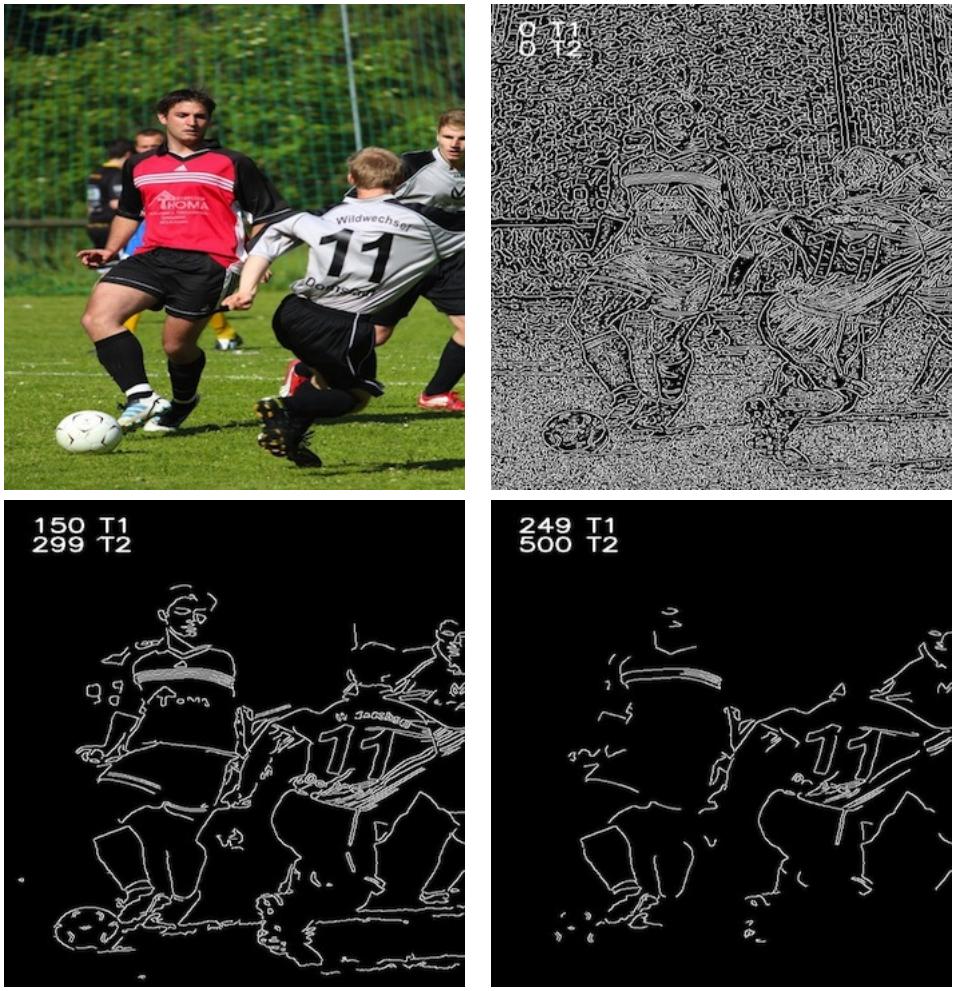


Abbildung 3.1: Ausgangsbild, Kantenbild mit T_1 = T_2 = 0, Kantenbild mit T_1 = 150, T_2 = 299, Kantenbild mit T_1 =249,T_2 = 500 (von o.l. nach u.r.)

Kanten zu erhalten. Der Canny Operator benutzt zwei Schwellwerte T_1 und T_2, welche im Verhältnis $T_1 < T_2$ zueinander stehen. Ein Pixel wird als Kante gekennzeichnet, falls

- der Wert des Pixels $> T_2$ ist.
- der Wert des Pixels $> T_1$ und in Kantenrichtung ein Pixel mit Wert $> T_2$ vorhanden ist.

Aus der Abbildung 3.1 ist ersichtlich, dass mit höher gewählten T_1 und T_2 die

Anzahl und Länge der Kanten vermindert sind. Canny schlägt ein Verhältnis von T_2 zu T_1 zwischen 3:1 und 2:1 vor.

3.2 Hough Transformation

Die Hough Transformation wird in der digitalen Bildverarbeitung für die Findung von Formen wie Linien oder Kreisen benutzt. Der Algorithmus dient als Transformation eines Punktes im x, y - Koordinatensystem in einen Parameterraum [Ped07]. Der Parameterraum ist durch die Form des gesuchten Objektes definiert. Er agiert auf einem durch Kantenerkennung erhaltenen Binärbild.

Eine Linie in einem kartesischen x,y Koordinatensystem kann mit der Funktion $y = ax + b$ beschrieben werden, wobei a die Steigung und b den y-Achsenabschnitt definiert. Da a bei zur x-Achse im Lot stehenden Linien unendlich ist, wird der Parameterraum nicht durch a und b dargestellt. Stattdessen wird die Normalform der Linie, welche durch den Winkel θ und die Länge τ definiert wird, verwendet. Für jeden Kantenpunkt im Ausgangsbild wird durch die Formel $\tau = x \cdot \cos(\theta) + y \cdot \cos(\theta)$ die Repräsentation jeder möglichen Linie, die den Punkt durchlaufen kann, im Parameterraum eingetragen. In Abbildung 3.2 wird dieser Schritt für den Punkt im Originalen Bild x = 8 und y = 6 illustriert.

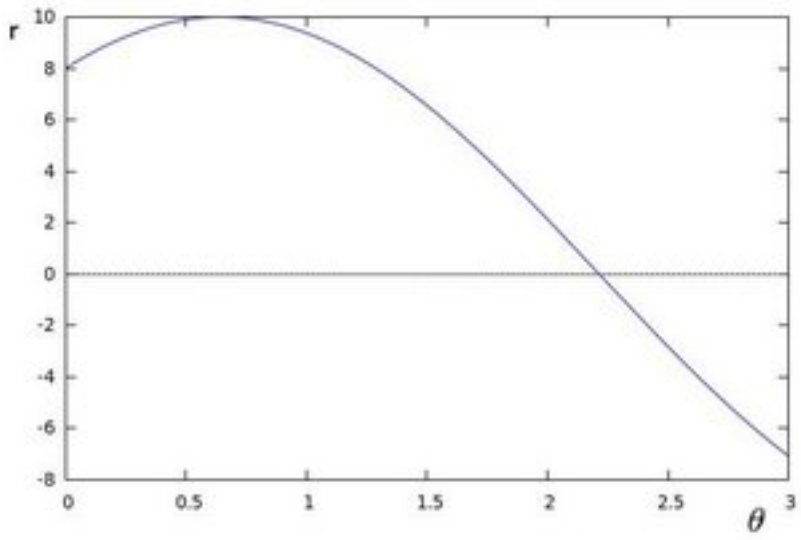
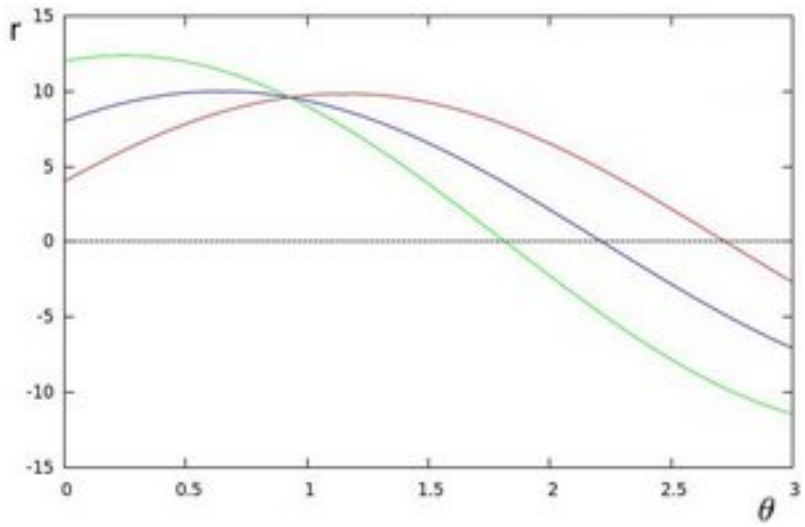


Abbildung 3.2: Alle möglichen Linien eines Punktes, dargestellt im Parameterraum (aus [Wiknt])

Der Parameterraum dient als Akkumulator, bei dem für jeden möglichen (θ, τ) Wert gespeichert wird, wie viele Punkte nach der Transformation darauf abgebildet werden. Je höher der akkumulierte Wert eines Punktes im Parameterraum, d.h. je mehr Schnittpunkte von Linien auf einem Punkt im Parameterraum liegen, desto

wahrscheinlicher ist es, dass dieser Punkt eine Linie darstellt. In der Abbildung 3.3 sind zusätzlich die möglichen Linien für die Punkte (9, 4) und (12, 3) in den Parameterraum übertragen worden. Dabei ist zu erkennen, dass die Linie um $\tau = 9.6$ und $\theta = 0.925$ eine Linie darstellt, auf welcher die 3 Punkte aus dem ursprünglichen Bild liegen.

Abbildung 3.3: Alle möglichen Linien von drei Punkten, dargestellt im Parameterraum (aus [Wiknt])



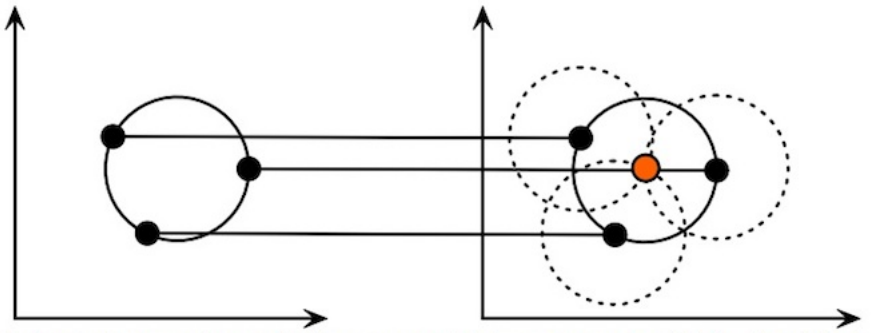
Ein Kreis wird im Gegensatz zu einer Linie mit drei Parametern beschrieben, nämlich x, y und r, wobei x und y für die Koordinate des Mittelpunktes stehen. r definiert den Radius. Die drei Parameter zur Beschreibung eines Kreises implizieren einen dreidimensionalen Parameterraum. Ist der Radius der gesuchten Kreise bekannt, kann der Parameterraum auf eine zweidimensionale Auswertung reduziert werden.
Im Gegensatz zur Linie können die Parameter des Kreises direkt in den Parameterraum überführt werden. Der Algorithmus besteht nun darin, für jeden Punkt im Kantenbild einen Kreis im Parameterraum zu zeichnen. Ein Kreis mit einem bestimmten Radius R und Mittelpunkt (a,b) kann mit der Gleichung 3.3 beschrieben werden. Wird für θ der Wertebereich von 0 - 360 Grad durchlaufen, zeichnen die Punkte x, y den Umfang eines Kreises.

$$\begin{aligned} x &= a + R\cos(\theta) \\ y &= b + R\sin(\theta) \end{aligned} \tag{3.3}$$

Analog zur Hough Transform mit einer Linie wird bei der Kreissuche der Punkt im Parameterraum mit den meisten Schnittpunkten als wahrscheinlichster Kreismittelpunkt definiert.
In der Abbildung 3.4 ist der Ablauf der Transformation anhand eines Kreises mit bekanntem Radius gezeigt, links den Bildraum und rechts den Parameterraum. Die gestrichelten Kreise stellen den um den gewählten Punkt gezeichneten Kreis dar.

Der orange Punkt im Parameterraum repräsentiert den gefundenen Kreismittelpunkt.

Abbildung 3.4: Kreis in Originalbild links, Detektierter Mittelpunkt in Parameterraum rechts (aus [Car07])



Each point in geometric space (left) generates a circle in parameter space (right). The circles in parameter space intersect at the (a, b) that is the center in geometric space.

3.3 Vordergrundextraktion [1]

In einer Videosequenz, welche mit einer fix installierten Kamera aufgenommen ist, interessiert bei der Balldetektion nur der Vordergrund. Der Hintergrund wird durch die statischen Bildelemente dargestellt, nämlich das Feld, Linien und Tore, der Vordergrund durch die sich bewegenden Elemente, Spieler, Schiedsrichter oder Ball.
Um den Vordergrund vom Hintergrund zu trennen, gibt es mehrere Ansätze. Einer davon beruht darauf, dass der Hintergrund ohne Vordergrundelemente als Bild vorhanden ist. Aus einem Bild mit Vordergrund kann durch die Bildung der absoluten Differenz der beiden Bilder der Hintergrund entfernt werden. Dieser Vorgang ist in Formel 3.4 mathematisch dargelegt. Für jedes Pixel wird die Differenz des Pixelwertes berechnet. Für jeden Kanal des Bildes geschieht dies unabhängig. [2]

$$
\begin{aligned}
&dst(I) = saturate(|src1(I) - src2(I)|) \\
&\\
&dst = \text{Bild der absoluten Differenz} \\
&src1 = \text{Bild ohne Vordergrundelemente} \\
&src2 = \text{Bild mit Vorder- und Hintergrundelementen} \\
&I = \text{Index, steht für jedes Element des Arrays}
\end{aligned}
\tag{3.4}
$$

[1] [Re11, p272-277]

[2] `http://docs.opencv.org/modules/core/doc/operations_on_arrays.html`, 15.12.2012

Für die Videosequenzen, welche im Rahmen eines Fussballspiels anfallen, genügt der Ansatz der absoluten Differenz nicht, da meist kein Bild ohne Vordergrundelemente verfügbar ist. Zusätzlich ist die Bildung der absoluten Differenz anfällig auf sich verändernde Lichtverhältnisse. Änderungen der Lichtverhältnisse können durch Wolken oder durch die Positionsänderung der Sonne während eines Spiels verursacht werden.

Um die Lichtveränderungen zu berücksichtigen ist es nötig, ein dynamisches Modell des Hintergrundbildes zu erstellen. Dieses kann durch die Berechnung eines *running average* der Bilder erhalten werden. Der *running average* wird mittels Formel 3.5 berechnet. Dabei steht p_t für den aktuellen Pixelwert und u_{t-1} für den aktuellen *running average* Wert. α wird die Lernrate genannt. Damit wird angegeben, wie stark der Einfluss von p_t auf den resultierenden *running average* ist. Je höher die Lernrate gewählt wird, desto schneller passt sich das Hintergrundmodell an Änderungen in den Bildern an. Um den Vordergrund zu erhalten, muss die Differenz vom aktuellen Bild zum Hintergrundmodell berechnet und ein absoluter Schwellenwert angewendet werden, ab welcher Differenz ein Pixel als Vordergrund klassifiziert wird.

$$\mu_t = (1-\alpha)\mu_{t-1} + \alpha p_t \tag{3.5}$$

Als Erweiterung der gezeigten *running average* Variante ist die '*Improved Adaptive Gaussian Mixture Model for Background Subtraction*'[Ziv04] Methode entwickelt worden. Dabei werden pro Pixel ein oder mehrere *running average* Werte geführt. So werden für ein Pixel, welches häufig zwischen zwei Werten wechselt, zwei *running average*'s gespeichert. Zusätzlich zum *running average* wird in dieser Methode auch eine *running variance* berechnet. Die Berechnung ist unter der Formel 3.6 aufgezeigt. Dabei sind die Variablen gleichbedeutend wie in der Formel 3.5. Zusätzlich steht σ_t^2 für die neu berechnete *running variance* sowie σ_{t-1}^2 für den alten Wert der *running variance*.
Mit Hilfe des *running average* sowie der *running variance* wird ein Gaussisches Modell für ein Pixel geformt. Nun kann geschätzt werden, wie wahrscheinlich ein Pixelwert zu diesem Modell passt. Dadurch wird der absolute Schwellenwert nicht mehr benötigt, denn es wird pro Gaussisches Modell entschieden, was ein angebrachter Schwellenwert ist. Wird ein Gaussisches Modell nicht genügend angesprochen, wird es nicht mehr als Teil des Hintergrundes angesehen. Wird jedoch ein Pixelwert als Vordergrund deklariert, wird dafür ein Gaussisches Modell angelegt. Dieses wird schliesslich als Hintergrund angesehen, falls es genügend oft angesprochen wird.

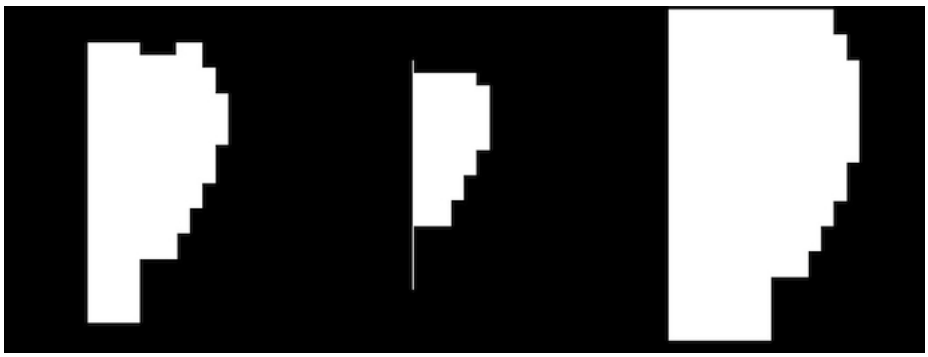
$$\sigma_t^2 = (1-\alpha)\sigma_{t-1}^2 + \alpha(p_t - \mu_t^2) \tag{3.6}$$

3.4 Erosion und Dilatation [3]

Das durch Anwenden der Vordergrundextraktion erhaltene Binärbild, bei dem der Vordergrund als '1' und der Hintergrund als '0' definiert ist, weist meist einige Schwächen auf. Zum einen entstehen Konturen, welche nicht durchgehend verbunden sind, jedoch eigentlich zum gleichen Objekt gehören. Zum anderen verschmelzen Konturen, welche eigentlich unterschiedliche Objekte darstellen. Um die beiden Fälle auszubessern, wird in der digitalen Bildverarbeitung meist eine *Erosion* bzw. *Dilatation* angewendet.

Bei der Erosion sowie der Dilatation wird ein Eingangsbild A zusammen mit einem *Kernel* B verarbeitet. B kann verschiedene Formen darstellen, wobei es nachfolgend als Rechteck angenommen wird. B besitzt einen Ankerpunkt und amtet bei der Erosion als *'local minimum'*, bei der Dilatation als *'local maximum'* Operator. Der Kernel B wird für jedes Pixel in A angewendet, wobei der Wert des Ankerpunktes in A durch den minimalen (Erosion) bzw. maximalen (Dilatation) Wert, welcher vom Kernel bedeckt wird, ersetzt wird.
In Abbildung 3.5 wird der Effekt der Erosion und der Dilatation visuell dargestellt. Ersichtlich ist, dass mittels Erosion helle Bereiche verkleinert werden. Für die Dilatation gilt das umgekehrte.

Abbildung 3.5: Originalbild links, Erosion mitte, Dilatation rechts [5]



$$dst(x,y) = \max_{(x',y'):element(x',y')\neq 0} src(x+x',y+y') \tag{3.7}$$

$$dst(x,y) = \min_{(x',y'):element(x',y')\neq 0} src(x+x',y+y') \tag{3.8}$$

Die Formeln 3.7 und 3.8[6] zeigen die mathematische Berechnung der Dilatation bzw. Erosion. Sie unterscheiden sich lediglich durch den *max* Gebrauch bei der

[3] [BK08, p115-117]

[5] Bild selbst erstellt, dann mit 20 x 20 Kernel in 2 Iterationen erodiert bzw. dilatatiert

[6] `http://docs.opencv.org/modules/imgproc/doc/filtering.html`, zuletzt besucht 18.12.12

Dilatation (Formel 3.7) und dem *min* Gebrauch bei der Erosion (Formel 3.8).

Die Erosion wird benutzt, um Rauschen zu entfernen; die Dilatation hingegen um Löcher im Vordergrund zu schliessen.

3.5 Bildtransformation

Wird ein Gebiet mit einer Kamera aufgenommen, ist die Perspektive verschoben. So wird aus einem Rechteck in Original (Abbildung 3.6 rechts) auf einem Bild als Trapez dargestellt (Abbildung 3.6 links).

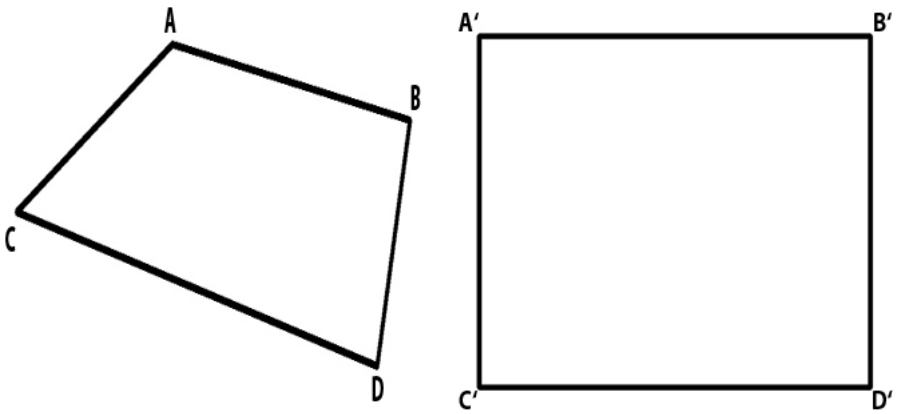


Abbildung 3.6: Umrandung eines Rechtecks aus Videobild, umgerechnete Punkte rechts [8]

Um das Bild auf die Realität abzubilden, ist es nötig, eine geometrische Bildtransformation anzuwenden. Eine Transformation lässt sich durch die Formel 3.9[9] beschreiben. Für jeden Punkt im Ausgangsbild wird der Punkt im resultierenden Bild berechnet.

$$dst(x,y) = src(f_x(x,y), f_y(x,y) \tag{3.9}$$

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = map\_matrix \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.10}$$

Für eine Transformation der Perspektive müssen in jedem der zwei Bilder vier Punkte definiert sein, welche in der unter Formel 3.10 gezeigten Beziehung zueinander stehen. Die map_matrix wird dann mit Hilfe der vier Punkte berechnet und kann für jede Szene welche aus der gleichen Position gefilmt ist angewendet

[8] Bilder selbst erstellt, bzw. aus Videobildern extrahiert

[9] `http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html`, zuletzt besucht 18.12.12

Abbildung 3.7: Gefilmte Szene links, mit map_matrix umgerechnetes Bild rechts

werden. Die Umformung der Perspektive anhand eines selbst gefilmten Bildes ist unter Abbildung 3.7 zu sehen. Dabei stellt das linke Bild die gefilmte Szene dar. Das rechte Bild illustriert das Resultat nach der Transformation der Perspektive.

Kapitel 4

Realisierung

In diesem Kapitel wird die Realisierung des Prototyps sowie der Demoapplikationen erläutert.
Alle Applikationen, welche im Rahmen dieser Arbeit entstanden sind, sind in C++ geschrieben. Zusätzlich wird auf die OpenCV Bibliothek zurückgegriffen.

4.1 OpenCV

OpenCV stellt eine Programmbibliothek dar, welche über 2500 Algorithmen zur Bildverarbeitung zur Verfügung stellt[1]. OpenCV ist in C++/C geschrieben, stellt Schnittstellen für C++, Python sowie C zur Verfügung. OpenCV wird unter der BSD-Lizenz veröffentlicht, womit die Nutzung für akademische sowie kommerzielle Benutzung kostenfrei erfolgt. Für diese Arbeit ist die Version 2.4.2 verwendet worden.

OpenCV behandelt Bilder standardmässig im Blau-Grün-Rot-Farbmodell. Dieses Farbmodell entspricht dem RGB-Farbmodell, nur die Reihenfolge der Primärfarben ist umgekehrt.

OpenCV hat seine Funktionalitäten in mehrere *.hpp Files aufgeteilt, von denen, je nach Anwendungsfall und Methoden, die benutzt werden, andere in den eigenen Code eingebunden bzw. included werden müssen.

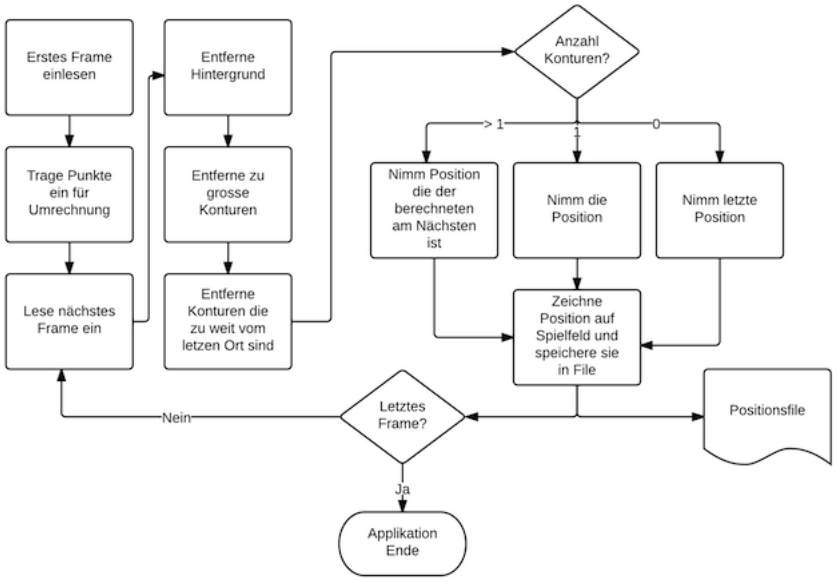
4.2 Demonstrator

Der Demonstrator ist die Applikation, welche eine Videosequenz verarbeitet, den Ball detektiert und die auf dem Videobild gefundende Koordinate des Balles auf eine 2D Repräsentation des Fussballfeldes überträgt. Der Algorithmus, der den Ablauf des Demonstrators bestimmt, ist in Abbildung 4.1 zu sehen. Die Hauptüberlegung des Algorithmus ist, dass alles, was sich auf dem Feld bewegt, aber

[1] `http://opencv.willowgarage.com/wiki/`, 26.11.12

kein Mensch (Schiedsrichter, Linienrichter, Spieler) ist, den Ball darstellt.

Abbildung 4.1: Flussdiagramm des Prototyp Algorithmus



Mit dem Start des Algorithmus wird das erste Bild der angegebenen Videosequenz eingelesen. Dieses Bild wird zusammen mit dem in Abbildung 2.1 gezeigten Fussballfeld angezeigt. Damit wird der Benutzer aufgefordert, vier Punkte aus dem Videobild anzuwählen und auf vier Punkte auf dem Fussballfeld zu übertragen.[2] Abbildung 4.2 zeigt die Wahl der Punkte anhand von Screenshots. Sind auf beiden Bildern die vier Punkte markiert, wird mittels Drücken einer beliebigen Taste der Algorithmus fortgesetzt. Mit den zur Verfügung gestellten Punkten wird die Umrechnungsmatrix berechnet, welche zur Abbildung von Bildpunkten auf Feldpunkte gebraucht wird.

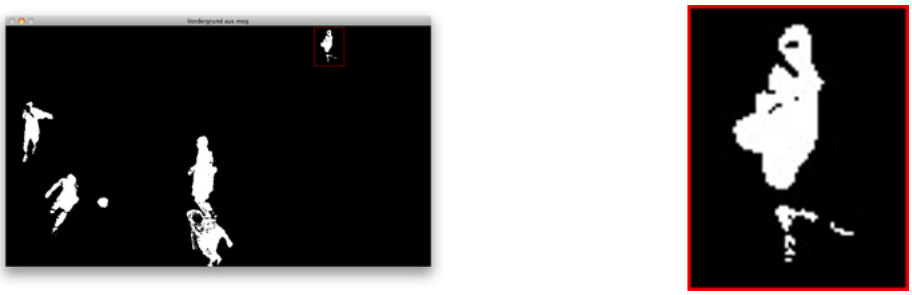
Nach der Spezialbehandlung des ersten Frames wird nun Frame für Frame in einer Schleife verarbeitet. Dies so lange, bis entweder der Benutzer die Verarbeitung mit Drücken der 'q' Taste verlässt, oder das letzte Bild der Videosequenz erreicht ist.
Als erster Schritt wird das gesamte Bild geglättet (engl. *blurred*, *smoothed*), um Rauschen zu reduzieren. Dieser Vorgang ist eine in der Bildverarbeitung oft gebrauchte Operation, weshalb auf die Nennung im Algorithmus Flussdiagramm verzichtet wird.
Nach der Glättung wird der Hintergrund mittels der *'Improved adaptive Gausian*

[2] Die Reihenfolge, wie die Punkte angeklickt werden, ist zu beachten.

Abbildung 4.2: Wahl der 4 Referenzpunkte auf schematischem Spielfeld (links) und Videobild (rechts)

mixture model for background subtraction'-Methode entfernt. Die Methode liefert eine Binärmaske. Der Bereich des Vordergrundes wird weiss dargestellt und der Hintergrund schwarz. Abbildung 4.3 zeigt auf dem linken Bild den extrahierten Vordergrund.

Abbildung 4.3: Extrahierter Vordergrund (links), vergrösserte unterbrochene Kontur (rechts)



Das so erhaltene Vordergrundbild enthält typischerweise eine gewisse Anzahl von weissen Pixeln, welche als Rauschen interpretierbar sind. Zusätzlich sind sich bewegende Spieler meist nicht als durchgehende Kontur dargestellt, sondern weisen Lücken auf oder die Extremente sind abgetrennt. Abgetrennte Extremente sind in der Abbildung 4.3 rechts gezeigt. Um diese unerwünschten Effekte auszumerzen, wird eine Variation von Erosionen und Dilatationen auf das Bild angewendet. Die Erosionen werden jeweils mit einem quadratisch geformten Kernel durchgeführt, die Dilatationen hingegen mit einem rechteckigen Kernel mit einem Verhältnis Höhe zu Breite von 2:1. Dies, da die Unterteilung eines Spielers in mehrere Konturen in Richtung der y-Achse geschieht. Zusätzlich wird damit vermieden, dass durch die Dilatation der Ball mit dem Schuh des Spielers zu einer Kontur verschmilzt.

Aus dem morphologisch veränderten Bild werden die Konturen bestimmt und mittels Ausschlussverfahren vermindert. Hauptkriterium zur Verminderung der Konturen ist das Verhältnis von Höhe zur Breite des Rechteckes, welches um die Kontur gelegt wird. Dabei werden alle Konturen, welche ein Verhältnis grösser als 2:1 (bzw. 1:2) herausgefiltert [Xia06]. Zusätzlich werden Konturen, bei welchen die Höhe oder die Breite nicht im Bereich des Erwarteten ist, aus der Maske entfernt. Schliesslich werden alle Konturen entfernt, welche von der letzten Position des Balles zu weit entfernt sind. Das Resultat der Entfernung der unpassenden Kontu-

Abbildung 4.4: Vordergrund nach Anwendung der morphologischen Operatoren (links), dadurch zusammengewachsene Kontur (rechts)

ren ist in Abbildung 4.5 dargestellt.

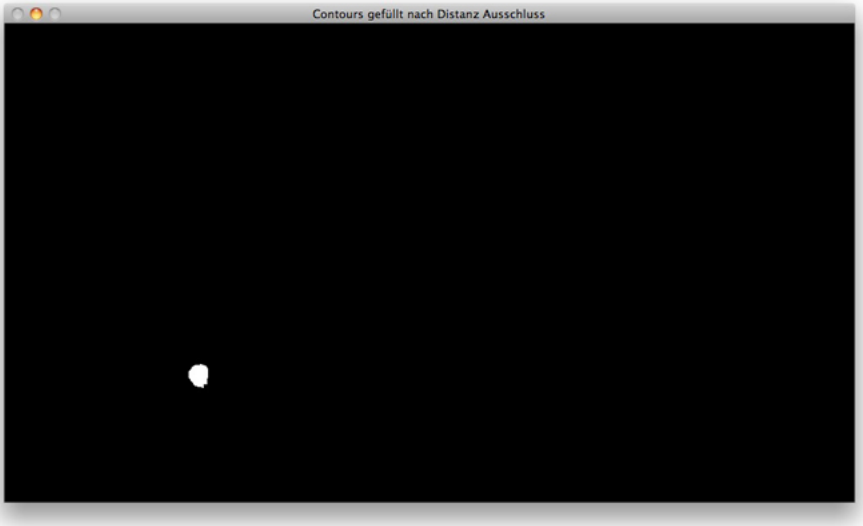


Abbildung 4.5: Binärmaske nach Entfernung der nicht als Ball klassifizierten Konturen

Das weitere Vorgehen des Algorithmus ist der Test auf die Anzahl der Konturen.
Ist mehr als eine Kontur übrig geblieben, wird diejenige Kontur als Ballposition angesehen, welche der vorhergesehenen Position am Nächsten ist. Die vorhergesehene Position berechnet sich aus der letzten (p_{t-1}) und vorletzten (p_{t-2}) Position des Balles. Nun wird der Vektor von p_{t-2} zu p_{t-1} errechnet und zu p_{t-1} addiert.
Ist eine Kontur vorhanden, wird diese als Ballposition angesehen. Falls keine Kontur vorhanden ist, wird die letzte Position weiterverwendet und die maximale Entfernung für die nächste Iteration um den Faktor 1.2 erhöht.

Die gefundene Ballposition wird dann mittels der am Anfang der Applikation bestimmten Umrechnungsmatrix auf die schematische Darstellung des Feldes umgerechnet. Abbildung 4.6 zeigt die schematische Darstellung des Feldes mit den Ballpositionen. Rote Punkte bedeuten, dass sich der Ball ausserhalb des Feldes befindet, aber trotzdem detektiert wurde. Die einzelnen Punkte sind mit Linien ver-

bunden, falls die Detektion nahtlos aneinander gelungen ist. Falls auf einem Bild der Ball nicht detektiert worden ist, wird keine Linie gezeichnet.

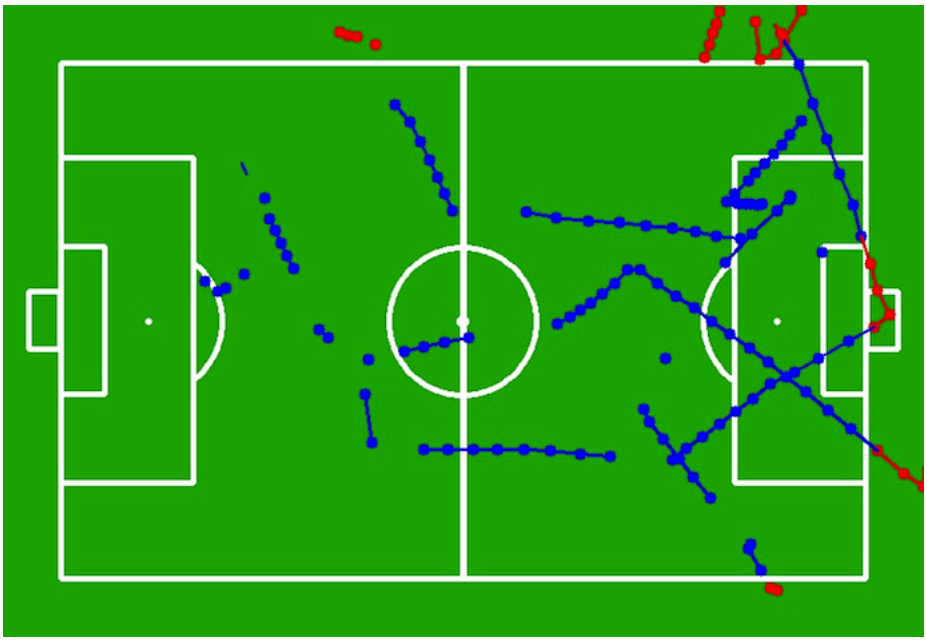


Abbildung 4.6: Aussehen des schematischen Spielfeldes nach 250 verarbeiteten Bildern

4.3 Anforderungen an das Videobild

Der Algorithmus hat aufgrund der Nutzung der Vordergrundextraktion einen hohen Anspruch an eine fixe Position der Kamera. Schon eine kleine Bewegung der Kamera kann den extrahierten Vordergrund beträchtlich stören. Dies ist in Abbildung 4.7 visualisiert. Es ist gut zu erkennen, dass die Linien auf dem Hallenboden als Vordergrund erscheinen, da der Wert des Pixels bei einer sich bewegenden Kamera oft zwischen dem Farbwert der Linie und derjenigen des Bodens wechselt.

4.4 Demoapplikationen

Die zur Erarbeitung der Bildverarbeitungsalgorithmen entstandenen Demoapplikationen, es sind deren drei, haben denselben funktionalen Aufbau. Sie erwarten ein Bild als Input und wandeln es mit dem Algorithmus, welcher durch diese Applikation veranschaulicht wird, um. Das resultierende Bild wird auf dem Bildschirm angezeigt. Die drei Demoapplikationen illustrieren den HoughCircles- und Canny-Algorithmus, sowie die Generierung einer Binärmaske durch Wählen der geeigneten HSV Schwellenwerte.

Abbildung 4.7: Extrahierter Vordergrund bei sich minim bewegender Kamera

Kapitel 5

Testing

Dieses Kapitel widmet sich dem Test des Algorithmus. Der realisierte Algorithmus ist mit Bildmaterial eines Hallenfussballtrainings getestet worden. Das Bildmaterial ist mit einer GoPro HD Hero2 mit einer Auflösung von 1920 x 1080 sowie einer Bildrate von 30fps aufgenommen worden. In diesem Training wird 3 gegen 3 gespielt. Die Hallenmasse betragen 22 x 11.6m [1], was eine Fläche von 255m^2 ergibt. Dies entspricht $\frac{1}{25}$ eines standardmässigen Fussballfeldes. Es sind 6 Personen anstatt 23 (22 Spieler, 1 Schiedsrichter) auf dem ganzen Feld, was einem Verhältnis von ca. $\frac{1}{4}$ entspricht.

5.1 Performance

Für die Performance des Algorithmus dient die Angabe, wie viele Bilder pro Sekunde verarbeitet werden können. Getestet wurde auf einem *MacBook Pro* mit einem *3.06 GHz Intel Core 2 Duo* Prozessor und *4 GB 1067 Mhz DDR3* Arbeitsspeicher. In Tabelle 5.1 ist der Durchsatz der Applikation angegeben, wenn jeweils Bild für Bild der Videosequenz nur in den Speicher gelesen wird. OpenCV kann 44.4 Bilder einer 1920 x 1080 Videosequenz pro Sekunde öffnen und in den Hauptspeicher laden.

Tabelle 5.1: Durchsatz der Applikation ohne Durchlaufen des Algorithmus

Anzahl Bilder	Laufdauer	Bilder pro Sekunde	Laufdauer für ein Spiel
5000	113	44.25	3661s

Wird der gesamte Algorithmus durchlaufen, ergeben sich die Daten in der Tabelle 5.2. *Anzahl Bilder* steht für alle Bilder der Videosequenz, welche geöffnet worden sind. Der Wert *i* sagt aus, dass jedes *i*-te Bild durch den Algorithmus ver-

[1] `http://2reserve.ch/Raum/Customizations/GlarusNord/Content/Objektbeschriebe/Objektbeschrieb_Schulhaus%20Rauti%20Oberurnen.pdf`, zuletzt besucht 20.12.2012

arbeitet wird. Die restlichen Bilder werden von der Applikation gelesen, aber nicht verarbeitet. Die Laufdauer pro Spiel wird gerechnet, indem die Anzahl Bilder für ein Spiel als *90 Minuten · 60 Sekunden · 30 Bilder pro Sekunde* angenommen wird. Daraus ergibt sich eine idealisierte Bildanzahl von 162000 für ein Fussballspiel. Die Halbzeitpause und Nachspielzeit wird ignoriert. Die Werte für i sind bewusst mit 3 und 6 gewählt worden, da die cnlab/HSR Tracer Anwendung die Positionen mit einer Frequenz von 5Hz aufnimmt.

Tabelle 5.2: Durchsatz der Applikation mit Bearbeiten jedes i Bild

Anzahl Bilder	i	Laufdauer	Bilder pro Sekunde	Laufdauer für ein Spiel
1000	1	392s	2.55	63529s
3000	3	424s	7.07	22914s
6000	6	468s	12.83	12626s

5.2 Erkennungsrate

Für die Erkennungsrate des Algorithmus sind effektiv 1000 Bilder analysiert worden. i ist mit 3 gewählt worden. Dabei ist jedes der Bilder nach der Detektion klassifiziert worden. Folgende fünf Klassifikatoren stehen zur Verfügung:

- Verdeckt: Der Ball ist von einem Spieler ganz verdeckt.
- Ausserhalb: Der Ball befindet sich nicht auf diesem Bild, ist also ausserhalb des von der Kamera aufgenommenen Bereiches.
- Nicht detektiert: Der Ball ist nicht detektiert worden, obwohl er sich auf dem Bild befindet und nicht verdeckt wird.
- Falsch detektiert: Der Algorithmus hat eine Detektion vorgenommen, welche jedoch nicht den Ball darstellt.
- Korrekt detektiert: Der Ball ist korrekt detektiert worden.

Tabelle 5.3: Klassifizierung der Detektion in 1000 Bildern

Klassifikation	absolute Anzahl	relativer Anteil
Verdeckt	67	6.7%
Ausserhalb	274	27.4%
Nicht detektiert	166	16.6%
Falsch detektiert	54	5.4%
Korrekt detektiert	439	43.9%

Die Resultate dieses Tests sind in Tabelle 5.3 sichtbar. Die Detektionsrate erreicht 43.9%, wobei dieser Wert nicht als endgültige Detektionsrate angesehen wird. Denn von den 1000 analysierten Bildern ist der Ball in 274 Fällen nicht auf

dem Bild vorhanden und in 67 Bildern vollumfänglich von einem Spieler verdeckt. Daraus ergeben sich die korrigierten Werte, die in Tabelle 5.4 dargestellt sind. Von den 1000 ursprünglichen Bildern werden 274 und 67 Bilder abgezählt. Dies ergibt eine neue Anzahl von total 659 Bildern.

Tabelle 5.4: Klassifizierung der Detektion in 659 Bildern

Klassifikation	absolute Anzahl	relativer Anteil
Nicht detektiert	166	25.2%
Falsch detektiert	54	8.2%
Korrekt detektiert	439	66.6%

Damit erhöht sich die Zahl der korrekten Detektionen auf 66.6%. Von 659 Ballpositionen sind 449 korrekt. Zusätzlich ist zu beachten, dass mit diesen Detektionen eine Ballpositionsfrequenz von 10Hz erreicht wird. Dies ist das Doppelte der gewünschten Frequenz.

Kapitel 6

Ergebnisse und Schlussfolgerungen

Die mit dieser Machbarkeitsstudie abgeklärte Frage, wie die Erkennung und Positionsbestimmung eines Fussballs am besten zu realisieren ist, wird in diesem Kapitel beantwortet.
Die Aufnahme der Spielsituationen ist mit einer Kamera aufzunehmen, welche eine Auflösung von 1920 x 1080 Pixel bietet. Für die Bildrate der Kamera sind im Anbetracht des Berechnungsaufwandes 30 Bilder pro Sekunde das obere Maximum. Eine geringere Bildrate hätte eine geringere Datenmenge und eine schnellere Verarbeitung einer Videosequenz zur Folge.
Die Berechnung der Position im Feld erfolgt mittels Referenzpunkten, die durch Schnittpunkte der Spielfeldlinien definiert werden. Diese müssen von Hand auf das Spielfeld übertragen werden, da eine Erkennung der Ecke, in welcher sich eine Kamera befindet, nicht möglich ist.
Der Algorithmus zur Ballerkennung der im Demonstrator arbeitet, ist auf das Testset aus der Turnhalle gut eingestellt. Fraglich ist, wie er sich bei einem Test bei realen Bedingungen, sprich Videosequenzen von richtigen Fussballspielen behaupten wird.
Der Einsatz von Wärmebildkameras für die Unterscheidung von Mensch und Ball ist zwar ein vielversprechender Ansatz, jedoch müssten weitere Tests vorgenommen werden. Die Testphase mit der Wärmebildkamera fand nämlich bei bewölktem Wetter statt. Würde die Sonne scheinen, würde sich der Boden - und damit auch der Ball - schnell aufheizen. Damit könnte der Temperaturunterschied zwischen Mensch und Ball nicht mehr so klar ersichtlich sein. Zusätzlich würden schattige Bereiche auf dem Spielfeld zu grossen Unterschieden im Wärmebild führen.

6.1 Ausblick

Für einen produktiven Einsatz der Ballerkennungstechnologie sind noch weitere Implementierungen nötig. Um die Positionsfehler des Fussballes zu korrigieren, wenn dieser in der Luft ist, muss jeder Bereich des Feldes von mindestens zwei Kameras abgedeckt werden. Dadurch kann eine dreidimensionale Abbildung des ganzen Spielfeldes generiert werden. Aus diesem Anwendungsfall entstehen weitere Arbeitsfelder, welche die zeitliche Synchronisation der Kameras sowie das Abstimmen der Kamerapunkte aufeinander beinhalten.
Zusätzlich ist die Nachbearbeitung der Ballpositionen ein Schritt, welcher die Qualität der Positionen erhöhen könnte. Mittels Iteration über die Ballpositionen könnten Ausreisser besser detektiert und im Nachhinein ausgeschlossen werden.

Abbildungsverzeichnis

Glossary

BGR
Die Kanäle sind im Vergleich zum RGB-Farbmodell in genau anderer Reihenfolge. 36

bzw.
Abkürzung für beziehungsweise. 13–15, 19, 26, 33, 34, 38

fps
Frames per Second, bezeichnet die Anzahl Frames (=Bilder), welche pro Sekunde aufgenommen werden. 22, 25, 42

HSV
Farbmodell welches auf dem Farbton (Hue), der Farbsättigung (Saturation) und dem Value (Helligkeit) beruht. 15–18, 40

OpenCV
Open-Source Computer Vision Library. Stellt eine Bibliothek mit Bildverarbeitungsalgorithmen dar. 2, 6, 7, 10, 16, 36

RGB
Additives Farbmodell mit den drei Primärfarben Rot, Grün und Blau.. 15–17, 47

z.B.
Abkürzung für zum Beispiel. 14

Literaturverzeichnis

[AG] Mobotix AG. MobotixM24M-sec. http://www.mobotix.com/ger_CH/Produkte/Kameras/Allround-M24. letzter Zugriff am 21.12.2012.

[BK08] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. Software that sees. O'Reilly Media, Incorporated, 2008.

[Can86] F. John Canny. A Computational Approach to Edge Detection. 8(6):679–698, 1986.

[Car07] Chester F. Carlson. Lecture 10: Hough Circle Transform. http://www.cis.rit.edu/class/simg782/lectures/lecture_10/lec782_05_10.pdf, 2007.

[DPPH12] cnlab AG, Dr. Prof. Peter Heinzmann. Administratives zu Studien- und Bachelorarbeiten. http://www.cnlab.ch/kurse/SABA/, 2012. letzter Zugriff am 21.12.2012.

[FLI] Flir T620 & T640 datasheet. http://www.flir.com/uploadedFiles/Thermography_USA/Products/Product_Literature/flir-t620-datasheet.pdf. letzter Zugriff am 19.12.2012.

[Fus] Schweizerischer Fussballverband. Fussball-Spielregeln. http://www.football.ch/de/Portaldata/1/Resources/dokumente/flippingbook/SFV_Fussballregeln_d_Neu/HTML/index.html. letzter Zugriff am 19.12.2012.

[GoP] GoPro. GoPro HD Hero2. http://de.gopro.com/cameras/hd-hero2-outdoor-edition/#specs. letzter Zugriff am 21.12.2012.

[Ped07] Simon Just Kjeldgaard Pedersen. Circular Hough Transform. http://opencf.pcg.ull.es/redmine/projects/opencfr/repository/changes/branches/tesis/celulas/documentos/Simon_Pedersen_CircularHoughTransform.pdf, 2007.

[Re11] R.L. Re. *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 Recipes to Master This Library of Programming Functions for Real-time Computer Vision*. Quick answers to common problems. Packt, 2011.

[VJ01] Paul Viola and Michael Jones. Robust Real-time Object Detection. In *International Journal of Computer Vision*, 2001.

[Wiknt] OpenCV Wiki. Hough Line Transform. `http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html`, Unbekannt. letzter Zugriff am 20.12.12.

[Xia06] Limeng Xiao. Football Player Tracking, 2006.

[YPIK90] HK Yuen, J Princen, J Illingworth, and J Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71 – 77, 1990.

[Ziv04] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, pages 28–31, Washington, DC, USA, 2004. IEEE Computer Society.

Kapitel 7

Anhang

7.1 Projektplan

In dieser Arbeit kam kein Projektplan zum Einsatz, wie er von gewöhnlichen Software Projekten bekannt ist. Der Projektplan bestand aus wöchentlichen Zielen bzw. Milestones.

Projektstart 17.09.2012
Projektwoche 1 24.09.2012

- Inbetriebnahme Mobotix & GoPro HD Hero2
- Erstkontakt mit Präsident FC Linth 04
- Erstellung Projektplan

Projektwoche 2 01.10.2012

- OpenCV Installation und erste Inbetriebnahme
- Es kann ein Ball auf einem Bild erkennt werden
- Beschaffung Videomaterial Fussball
- Fortsetzung von Projektwoche 2

Projektwoche 3 08.10.2012

- OpenCV - Es kann ein Video verarbeitet werden

Projektwoche 4 15.10.2012

- OpenCV - Dokumentation mit Videobeispielen (evt. Mit GUI Anwendung)

Projektwoche 5 22.10.2012

- OpenCV - Es kann ein Ball auf einem Video getrackt werden

Projektwoche 6 29.10.2012

- Bester Standort für die Kameras ist ermittelt.

- Minimale Anzahl Kameras ist ermittelt.
- Kameramodell ist ermittelt

Projektwoche 7 05.11.2012

- Präsentation Co-Referent

Projektwoche 8 - 9 12.11.2012

- Implementation Algorithmus

Projektwoche 10 26.11.2012

- Wärmebildkamera testen

Projektwoche 11 03.12.2012

- Plakat an P. Heinzmann abgegeben
- Abstract an P. Heinzmann abgeben

Projektwoche 12 10.12.2012

- Dokumentation fertiggestellt, bis auf Korrekturen

Projektwoche 13 17.12.2012

- Korrektur Dokumentation
- Dokumentation drucken, binden
- Abgabe CD vorbereiten

Projektende 21.12.2012

Die geplante Zeiteinteilung, welche auf der CD in digitaler Form vorhanden ist, zeigt mehr Ist- als Sollstunden auf. Vor allem die Dokumentation hat mehr Zeit in Anspruch genommen als ursprünglich geplant. Dies hat seinen Ursprung darin, dass die Arbeit alleine verfasst wurde.

7.2 Externe Libraries, Installation und Entwicklungsumgebung

Auf der CD befinden sich zwei gepackte Dateien (*.zip). In der einen befinden sich die drei Demoapplikationen, in der anderen der Demonstrator. Die Source-Files liegen in nicht kompiliertem Zustand vor. Um diese kompilieren zu können, ist die Installation der OpenCV Library notwendig. Für eine Anleitung zur Installation von OpenCV sei auf `http://opencv.willowgarage.com/wiki/InstallGuide` verwiesen. Die in dieser Arbeit benutzte Version von OpenCV ist 2.4.2.
Als Entwicklungsumgebung ist zu Beginn der Arbeit die Xcode IDE genutzt worden. Aufgrund der grösseren Vertrautheit mit den Eclipse IDE's, wurde zum Schluss der Arbeit hin mit der Eclipse CDT IDE entwickelt.

7.3 Persönlicher Bericht Daniel Stucki

Nach der erfolgreichen und interessanten Studienarbeit im Frühjahrssemester 2012 war es für mich von Anfang an klar, dass ich die Bachelorarbeit wieder mit Herrn Bentele und der Betreuung von Prof. Dr. Heinzmann schreiben werde.
Die Ernüchterung kam jedoch zeitgleich mit dem Projektstart, denn Herr Bentele hatte die Zulassungskriterien zur Bachelorarbeit nicht gemeistert und somit stand ich vor der Entscheidung, die Arbeit um ein Semester zu verschieben, oder sie alleine in Angriff zu nehmen. Ich habe mich dann für Variante zwei entschieden.
Die Arbeit, Ballerkennung mittels Kameras, stellte für mich ein reizvolles Thema dar, da ich aufgrund meines Teilzeitstudiums schon genügend 'normale' Software Projekte gemacht habe und nun für einmal etwas völlig Neues machen wollte. Doch wie gross die Euphorie am Anfang auch war, mit C++ hatte ich seit meinem zweiten Semester an der HSR nichts mehr zu tun. Und so wurde nur schon der Task, eine Hello World Applikation mit Hilfe der OpenCV Bibliothek zu erstellen, zu einem zeitraubenden Abenteuer. Nur schon die exotischen Compilermeldungen und die, im Vergleich zur Java Eclipse IDE, weniger aussagekräftige Xcode IDE (später wechselte ich auf die C++ Version der Eclipse) führten zu grossen Zeitinvestitionen.
Daraus resultierte auch das schleppende Vorankommen, welches jedoch gegen Ende des Projektes immer schneller aufgeholt werden konnte. Dazu kamen immer mehr Erfolgserlebnisse, wie zum Beispiel die Möglichkeit, eine Wärmebildkamera im Wert eines besseren Kleinwagens für ein Wochenende zu benutzen oder die gelungene Umrechnung von Bildpunkten auf die Spielfeldfläche.
Abschliessend ist zu sagen, dass ich bei einem nächsten Mal die Arbeit wahrscheinlich nicht mehr alleine schreiben würde, vor allem der Dokumentation wegen. Denn ob man alleine ist oder zu zweit, der Umfang der Dokumentation ist immer der gleiche. Ausserdem entstehen in einem Team immer Gespräche, welche auf die Qualität des Ergebnisses sicher keinen negativen Einfluss haben. Vor allem bei dieser Arbeit, bei der mögliche Ansätze zur Erkennung des Balles gesucht sind, hätten solche Gespräche wahrscheinlich einen guten Input geliefert.
Trotz diesen negativen Eindrücken überwiegt für mich das Positive. Ich konnte fundierteres Wissen der C++ Programmiersprache erarbeiten und einige Grundkenntnisse der digitalen Bildverarbeitung aufschnappen.

7.4 GoPro HD Hero2

Für die HD Hero2 Actionkamera des Herstellers GoPro (nachfolgend GoPro genannt) ist ein so genanntes WiFi BacPac erhältlich. Als BacPac wird ein Zubehör für die GoPro bezeichnet, welches wie ein Rucksack hinten auf die GoPro montiert wird. Das Aussehen und ein Beispiel eines montierten BacPacs sind auf der Abbildung 7.1 illustriert. Dieses WiFi BacPac ermöglicht es, die GoPro via WiFi-

Fernbedienung, oder seit Mitte Oktober 2012, mit einer App[1] fürs iPhone und iPad zu bedienen.

Abbildung 7.1: GoPro mit demontiertem WiFi BacPac links,GoPro mit montiertem WiFi PacPac rechts [3]

Via die Fernbedienung oder die App können alle Funktionen der GoPro verändert bzw. bedient werden, auch ohne physischen Kontakt zur Kamera. Sogar das Ein- und Ausschalten der Kamera kann damit bewerkstelligt werden. Einzig das WiFi BacPac muss eingeschaltet sein.

GoPro App (v. 1.0.40)

Die im Apple App Store seit Mitte Oktober 2012 erhältliche App zur Fernsteuerung der GoPro bietet neben den bereits genannten Funktionalitäten auch ein Live-Vorschaubild. Dies ist insofern nützlich, da die Kamera kein Display besitzt, mit welchem überprüft werden kann, welcher Bereich zurzeit anvisiert ist. Jedoch muss dazu angefügt werden, dass die Vorschau um einige Sekunden verzögert erscheint. Damit die App die Kamera erkennt, muss das WiFi BacPac an der Kamera eingeschaltet werden. Das BacPac stellt darauf ein eigenes WiFi zur Verfügung. Die SSID des WiFi's sowie dessen Passwort ist über die Software „GoPro CineForm Studio" einstellbar. Dazu muss das BacPac via USB an den Computer angeschlossen werden. Mit dem zur Verfügung gestellten WiFi muss das iPhone verbunden werden. Wird dann die App gestartet, kann die Kamera gesteuert werden. Aus der

[1] `https://itunes.apple.com/ch/app/gopro-app/id561350520?mt=8&affId=2157876&ign-mpt=uo%3D4` , 30.11.12

[3] Foto aus eigenem Besitz

Abbildung 7.2 kann der Funktionsumfang der App abgeschätzt werden. So ist links das Vorschaubild, rechts davon kann eine Aufnahme gestartet bzw. gestoppt werden, die Einstellungen der Kamera verändert werden und mittels des Sliders die GoPro ein- bzw. ausgeschaltet werden.

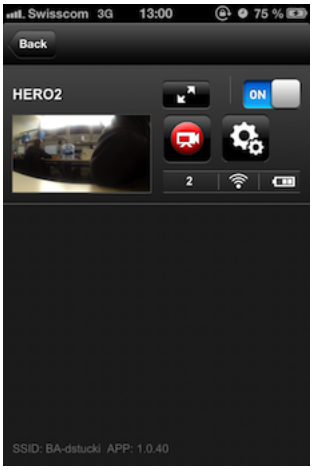


Abbildung 7.2: Screenshot aus der GoPro App fürs iPhone [5]

Cherokee Webserver

Aus der Überlegung, dass das WiFi BacPac das Vorschaubild an die App sendet, oder die App das Vorschaubild beim BacPac abfragt, kann in Betracht gezogen werden, dass es auch von einem Notebook aus möglich sein muss, dieses Vorschaubild zu erhalten. Dies würde eine Live-Verarbeitung der gefilmten Szenen ermöglichen. Aus der WiFi Infoseite auf dem iPhone kann eruiert werden, dass das BacPac die IP 10.5.5.9 hat (Infopunkt „Router“). Mit einer Portscan App (Fink) für das iPhone konnten die Ports 80 sowie 8080 als offen detektiert werden.
Ein gewöhnliches Notebook kann sich mit dem WiFi des BacPac's verbinden. Indem die Adresse des BacPac's – 10.5.5.9 – mit dem offenen Port 8080 in einen Webbrowser eingegeben wird, kann auf die Ordnerstruktur des auf dem BacPac laufenden cherokee[6] Webservers zugegriffen werden. Über die Directory Struktur können die auf der SD-Karte der Kamera gespeicherten Files auf das Notebook übertragen werden.
Für eine Live-Übertragung reicht der Durchsatz des gebotenen WiFi's jedoch nicht. Mit Einsatz eines Kommandozeilentools lftp[7] wird ein Durchsatz von durchschnittlich 300KB/s erreicht.

[5] Foto aus eigenem Besitz
[6] `http://www.cherokee-project.com/`, 30.11.12
[7] `http://lftp.yar.ru/`, 30.11.12

7.5 Kontaktdaten

Dieser Abschnitt enthält die E-Mail-Adressen der an der Arbeit beteiligten Personen. Über diese Adressen können die Personen auch nach der Zeit an der HSR erreicht werden.

Daniel Stucki, Student
daniel@stucki.me

Prof. Dr. Peter Heinzmann, Betreuer
peter.heinzmann@cnlab.ch

7.6 Inhaltsverzeichnis der beigelegten CD

TechnischerBericht.pdf

01_Code

- DemoApp.zip
- FussballVideoDetection.zip

02_Sitzungsprotokolle

- 20120917_Protokoll-1_BA_VideoDetection_V0.1.docx
- 20120924_Protokoll-2_BA_VideoDetection_V0.1.docx
- 20121001_Protokoll-3_BA_VideoDetection_V0.1.docx
- 20121011_Protokoll-4_BA_VideoDetektion_V0.1.docx
- 20121018_Protokoll-5_BA_VideoDetektion_V0.1.docx
- 20121029_Protokoll-6_BA_VideoDetektion_V0.1.docx
- 20121102_Protokoll-7_BA_VideoDetektion_V0.1.docx
- 20121108_Protokoll-8_BA_VideoDetektion_V0.1.docx
- 20121126_Protokoll-9_BA_VideoDetektion_V0.1.docx
- 20121204_Protokoll-10_BA_VideoDetektion_V0.1.docx

03_Projektmanagement

- Zeitplan.xlsx

04_Dokumente_Org

- HoughCircleInOpenCV.docx
- Poster.pptx
- Zwischenpraesentation.pptx

04_Dokumente_PDF

- HoughCircleInOpenCV.pdf
- Poster.pdf
- Zwischenpraesentation.pdf

7.7 Poster

Für diese Arbeit wurde ein Poster zusammengestellt. Auf der beiliegenden CD befindet sich dieses Poster im .pdf Format.